CP/M®

CON MP/M™

Rodnay Zaks

EDIZIONE ITALIANA

CP/M®

CON MP/M™

di
Rodnay
Zaks

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

Annotazione

CP/M è un marchio registrato della Digital Research.
MP/M è un marchio della Digital Research.

NAD, QSORT sono marchi registrati della Structured System Group Inc.
WORDMASTER, WORDSTAR sono marchi della Micropro International Corporation.
MDS è un marchio registrato della Intel Corporation.
Z8000, Z80 sono marchi registrati della Zilog Inc.
TRS80 è un marchio registrato della Tandy Corporation.
APPLE è un marchio registrato della Apple Inc.
PET E CBM sono marchi registrati dalla Commodore Inc.

Copertina e grafica di Daniel Le Noury

Illustrazioni tecniche di J. Trujillo Smith

Ogni sforzo è stato compiuto per fornire informazioni complete e accurate. Comunque, la Sybex non assume responsabilità per il loro uso, né per violazioni di brevetti o di altri diritti di terzi che ne possono risultare.

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura dell'edizione italiana la signora Rosi Bozzolo, l'ing. Roberto Pancaldi, l'ing. Rivera.

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

RINGRAZIAMENTI

Vorrei ringraziare per il loro contributo tutte le persone che hanno fornito valida assistenza e suggerimenti, migliorando la completezza di questo libro. Tony Bove ha verificato diligentemente i programmi di esempio e ha scritto molte delle descrizioni iniziali dei comandi. David Haverty del Computer Center di Berkeley ha fatto validi commenti di miglioramento. Dorothy Kildall della Digital Research ha continuamente sostenuto questo lavoro fornendo informazioni sui nuovi sviluppi. Infine, tutti gli utenti interni alla Sybex hanno determinato la necessità di molte delle spiegazioni pratiche fornite nel libro. L'autore sarà grato per ogni ulteriore miglioramento suggerito dagli utenti del CP/M.

PREFAZIONE

Questo libro si propone di insegnarvi come usare il CP/M e le sue risorse. Non si presuppone alcuna conoscenza preventiva del calcolatore. Quello che si presuppone però è che abbiate accesso a un sistema di calcolo equipaggiato con il CP/M.

Il CP/M è diventato un sistema operativo standard per microcalcolatori. Molti degli utilizzatori di sistemi basati su microcalcolatori utilizzeranno prima o poi il CP/M. In base ai programmi applicativi che essi eseguono sul calcolatore useranno in parte o del tutto le risorse fornite dal CP/M. Per esempio un impiegato che introduce dati in un programma per la gestione dei crediti avrà solo bisogno di sapere come attivare il programma di cui ha bisogno e come correggere gli errori. D'altra parte un programmatore esperto potrebbe voler installare un nuovo programma permanente su un calcolatore o realizzare sofisticate funzioni di editing sui files. Questo libro è stato strutturato per soddisfare quest'ampia varietà di esistenza.

Il Capitolo 1 vi introduce al CP/M. Vi prende per mano e vi mostra come accendere il calcolatore e come realizzare le più comuni operazioni sui files, compresa la duplicazione dei dischetti. Dopo aver letto il Capitolo 1, saprete come operare sul vostro calcolatore equipaggiato con il CP/M e come realizzare le seguenti funzioni: creare un file, copiare un file, maneggiare dischetti, copiare dischetti così come usare parecchi importanti comandi operanti sui files. Questa conoscenza sarà sufficiente per permettervi di eseguire programmi applicativi noti senza errori. In effetti, probabilmente sarete sorpresi dal poco tempo che occorrerà per diventare abili a usare il calcolatore servendovi del CP/M.

Dopo aver imparato le basi del CP/M, vorrete probabilmente sapere di più. Il Capitolo 2 è un capitolo di consultazione sul CP/M, da leggere e poi da consultare quando occorrono informazioni specifiche. Presenta una descrizione globale ed esauriente di tutti i comandi del CP/M con l'eccezione di PIP, che è descritto nel Capitolo 3. Sebbene non sia necessario a molti utenti comprendere tutte le opzioni disponibili sul CP/M, una conoscenza generale perfezionerà la capacità di un utente del CP/M.

Una comprensione totale del programma PIP per il trasferimento dei files è indispensabile a un utente esperto del CP/M. Il Capitolo 3 descrive PIP molto dettagliatamente e mostra come collegare files, stampare files, multipli sulla stampante e usare le numerose altre possibilità fornite.

Il Capitolo 4 vi conduce attraverso una semplice sessione con il programma editor, 'ED'. ED è un potente programma di gestione dei testi che può essere usato per creare o manipolare convenientemente files di testo. Per quanto ED sia complesso, è relativamente facile da imparare.

Da questo punto del libro, avrete imparato tutte le possibilità del CP/M in dettaglio e potreste essere interessati ora a sapere come opera il CP/M. Il Capitolo 5 vi porta dentro il CP/M e spiega le sue operazioni interne. Questa conoscenza non è necessaria per usare il CP/M, ma occorre se avete intenzione di modificarlo.

Il Capitolo 6 usa un formato opportuno per riassumere tutti i comandi e i simboli usati dal CP/M (dettagliati nel Capitolo 2). Il Capitolo 6 sarà un manuale indispensabile per l'utente del CP/M.

Il Capitolo 7 presenta un'importante collezione di consigli pratici. Una volta che avete acquistato famigliarità con il CP/M e usate il calcolatore frequentemente, ci sono importanti modalità che dovrebbero essere seguite. Il Capitolo 7 fa delle raccomandazioni su come trattare o prevenire dei problemi pratici che possono sorgere usando il CP/M. Questo capitolo dovrebbe essere considerato una lettura essenziale per ciascuno.

Infine, il Capitolo 8 presenta un breve sguardo storico sul CP/M e sul suo futuro.

Molte tabelle utili di consultazione sono presentate nelle appendici. Dovrebbero essere consultate dopo che avete letto il libro. Queste tabelle contengono i comuni codici binari, i messaggi di errore, i simboli e i comandi forniti dal CP/M, da ED e da PIP.

Il CP/M è stato progettato per rendere semplice l'uso del microcalcolatore. Questo libro dovrebbe rendervi semplice l'uso del CP/M.

Viene analizzato il CP/M nelle sue varie versioni, comprendendo il CP/M 1.4 e il CP/M 2.2, e il nuovo sistema operativo multi-utente chiamato MP/M. È anche applicabile ai sistemi operativi compatibili con il CP/M, come il CDOS del Cromemco.

SOMMARIO

CAPITOLO 1

INTRODUZIONE AL CP/M E ALL'MP/M

INTRODUZIONE

Lo scopo di questo capitolo è di insegnarvi come compiere le operazioni fondamentali sul vostro sistema di elaborazione usando il CP/M. Non è richiesta nessuna conoscenza precedente dei calcolatori. Imparerete prima il vocabolario e le definizioni relative alle operazioni dei calcolatori. Imparerete poi come accendere il calcolatore, inserire il vostro *Dischetto di Sistema* e caricare il CP/M. Conoscerete i *files*; come crearli, dar loro dei nomi, fare copie di un file o di un dischetto completo. Imparerete ad usare la tastiera, lo schermo e la stampante per trattare, visualizzare e stampare il contenuto di un file. Alla fine di questo capitolo, avrete imparato ad usare tutti i più importanti comandi del CP/M.

DEFINIZIONI DI BASE

Un sistema di elaborazione tipico è mostrato in Fig. 1.1. Questo sistema comprende il calcolatore, le unità a dischi, la stampante e il terminale CRT. Per usare il calcolatore dovrete sedervi al terminale e digitare sulla tastiera. I messaggi compariranno allora sullo schermo del terminale CRT. Usando la stampante, sarete in grado di stampare il testo, se lo desiderate. I programmi che il calcolatore dovrà eseguire saranno immagazzinati su dischetti inseriti ín una unità.

In questo capitolo imparerete, passo passo, come compiere tutte le operazioni richieste per usare il sistema di elaborazione.

IL SISTEMA DI ELABORAZIONE

Un sistema di elaborazione consiste in componenti *hardware* e *software*. L'hardware si riferisce ai componenti fisici di un sistema (bulloni, viti, fili ecc.). Il software si riferisce ai programmi e ai files.

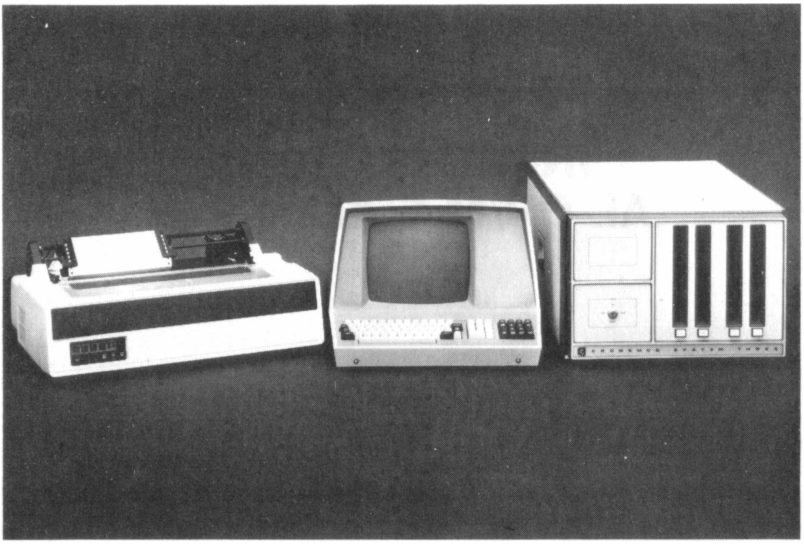


Figura 1.1: Esempio di computer

Gli elementi hardware

Gli elementi hardware di un tipico calcolatore di piccole dimensioni (il calcolatore, la tastiera e lo schermo CRT, la stampante e una o più unità a dischi) appaiono nella Fig. 1.1. Elementi addizionali dell'hardware, come una unità a nastri ed altri dispositivi (un microfono, un lettore di schede ecc.) possono pure essere aggiunti al sistema di calcolo.

Il calcolatore

Il calcolatore stesso è tipicamente chiuso in un contenitore. Poiché molte applicazioni del CP/M di solito richiedono una grande quantità di memoria (48K o 64K) e una unità a due dischi, molti fabbricanti riuniscono il calcolatore e l'unità a dischi nella stessa scatola. Questo è il caso della Fig. 1.1. Con il TRS80, l'Exidy e il più vecchio SOL, il microcalcolatore è incluso nella stessa scatola con la tastiera.

Il ruolo del calcolatore è di elaborare le informazioni. La sua attività è controllata dai *programmi* installati nella *memoria* del calcolatore. Lo scopo della memoria del calcolatore è di immagazzinare informazioni, sia programmi che dati. La sua capacità è misurata in parole (bytes di 8 bit per un microcalcolatore a 8 bit), in multipli di 1K, dove 1K=1024. Le capacità tipiche sono 16K, 32K, 48K e 64K.

Con l'odierna tecnologia, gran parte della memoria del calcolatore è volatile, e i suoi contenuti spariscono quando il calcolatore viene spento. In altre parole, ogni volta che un programma deve essere eseguito, deve essere caricato dal disco nella memoria del calcolatore. Questa operazione è compiuta dal CP/M automaticamente.

I dischi

Poiché la memoria del calcolatore (chiamata "RAM" da Random Access Memory, cioé memoria ad accesso casuale) è volatile, cioé non conserva l'informazione quando non è più alimentata, occorre un dispositivo di memorizzazione permanente per ogni calcolatore. Unità e dischi, sia flessibili che rigidi, sono usate a questo scopo sui piccoli calcolatori. Tutte le informazioni possono essere conservate su questo supporto, compresi i programmi, i files (collezioni di testi o dati) e una copia del programma CP/M stesso.

Il terminale CRT (schermo e tastiera)

Il terminale CRT consiste nella combinazione di uno schermo CRT (Cathod Ray Tube, cioé Tubo a Raggi Catodici, simile a quello degli apparecchi televisivi) e di una tastiera. È il mezzo con cui una persona può comunicare direttamente con il sistema di elaborazione. La tastiera è usata dall'utente per digitare caratteri che sono interpretati dal programma in esecuzione sul calcolatore. Lo schermo CRT visualizza le informazioni per l'utente. Sfortunatamente, come la memoria interna del calcolatore, il CRT è volatile, cioé l'informazione è visualizzata temporaneamente sullo schermo e poi scompare.

In molti sistemi commerciali si usa un classico terminale CRT che riunisce una tastiera e uno schermo CRT. Nei casi in cui la tastiera è già incorporata nella struttura del calcolatore, si aggiunge uno schermo video separato (o integrato).

La stampante

La stampante è un dispositivo *hard-copy*. È compito della stampante fornire una stampa permanente di tutte le informazioni richieste dall'utente. La stampante è usata per stampare programmi e documenti.

Ora che ci sono diventati *famigliari* i componenti hardware di un sistema, definiamo i componenti software.

I componenti software

Il termine "componenti software" si riferisce al programma (una

sequenza di istruzioni) e ai dati. Per essere più precisi, un *programma* è una sequenza di istruzioni che, una volta installata nella memoria del calcolatore, lo dirigeranno a compiere specifiche azioni. I *dati* sono collezioni di caratteri o numeri manipolati dai programmi. I programmi e i dati hanno il nome logico di *files* quando gli è stato assegnato un nome dall'utente. Più avanti in questo libro imparerete come usare una varietà di programmi e come creare e manipolare i comuni tipi di files.

Il CP/M stesso è uno speciale programma, o piuttosto una collezione di programmi forniti su un dischetto. I programmi usati in questo libro saranno memorizzati su dischetto.

Ci sono sostanzialmente due classi di software: software di sistema e software applicativo. Il *software di sistema* è il software abitualmente fornito con il sistema di elaborazione ed è richiesto per il suo funzionamento. Comprende il CP/M e alcuni programmi di utilità, quali PIP e ED, che saranno descritti in dettaglio più avanti.

Il *software applicativo* è l'insieme di programmi che un utente può usare per realizzare un lavoro specifico. Esempi di programmi applicativi sono un programma per gestire una lista di indirizzi, un programma per fare un inventario, un programma per tenere un libro mastro generale o un programma di trattamento dei testi.

La definizione di CP/M e MP/M

CP/M significa *Control Program for Microprocessor* (programma di controllo per microprocessori). MP/M significa *Multiprogramming Control Program for Microprocessor* (programma di controllo in multiprogrammazione per microprocessori). Sia il CP/M che l'MP/M sono entrambi *sistemi operativi*. Lo scopo del CP/M o di ogni sistema operativo è di eseguire i comandi dell'utente e permettergli di usare convenientemente tutte le risorse hardware fornite dal calcolatore. Per esempio, manderà testi alle stampanti, leggerà e tratterà informazioni dalla tastiera e visualizzerà informazioni sul CRT (schermo video). Inoltre, il sistema operativo CP/M eseguirà compiti interni, come gestire lo spazio dei dischetti e la memoria del calcolatore.

Una volta installato nella memoria del calcolatore, il CP/M diventa una parte integrante del sistema completo e spesso viene indicato con "il sistema". (Si noti che nel gergo dei calcolatori "il sistema può anche essere usato per descrivere l'insieme dei componenti hardware, cioé il calcolatore, la stampante, il CRT e le unità a dischi). In questo testo, quando si riferisce esclusivamente a programmi, "il sistema" si riferisce al CP/M - il sistema operativo.

Il funzionamento del sistema

Il funzionamento di un sistema completo diventa chiaro usandolo. La

funzione essenziale del sistema operativo CP/M è di permettere all'utente di usare convenientemente le risorse del sistema di elaborazione. Non appena il calcolatore è acceso, il sistema operativo è installato nella memoria del calcolatore e incomincia ad attendere comandi dalla tastiera. L'utente è allora in grado di entrare in dialogo con il CP/M e di attivare i programmi applicativi desiderati. Quando il programma applicativo termina, il CP/M riprende il controllo e attende il comando successivo. Il CP/M potrebbe essere visto come un servitore sempre presente, pronto ad obbedire ai comandi e a gestire le risorse del calcolatore, fintanto che l'utente non è nel mezzo dell'esecuzione di un programma applicativo. Specificamente, appena un programma applicativo (ad esempio un programma per gestire una lista di indirizzi) entra in esecuzione, prende il controllo della memoria del calcolatore e tutto il dialogo successivo è con quel programma. Però, quando il programma applicativo termina, il CP/M viene attivato di nuovo ed è pronto ad accettare nuovi comandi.

Riassumendo, il CP/M è una collezione di programmi che risiede su un dischetto chiamato dischetto di sistema. Il *monitor residente* o il *caricatore di bootstrap* (presente in ogni computer) di solito lo caricherà automaticamente dal dischetto non appena il sistema viene acceso. (Occasionalmente, è richiesto un intervento manuale da parte dell'utente).

Il CP/M fornisce comandi specifici per trasferire informazioni tra i dispositivi connessi al sistema di elaborazione, eseguire programmi e gestire files convenientemente. Come ogni buon sistema operativo, il CP/M fornisce molte caratteristiche addizionali. Le caratteristiche più importanti saranno descritte in questo capitolo e descrizioni comprensive di tutte le caratteristiche saranno fornite nei capitoli seguenti.

CP/M, MP/M e altre versioni

CP/M e MP/M

Sono state realizzate versioni successive del CP/M. Questo libro dapprima presenta le caratteristiche standard del CP/M fino alla versione 1.4, e poi indica i miglioramenti disponibili con le ultime versioni, come il CP/M versione 2.2 e l'MP/M versione 2.1. Molte altre versioni del CP/M sono anche state realizzate da altri produttori come "miglioramenti al CP/M". Per esempio, il CDOS del Cromemco è "compatibile con il CP/M" e fornisce possibilità aggiuntive. Tutte le caratteristiche del CP/M descritte in questo libro sono applicabili a queste versioni. Nel caso del CDOS del Cromemco, commenti specifici sono presentati nelle sezioni relative.

La differenza essenziale tra il CP/M e l'MP/M è che il CP/M è stato

progettato come sistema operativo a utente singolo. L'MP/M invece è un sistema operativo a più utenti che permette di usare contemporaneamente parecchi terminali in un sistema di elaborazione. L'MP/M fornisce tutte le possibilità del CP/M ed altre ancora. Le possibilità aggiuntive fornite dall'MP/M saranno descritte sistematicamente in ogni capitolo.

Il CDOS Cromemco

Si afferma che il CDOS Cromemco sia compatibile con la versione 1.3 del CP/M. In altre parole, i comandi della versione 1.3 del CP/M sono contenuti nel CDOS. Però il contrario non è vero: i programmi che si basano sulle possibilità del CDOS possono non funzionare sotto CP/M. Inoltre, il CDOS fornisce molte possibilità addizionali se confrontato con il CP/M. Il CDOS usa un file system che è identico al CP/M per cui ogni dischetto che può essere letto dal CP/M può anche essere letto dal CDOS. Ci sono differenze poco rilevanti: il prompt di sistema usato dal CDOS è il punto invece del segno >.

Inoltre, lo speciale programma CONPROC (Elaboratore di Console) deve essere presente su tutti i dischetti di sistema sotto forma di file. Nel CDOS viene fornita una versione differente del programma PIP con nome XFER. Funziona essenzialmente come PIP con pochi miglioramenti. Comunque PIP può essere eseguita anche sotto CDOS.

La più importante differenza pratica è che alcuni caratteri di controllo che non hanno significato per il CP/M sono interpretati dal CDOS e non possono essere usati dai programmi applicativi scritti per funzionare sotto CP/M. Tipicamente, il programma funziona ancora ma può non essere possibile usare alcuni dei caratteri di controllo.

Altri programmi

A rigor di termini, il sistema operativo CP/M comprende solo quei programmi richiesti per dialogare con il calcolatore e trattare il file system. Però la versione standard del CP/M viene fornita con molti programmi di utilità standard come PIP e ED (descritti in dettaglio nei capitoli successivi).

Naturalmente ogni utente del sistema di elaborazione eseguirà molti programmi applicativi specifici. Verranno svolti parecchi esempi specifici per dimostrare come questi programmi sono eseguiti sotto CP/M, e verranno presentate definizioni importanti. Poiché molti programmi applicativi richiedono un'organizzazione particolare del file system, è importante ricordare che i programmi applicativi che devono funzionare sul vostro sistema devono essere compatibili con il CP/M. Inoltre se sono scritti in un linguaggio specifico come il BASIC, richiederanno un

interprete del linguaggio come un interprete BASIC (trattato più avanti in questo stesso capitolo).

Abbiamo ormai imparato tutte le definizioni fondamentali. Accendiamo il nostro calcolatore e dialoghiamo con il CP/M.

PRIMO CONTATTO CON IL CP/M

Il calcolatore

Il modo migliore per superare la paura dei calcolatori è imparare come accenderli e come spegnerli senza danneggiare nulla. Se accendete il calcolatore in modo corretto, il sistema operativo prende il controllo della macchina e attende che voi digitiate un comando (cioé che gli comunichiate la vostra presenza e gli richiediate qualcosa). Se non riuscite a dire qualche cosa di coerente o date delle istruzioni sbagliate, il sistema operativo vi chiederà di ripetere la domanda.

Provate, se il vostro calcolatore è già acceso. Digitate adagio parole o lettere e guardate cosa succede. Se non succede nulla, premete il "return". Il sistema probabilmente ripeterà quello che voi avete digitato, seguito da un punto interrogativo. Attenderà poi di nuovo la vostra richiesta successiva. Non potete danneggiare il sistema operativo digi-

tando sul terminale. Tuttavia, continuando a provare, potreste cancellare dei files. Perciò aspettate e leggete avanti.

L'accensione del sistema

Per accendere il sistema e "chiamare" il CP/M, dovete maneggiare un dischetto (a meno che il vostro sistema sia basato su un disco rigido). Poche parole di precauzione sui dischetti sono quindi opportune.

I dischetti

Si usano due tipi comuni di dischi magnetici: i dischi flessibili (floppy) e i dischi rigidi (hard). I *dischi flessibili* sono indicati anche come *dischi* o *dischetti* e sono disponibili in due formati: 8 pollici e 5 e 1/4 pollici. Sono indicati nelle Fig. 1.2 e 1.3. I dischetti (dischi flessibili) possono essere usati per memorizzare una grande quantità di dati a basso costo. Pero i dischi flessibili sono relativamente lenti e malgrado la loro grande capacità, sono ancora troppo piccoli per memorizzare alcuni files, (ad esempio i grandi files commerciali). I *dischi rigidi* risolvono questo problema. Essi offrono una grande capacità ed un'elevata velocità di accesso, ma a un costo più alto. Molti piccoli sistemi di elaborazione sono equipaggiati con uno o entrambi di questi tipi di dischi. Poiché i dischi flessibili sono di gran lunga quelli usati più frequentemente, tutti i nostri esempi in questo libro faranno riferimento ad essi.

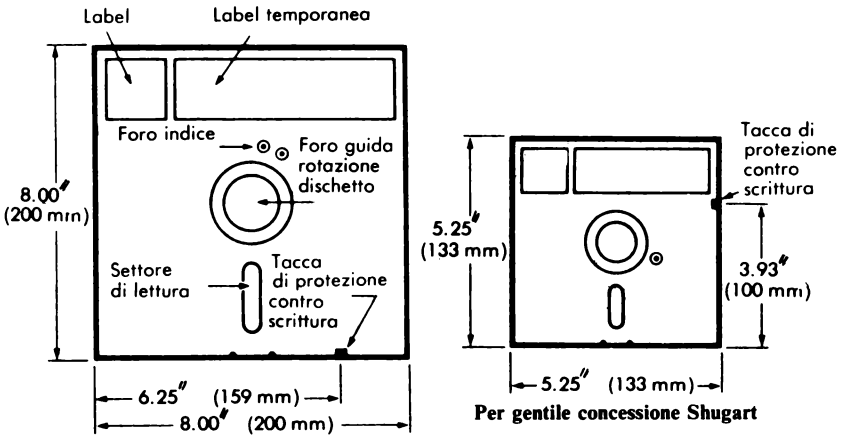


Figura 1.2: Dimensioni tra mini floppy e floppy

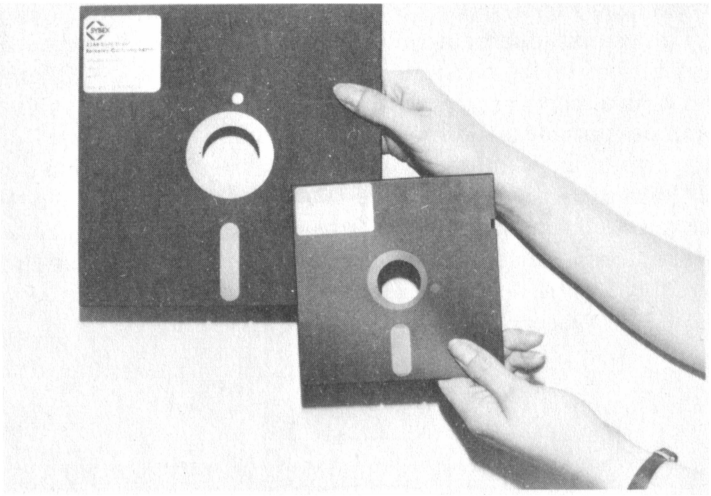


Figura 1.3: Confronto tra mini floppy e floppy

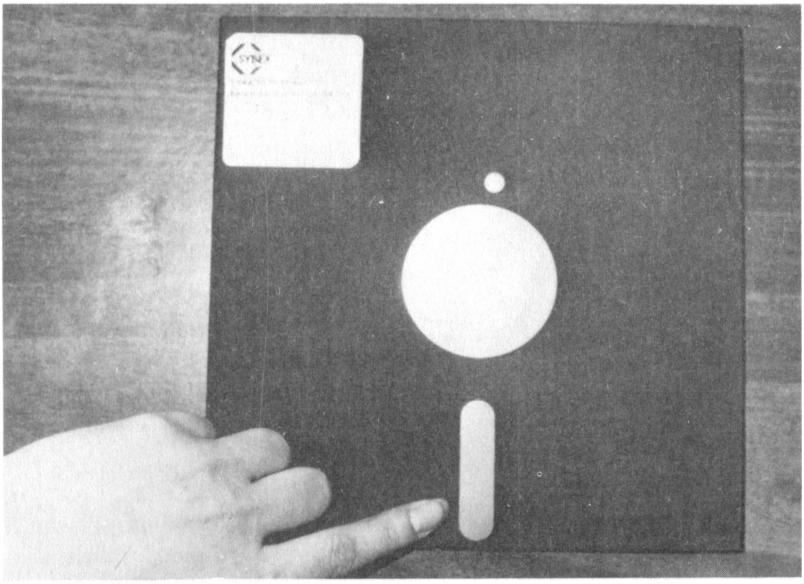


Figura 1.4: L'apertura consente il contatto con il disco alla testina di lettura/scrittura

I dischetti sono supporti magnetici fragili. Dovrebbero essere protetti dalle influenze magnetiche e trattati delicatamente. Un dischetto è mostrato nella Fig. 1.4.

La custodia quadrata (mostrata in Fig. 1.4) contiene un dischetto flessibile di mylar rivestito con un ossido magnetico. Quando viene usato, il dischetto ruota ad alta velocità dentro alla custodia. Il foro centrale consente al motore dell'unità a dischi di far ruotare il dischetto. L'apertura lunga (mostrata in figura) permette alla testina di lettura/scrittura di venire a contatto con la superficie del disco e di leggere o scrivere le informazioni su di esso, nello stesso modo in cui opera un registratore a nastro. Le informazioni sono registrate sul disco lungo cerchi concentrici chiamati *tracce*. Ogni traccia è divisa in *settori* logici dal CP/M (vedere Fig. 1.5).

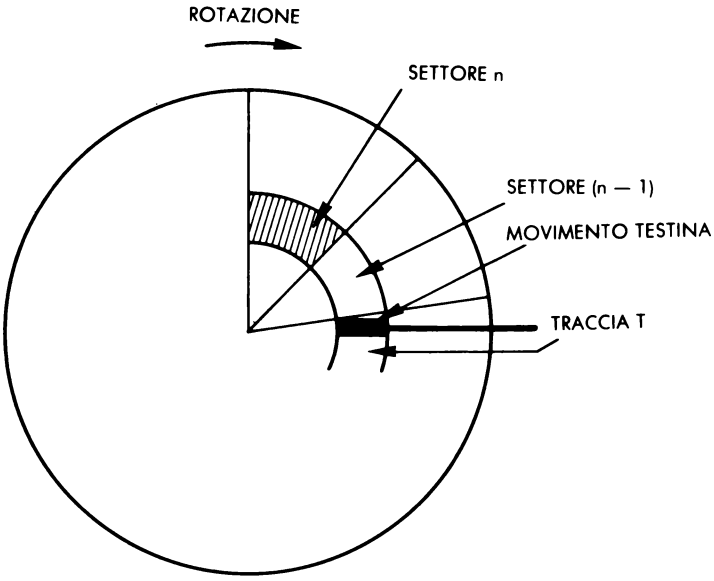


Figura 1.5: Tracce e settori

I dischetti hanno spesso - ma non sempre - una tacca di *protezione di scrittura*. Nei dischetti a 8 pollici, la tacca è ricoperta da un pezzo di carta rivestito di alluminio. Se si toglie questa etichetta, la tacca è scoperta e l'unità a dischi non è più in grado di scrivere sul disco.

Nei mini dischetti (5 pollici) è vero l'opposto. L'etichetta deve essere

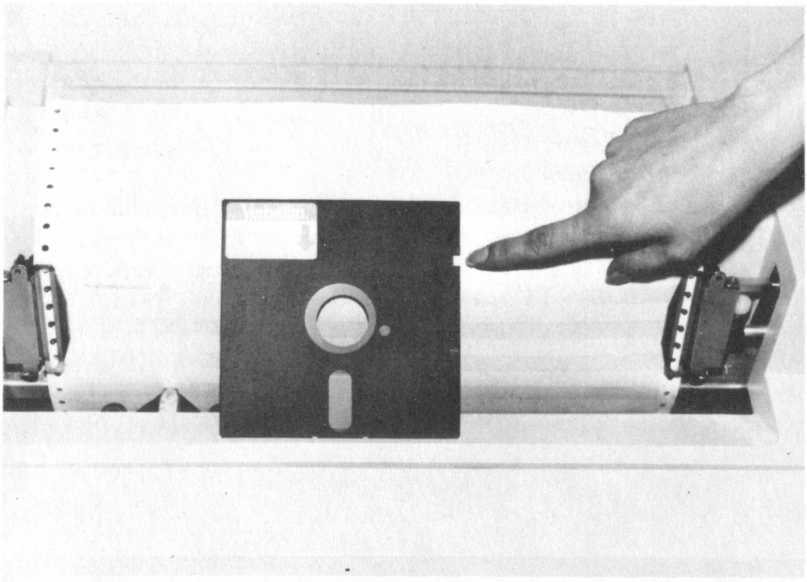


Figura 1.6: I dischetti da 8 pollici presentano una tacca per protezione scrittura

tolta per poter scrivere sul disco. Una volta che è posta sopra alla tacca, si può solo leggere. Questa caratteristica si usa per proteggere le informazioni importanti. Per esempio le copie originali dei dischetti, che sono conservati e archiviati, sono normalmente protetti contro la scrittura. Potete specificare questa opzione quando comprate i dischetti.

Il trattamento dei dischetti

Maneggiate sempre i dischetti con cura. Non toccate la zona scoperta. Non lasciate che vi si depositi la polvere e non graffiateli. Inoltre non mettete nessun oggetto magnetico vicino a un dischetto (ad esempio cacciaviti e telefoni) poiché questo può danneggiare il dischetto. È importante scoprire come inserire i dischetti nella vostra particolare unità a dischetti. Di solito si applica la "regola del pollice": dovreste tenere il dischetto col pollice sull'etichetta quadrata per inserirlo correttamente (vedi fig. 1.7). Se fate pratica per la prima volta, usate una *copia* del dischetto di sistema (nel caso doveste danneggiarlo).

Se spegnete il calcolatore (o l'unità se è separata dal calcolatore) mentre un dischetto è ancora dentro all'unità, il dischetto potrebbe diventare inutilizzabile. Impulsi di tensione possono provocare da parte

del calcolatore o dell'elettronica dell'unità l'invio di segnali non voluti al dischetto e la cancellazione delle informazioni esistenti. Se lasciate acceso il calcolatore o l'unità quando inserite o togliete i dischetti, non avrete questo problema (a meno che ci sia una caduta di tensione). Analogamente, quando volete spegnere il sistema, assicuratevi sempre di aver tolto prima i dischetti.

Procederemo adesso attraverso i passi che riguardano la "chiamata" del sistema. Ancora, con sistema intendiamo sia il CP/M versione 1.4, sia il CP/M versione 2.2, sia l'MP/M versione 1. Queste versioni sono simili, sebbene il CP/M versione 2.2 abbia alcuni vantaggi rispetto alla versione 1.4. L'MP/M versione 1 è quasi identico al CP/M versione 2.2. Tutti e tre i sistemi saranno descritti. Quando menzioniamo il "sistema", intendiamo uno qualsiasi dei tre; altrimenti specificheremo a quale sistema ci stiamo riferendo.

Figura 1.7: Inserimento del dischetto nel driver 1 (SOL)

Accendete, inserite il dischetto sistema e partite

Il procedimento

Prima di incominciare, tenete presente che il *Dischetto di Sistema* è un dischetto speciale che contiene il sistema operativo CP/M (o MP/M). Voi probabilmente avete ricevuto solo un Dischetto di Sistema, perciò dovreste chiederne o farne una copia da usare durante la fase di apprendimento.

Se non trovate qualcuno che vi faccia una copia del Dischetto di Sistema e dovete farla voi stessi; dapprima finite di leggere questa sezione e imparate come accendere il sistema, poi seguite i procedimenti descritti nel Capitolo 3 e riassunti qui sotto. Sullo schermo del video i caratteri sottolineati sono i caratteri digitati da voi. Un "ritorno carrello" (un tasto speciale sulla tastiera) è indicato con ↵. Sommariamente la procedura è:

1. Inserite il Dischetto di Sistema nella unità A.

```
A > SYSGEN ↵
SYSGEN VER 1.4
SOURCE DRIVE NAME # (OR RETURN TO SKIP) A
SOURCE ON A, THEN TYPE RETURN ↵
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION B, THEN TYPE RETURN ↵
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) ↵
A > PIP B:=A: *.* [V] ↵
(Messaggi di Copiatura)
A >
```

2. Inserite un dischetto vuoto nell'unità B.
3. Digitate i caratteri mostrati in questa rappresentazione del video:
4. Togliete la copia dall'unità B, mettetele un'etichetta e inseritela nell'unità A.

Ora, prendiamo una copia del nostro Dischetto di Sistema e impariamo come accendere e spegnere il microcalcolatore. Poiché calcolatori differenti hanno modi diversi per accendere il sistema, assicuratevi di seguire le istruzioni fornitevi col calcolatore.

Per caricare il CP/M dovete:

1. Accendere il calcolatore e le periferiche.
2. Trasferire il programma CP/M dal dischetto, dove è memorizzato, nella memoria del calcolatore.

La procedura esatta tende a variare leggermente per ogni calcolatore. I calcolatori che non sono stati progettati per il CP/M e che sono stati equipaggiati con il loro proprio monitor o sistema operativo (come il SOL) richiedono due operazioni successive per caricare il CP/M. I calcolatori che sono stati progettati per usare il CP/M, invece, eseguono il caricamento con una sola operazione. Il programma supervisore residente nel calcolatore carica automaticamente il CP/M dal disco.

Esamineremo ora un esempio di ogni caso descrivendo come accendere due differenti microcalcolatori.

L'accensione del Cromemco

Per accendere il calcolatore Cromemco, premete l'interruttore ON/OFF nella parte posteriore del sistema e ruotate la chiave anteriore nella posizione ON. Accendete il terminale, la stampante (se ne avete una), e ogni altro terminale (se usate l'MP/M). Le unità a dischi del Cromemco sono contenute nell'involucro del calcolatore e non hanno bisogno di essere accese separatamente. Se avete un altro dispositivo di memorizzazione, come una unità a disco rigido, accendete anch'essa.

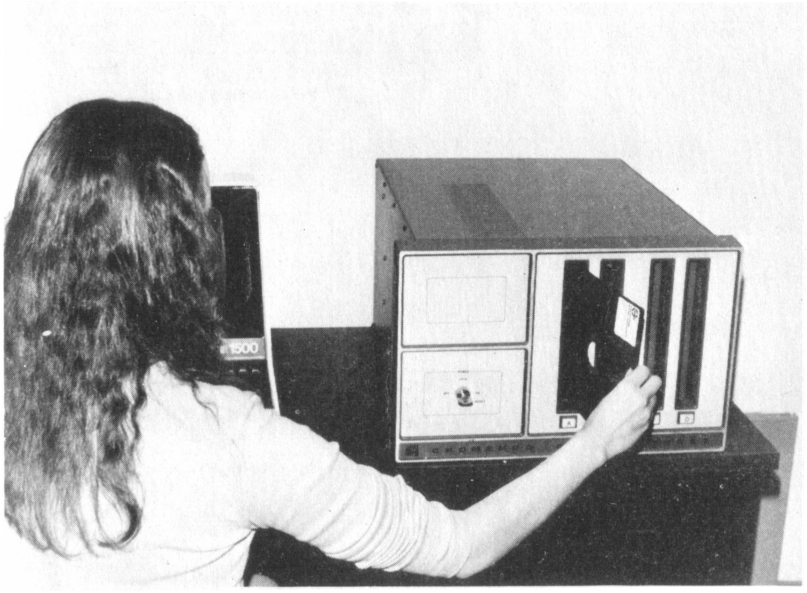


Figura 1.8: Inserimento di un dischetto

Ora inserite il Dischetto di Sistema nell'unità A - l'unità A è la più vicina alla chiave (come mostrato in Fig. 1.8). Ruotate la chiave nella posizione RESET e indietro nella posizione ON (vedi Fig. 1-9). Andate sulla tastiera del terminale, cercate il tasto Ritorno Carrello (normalmente indicato con RETURN o CR), e premetelo due o tre volte (Fig. 1.10). Improvvisamente apparirà sullo schermo il messaggio del sistema e un *prompt* (carattere che indica la presenza del sistema):

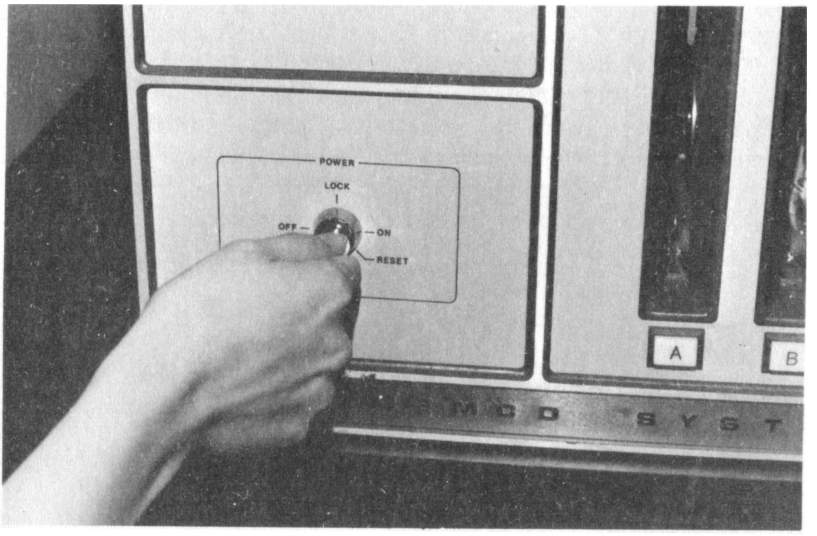
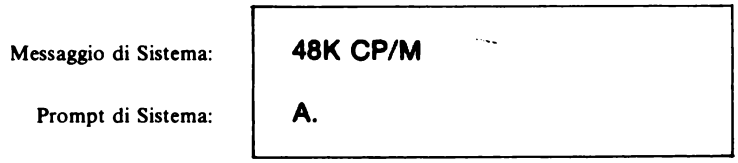
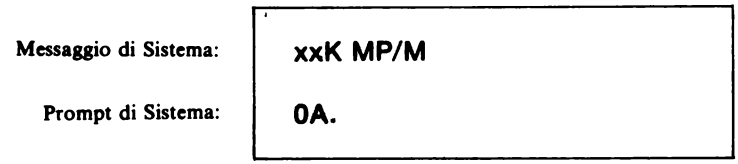


Figura 1.9: Accensione del Cromemco

Figura 1.10: La tastiera del terminale



o (con l'MP/M)



Il CP/M è ora caricato e funzionante e attende i vostri comandi.

L'accensione del SOL

Per accendere il calcolatore SOL (un sistema più vecchio) usate l'interruttore sulla parte posteriore del terminale, e accendete il monitor TV (schermo CRT). (Vedi Fig. 1.11). Accendete le unità a dischi separate e inserite il Dischetto di Sistema CP/M nell'unità A (quella più bassa). (Vedi Fig. 1.12). Apparirà immediatamente questo simbolo sul vostro shermo:

Questo è un prompt del programma supervisore del SOL, non del CP/M, che è ancora sul dischetto. Tenete presente che se il tasto LOCAL (sulla tastiera del SOL) è ON, non siete connessi al sistema. Mettete LOCAL nella posizione OFF premendolo.

Per caricare il sistema, digitate il seguente comando:

> EX E000 ↵

Il simbolo ↵ rappresenta il tasto RETURN. IL valore E000 è l'indirizzo del programma che carica automaticamente il CP/M dal disco.

Figura 1.11: Il sistema SOL

Figura 1.12: Inserimento di un dischetto in A

Questo valore varia con ogni unità a disco. Il venditore della vostra unità a disco vi dirà quale indirizzo deve essere usato con il suo sistema.

Dopo che avete digitato il comando per caricare il sistema, lo schermo conterrà:

```
48K CP/M
A >
```

o (con l'MP/M):

```
xxK MP/M
0A >
```

Il CP/M è pronto in attesa.

Che cosa fare se il SOL non funziona o non accade nulla

Verificate dapprima se il tasto LOCAL è ON (dovrebbe essere OFF). Se LOCAL è ON, ponetelo in OFF, e provate a digitare "EX E000" di nuovo seguito da RETURN (↵).

Se LOCAL era OFF e il sistema non viene caricato, verificate se il tasto UPPER è ON. Questo tasto rende tutti i caratteri maiuscoli invece che minuscoli. Dovete digitare "EX E000" tutto in caratteri maiuscoli (eccettuati gli zeri). Premete il tasto UPPER nella posizione ON (non il tasto SHIFT LOCK) e il vostro comando "EX E000" funzionerà.

L'USO DEL CP/M

Pronto a partire

Avete appena eseguito un'operazione di "bootstrap", o "partenza a freddo", o "cold boot". Alcuni preferiscono pensare che le macchine siano fredde fino a quando non le accendete, o che voi "carichiate" un sistema operativo con un calcio. Il termine "bootstrap" in effetti è venuto dall'idea che se voi foste abbastanza forti potreste "sollevarvi prendendovi per le cinghie degli stivali". In realtà, il supervisore residen-

te "tira fuori il CP/M dal dischetto e lo fa partire", cioé il sistema "fa partire se stesso".

Una "partenza a freddo" si differenzia da una "partenza a caldo", che sarà descritta più avanti.

Ora, che cosa significa questo "A" (o "0A"), e che cos'è un prompt?

I prompt di sistema

Un *prompt* è un messaggio o un simbolo che il sistema fornisce quando è pronto per il vostro comando successivo. Tutti i sistemi hanno dei prompt, ma ciascuno usa un simbolo differente. Per il CP/M versione 1.4 o precedenti, il simbolo di partenza è "A >". Per il CP/M versione 2.2 e l'MP/M, il simbolo è "0A >". La A indica l'unità a dischetti A, e lo "0" indica l'area utente 0. Le aree utente sono descritte nel Capitolo 2, ma per ora non avete bisogno di queste informazioni (non dovrete cambiare la vostra area utente).

Il prompt di sistema vi dice sempre "in" quale unità a dischetti (o a dischi) siete, cioé quella che state usando. Avete almeno una unità ed è indicata con "A". Le unità successive saranno indicate con "B", "C", ecc. Passiamo all'unità B, assumendo che abbiate due unità.

Voi digitate:

```
B: ↵
```

La risposta è:

```
B >
```

Il sistema ora funziona con l'unità B, e il prompt è ora:

```
B >
```

Torniamo indietro ad A:

```
B > A: ↵
A >
```

Files

I dischetti contengono le informazioni nei *files*. Per estrarre queste informazioni, dovete dire al calcolatore di andare in un particolare

dischetto o disco (per mezzo dell'unità a dischetti o a dischi) e cercare un file con un certo nome (un *filename*). Voi avete caricato il sistema usando l'unità a dischetti A. Poiché non siete "passati" a un'altra unità, siete ancora "in" A; pertanto avete il prompt "A >". Potete passare a un'altra unità *soltanto se*, avete un altro dischetto nell'unità.

Mostreremo come inserire un altro dischetto più avanti, sempre in questo capitolo.

Uno sguardo alla tastiera

Premete soltanto il tasto RETURN. Il prompt di sistema dovrebbe apparire di nuovo. Premete il tasto RETURN parecchie volte e osservate come è semplice mandare linee bianche al calcolatore. Il tasto RETURN è sempre usato per mandare un comando al calcolatore. Tutte le volte voi digitate un comando e lo fate seguire da RETURN rappresentato con ↵ in questo libro. Ci sono pochi casi speciali in cui non dovrete usare il tasto RETURN - per esempio, quando usate il tasto CTRL (control) e un altro tasto contemporaneamente. Questi casi speciali saranno spiegati in dettaglio più avanti.

Dapprima abituiamoci ad usare il sistema: digitate caratteri a caso e premete il tasto RETURN, come nell'esempio seguente:

```
A > QUALCOSA ↵
QUALCOSA?
A >
```

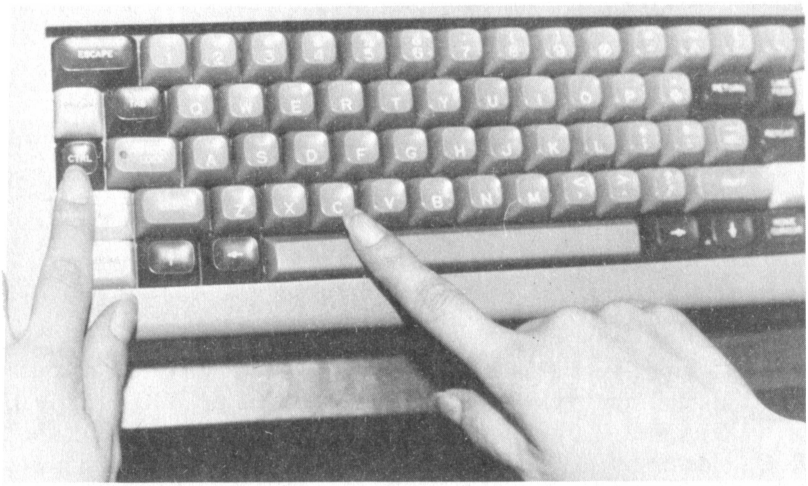


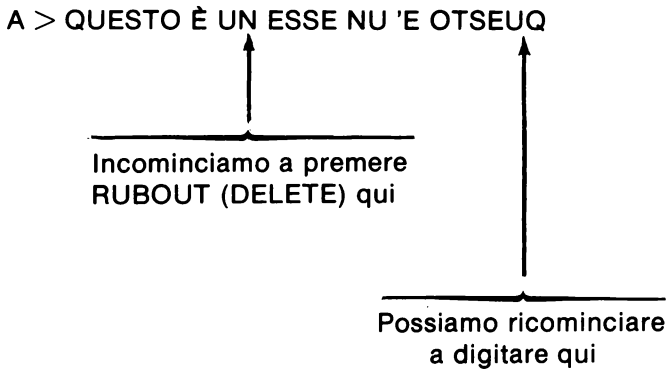
Figura 1.13: Il tasto di controllo C genera una "partenza a caldo"

Se qualche volta il sistema non risponde con un messaggio di errore e il prompt di sistema, come mostrato sopra, allora premete il tasto CTRL (tenendolo premuto) e premete contemporaneamente il tasto C (Fig. 1.13). Questa combinazione (CTRL AND C) produce una "partenza a caldo" (o "bootstrap a caldo", o " rilancio del sistema"). Una partenza a caldo essenzialmente interrompe qualunque cosa il computer stia facendo e rilancia il sistema operativo. Otterrete di nuovo il prompt di sistema.

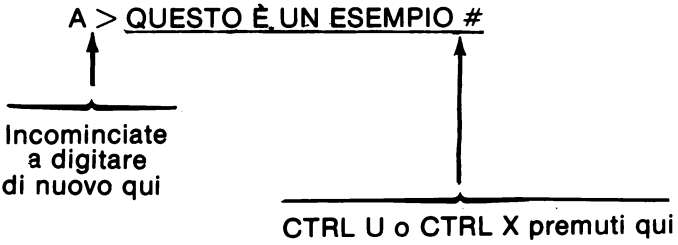
Se questo non succede, consultate la sezione "Che cosa c'é di errato..." in questo capitolo.

Dovreste abituarvi ad usare CTRL e C, (abbreviato ↑ C in questo libro; ↑ significa CTRL). Ricordate che CTRL deve essere tenuto premuto quando premete C. Dovreste anche abituarvi ad usare RUBOUT (DELETE o "←" in qualche terminale). Quando digitate qualcosa, potete cancellare l'ultimo carattere che avete digitato premendo RUBOUT (DELETE). RUBOUT (DELETE) cancellerà il primo carattere a sinistra. In qualche terminale potete tenere il tasto premuto e cancellare l'intera linea.

In molte versioni del CP/M RUBOUT (DELETE) *riscrive* il carattere che cancella invece di far scomparire il carattere. Se RUBOUT (DELETE) riscrive il carattere che cancella (ne fa l'eco), dovrebbe apparire qualcosa del genere:



Un altro modo di cancellare la linea è premere il tasto CTRL e il tasto U "contemporaneamente" (cioé, tenendo premuto CTRL mentre si preme U). CTRL e X faranno la stessa cosa. Sono abbreviati con ↑ U e ↑ X. Quando premete (CTRL U o CTRL X), il sistema scriverà il simbolo (#), con il significato di "qualsiasi cosa a sinistra di questo simbolo è cancellato":



Potete di nuovo digitare come se fosse una nuova linea. Potete, naturalmente, premere RETURN per mandare una linea bianca e riscrivere il prompt di sistema.

Che cosa c'é di errato

A volte accade qualcosa di insolito, e può accadere che non siate sicuri di ciò che lo ha causato. Se premete RETURN e non ottenete risposta dal calcolatore, il calcolatore è occupato a fare qualcosa (cioé a far funzionare un programma). Se inavvertitamente avete digitato il nome di un programma di cui non conoscevate l'esistenza, e quel programma ha incominciato a funzionare, potete abortire il programma (cioé fermarlo) e ritornare al sistema facendo una partenza a caldo ("bootstrap a caldo"): digitato ↑ C.

Se la partenza a caldo (↑ C) non restituisce il prompt di sistema (A >), verificate se qualche lampadina dell'unità a dischetti è accesa. Una lampadina accesa significa che il calcolatore sta cercando di leggere un dischetto in quell'unità. Se la lampadina è accesa e non c'é il dischetto, potete cercare di inserire un dischetto nell'unità in modo che il calcolatore abbia qualcosa da leggere. Se così facendo non ritorna il prompt (o non riparte il programma), allora dovrete ricominciare dall'inizio e fare una partenza a freddo (consultate l'inizio di questo capitolo nella sezione "Accendete, inserite il dischetto di sistema e partite!").

In qualche sistema c'é la possibilità di una interruzione per fermare il calcolatore. Nel caso del SOL, premendo UPPERCASE e REPEAT contemporaneamente ritornerete al monitor del SOL.

NOTA: Assicuratevi di togliere i dischetti prima di spegnere qualcosa.

Dischetto di sistema

È importante ricordare che il vostro dischetto di sistema è configurato per il vostro sistema specifico. Se qualche elemento dell'hardware del sistema è cambiato, il dischetto originale generalmente non funzionerà.

In particolare se cambiate la stampante, il terminale CRT, il controllo-

re dei dischi o lo spazio di memoria, avete bisogno di un nuovo dischetto di sistema. Se cambiate la configurazione hardware di volta in volta, assicuratevi di etichettare i vostri dischetti di sistema correttamente in modo da non confonderli.

L'esame della direttrice

Il nostro dischetto di sistema è nell'unità A. Contiene il CP/M insieme ad altri files. Esaminiamolo.

Potete scoprire quali files avete sul dischetto dell'unità A digitando il comando DIR ("direttrice"):

```
A > DIR ↵
A:PIP      COM
A:ASM      COM
A:LOAD     COM
A:PROGR    COM
A:STAT     COM
A:PROGR    INT
```

Se lo schermo scorre troppo velocemente, premete i tasti CTRL e S contemporaneamente (cioé ↑ S). In questo modo lo schermo si fermerà. Quando siete pronti a fare andare avanti la lista, premete di nuovo contemporaneamente i tasti CTRL e S per far ripartire lo schermo.

Questa è la rappresentazione del comando DIRettrice. Ogni dischetto (o disco) ha una "direttrice" di *nomi di files*, uno per ciascun file. Sappiamo che questi files sono sul Dischetto di Sistema perché siamo nell'unità A e il Dischetto di Sistema è nell'unità A. Ciascun nome di file è preceduto da 'A:' per indicare che il file è nell'unità A. Il primo 'PIP' e la successiva parola 'COM' formano un nome completo di file 'PIP COM'. 'PIP' è il nome primario e 'COM' è l'*estensione*, che indica il tipo di file. Nome ed estensione sono separati da un punto.

Le *estensioni* sono anche chiamate *tipi di files*. Per esempio tutti i files con estensione 'COM' sono *files di comando* (a volte chiamati *comandi transitori*). Tutti i files con estensione 'BAS' sono programmi sorgente BASIC e tutti i files con estensione 'INT' sono programmi intermedi BASIC. Non è necessario che usiate estensioni specifiche per un file di dati o un file di testo (un file che contiene certi tipi di informazioni come ad esempio un testo). Potete usare estensioni scelte da voi per suddividere i vostri files di dati in categorie.

È importante separare i differenti tipi di file, perché un programma potrebbe apparire nella direttrice con lo stesso nome e due o più tipi.
Per esempio:

TEXT.WRK (file di lavoro)

TEXT.BAK (copia di sicurezza del file o backup)

oppure

PROG.BAS (sorgente in BASIC)

PROG.INT (forma compilata)

Usate sempre il nome del file (cioé nome primario e estensione separati da un punto) quando vi riferite a un file, tranne quando usate un file di comandi come comando transitorio (discussi in questo capitolo più avanti). Poiché ora volete creare un nuovo file e non modificare files esistenti, dovreste imparare come inserire un nuovo dischetto e creare un nuovo file su di esso.

L'ESECUZIONE DI UN PROGRAMMA

Creeremo ora un file semplice. Questo file sarà usato nel resto del capitolo per dimostrare i procedimenti corretti e l'uso delle possibilità del CP/M. Il modo migliore per creare un file è eseguire un programma che crea un file. Un esempio di ciò è un programma di trattamento di testi o un programma commerciale. Se non ce n'é nessuno disponibile per mostrarvi come usare un tale programma, potete usare ED, l'*editor* che è fornito dal CP/M, per creare un file. Eseguiremo sia un esempio di programma commerciale sia un esempio dell'uso di ED per creare un file.

Dapprima inseriremo un dischetto bianco nell'unità B (vedi Fig. 1.14). Useremo il sistema di indirizzamento automatico NAD (della Structured System, Oakland, California), scritto in BASIC o organizzato con parecchi programmi. Per creare un nuovo file con nome e indirizzo, digiteremo:

A > CRUN NADENTRY ↵

Per funzionare, sia il CRUN (il compilatore CBASIC) e NADENTRY devono essere sul nostro Dischetto di Sistema. Si osservi che abbiamo

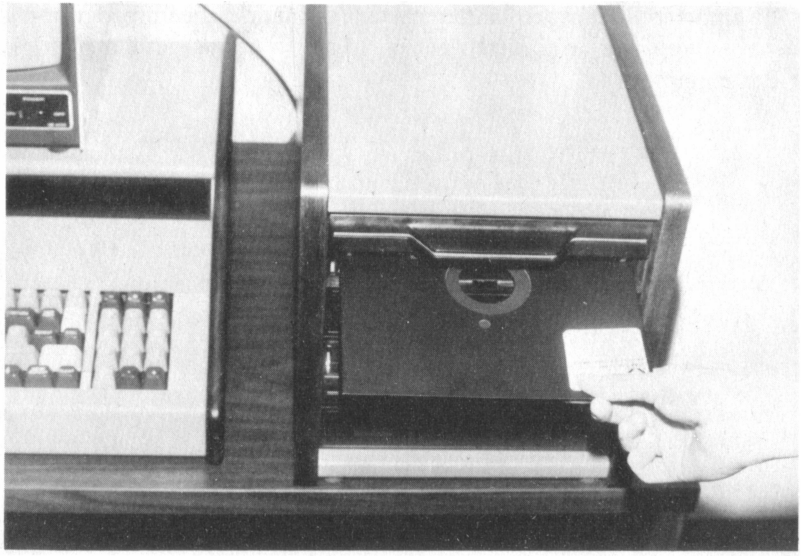


Figura 1.14: Inserimento di un dischetto in B

usato un nome di file incompleto per NADENTRY. Il programma CRUN automaticamente ipotizza che NADENTRY sia del tipo INT.

Per descrivere che cosa è accaduto, dobbiamo definire una nuova parola: *compilatore*. CRUN è un compilatore. Nell'istruzione CRUN NADENTRY, NADENTRY si riferisce a un programma scritto nel linguaggio per calcolatori BASIC. Per eseguire un programma scritto in BASIC, il calcolatore richiede un interprete e un compilatore BASIC. Qui viene usato il compilatore CRUN. La differenza teorica tra un compilatore e un interprete è che un compilatore esegue il programma in modo più efficiente, mentre un interprete permette lo sviluppo interattivo del programma. Una volta che il programma è in esecuzione, l'effetto è identico sia che si usi un interprete che un compilatore. La sequenza è mostrata di seguito (il dialogo è stato abbreviato per chiarezza).

```
A > CRUN NADENTRY ↵
CRUN VER 1.04
NAD VER 2.0
```

Creiamo un nuovo file NAMES sull'unità B (deve esserci un dischetto nell'unità B):

```
ENTER FILE NAME: NAMES
ENTER DISK DRIVE: B
```

Introduciamo un nome e un indirizzo (A significa "aggiungere" un nome):

```
ENTER FUNCTION (A, C, D, E, S, OR STOP): A ↵
RECORD NUMBER IS: 7
ENTER NAME: CHARLES FRIEND
ENTER LINE ONE OF ADDRESS: ABC COMPANY
ENTER LINE TWO OF ADDRESS: 123 LUNAR DR
ENTER CITY: PALO ALTO
ENTER STATE: CA
ENTER ZIP: 90010
ENTER PHONE: 408 123 4567
```

Salviamo tutto questo sul disco. Il comando 'S' è specifico del NADENTRY e significa "Salva":

```
ENTER FUNCTION (A, C, D, E, S, OR STOP): S ↵
1 RECORD SAVED
ENTER FUNCTION (A, C, D, E, S, OR STOP): STOP ↵
NADENTRY COMPLETED
A >
```

Siamo di nuovo nel CP/M. L'azione di salvare i dati introdotti ha creato un file su B chiamato NAMES. Questo file contiene il nome e l'indirizzo di CHARLES FRIEND.

LA CREAZIONE DI UN FILE CON ED

Un semplice file di testo può anche essere creato usando l'editor del CP/M, ED. Se avete un altro editor che è più semplice da usare e

funziona sul CP/M, e se la sua documentazione è semplice da leggere, provate usando il vostro editor; altrimenti, seguiteci nella nostra semplice lezione su ED.

Il comando ED è un *comando transitorio*, il che significa che esiste sul dischetto come file di comandi (come ED.COM). Il Dischetto di Sistema nell'unità A dovrebbe contenere ED.COM. Potete verificarlo usando il comando DIR:

Digitate:

A: ↵ (per tornare all'unità A)

Poi digitate:

DIR: ↵ (per esaminare la direttrice; dovreste vedere ED.COM)

Digitate:

B: ↵ (per tornare all'unità B)

Poiché adesso siete nell'unità B, dovete specificare l'unità A nel nome del file quando vi riferite a ED.COM.

Tenete presente che quando usate il CDOS Cromemco, non dovete specificare l'unità quando digitate un comando. Potete digitare: 'ED ↵' sull'unità B. Il CDOS cercherà il file sull'unità B poi automaticamente cercherà su A (se non lo trova in B). Questa è una caratteristica di comando.

Quando digitate un comando transitorio, non dovete digitare l'estensione '.COM' con il nome primario. Per eseguire 'ED.COM', dovete digitare soltanto 'ED' o 'A:ED' in dipendenza dal disco sul quale siete. (In effetti, se digitate 'ED.COM', avrete un errore!)

Dapprima pensate a un nome per il vostro file, come SAMPLE.TXT (separate sempre il nome primario SAMPLE dall'estensione TXT con un punto). L'estensione TXT non è richiesta, ma vi aiuta a identificare il file.

Eseguite il comando ED digitando: 'ED', uno spazio o poi il nome del vostro nuovo file:

```
B > ED SAMPLE.TXT ↵
ED?
B >
```

Aspettate! Abbiamo dimenticato di specificare l'unità A con il comando ED (perché ED.COM è sull'unità A e noi siamo sull'unità B).

Proviamo di nuovo:

```
B > A:ED SAMPLE.TXT ↵
NEW FILE
*
```

Funziona. Non dimenticate di digitare SAMPLE.TXT dopo ED!

L'asterisco (*) è il *prompt dell'editor.* Indica che il programma ED è attivo e funzionante e pronto per il vostro succesivo comando ED. I comandi ED sono descritti in dettaglio nel Capitolo 4, ma potete imparare alcuni qui: il comando I inserisce il nuovo testo nel file, il comando B vi porta all'inizio del file, e il comando T visualizza il contenuto del file ('Testo'). Il comando E *salva* il file e *termina* il lavoro con ED ('End').

Potete usare ED per creare e modificare file di testo. Un file di testo è del tutto simile ad ogni altro file e le informazioni vi sono rappresentate in un codice binario chiamato ASCII. Ogni carattere sui tasti della tastiera del terminale ha un numero di codice ASCII - questi numeri sono mostrati nell'Appendice C. Probabilmente non avrete mai bisogno di usare il numero di codice, ma questo vi aiuta a sapere come sono "codificati" - nel caso vi dovesse accadere di vedere strani numeri nel vostro file accompagnati da strani simboli.

Per inserire nuovi caratteri nel vostro nuovo file, usare il comando I poi digitate le linee di testo. Terminate ciascuna linea con un ritorno carrello (RETURN o CR), proprio come su una macchina da scrivere. Usate ↑ *Z* quando avete finito di inserire il testo.

```
*I ↵
QUESTO È IL MIO NUOVO FILE DI TESTO,
CHIAMATO SAMPLE.TXT ↵
↑Z
*
```

Adesso aggiungete una seconda linea:

```
'QUESTA È LA LINEA DUE.'
```

Se volete visualizzare di nuovo quello che avete digitato, usate questa sequenza di comandi:

```
* B ↵ (va all'inizio del file)
* #T ↵ (visualizza il contenuto)
```

```
QUESTO È IL MIO NUOVO FILE DI TESTO,
CHIAMATO SAMPLE.TXT.
QUESTA È LA LINEA DUE.
*
```

Il comando B riporta all'inizio del file di testo, e il comando T stampa o visualizza una linea di testo oppure l'intero file (se digitate un '#' prima del T).

Dovreste ora salvare il file usando il comando E:

```
*E↵
B>
```

Il comando E salva il vostro file e termina il programma ED. Ritornate al sistema e il vostro file è sul dischetto nell'unità B. SE NON ESEGUITE IL COMANDO E, PERDERETE LE INFORMAZIONI CHE AVETE APPENA INTRODOTTO NEL FILE. Usate il comando E frequentemente! Questo perché il testo che avete digitato è memorizzato nella memoria interna del calcolatore, dove può essere facilmente raggiunto e modificato. Quando digitate E, viene memorizzato su un supporto più permanente, il disco.

Se uscite da ED, spegnendo il sistema o riinizializzandolo (premendo ↑ C), perdete i contenuti della memoria interna del calcolatore. Solo quello che sarà stato esplicitamente salvato sul disco rimarrà accessibile. Questo è valido per ogni programma di editing. Per questo abbiamo usato il simbolo 'S' alla fine di NADENTRY nella sezione precedente.

Come raccomandazione generale, salvate frequentemente, specialmente se avete la tendenza a premere ↑ C accidentalmente, o se lasciate il terminale incustodito. Potete salvare il file tutte le volte che volete. In questo modo aggiornerete il vostro file sul disco, senza moltiplicarne le copie.

LA GESTIONE DEI FILES

La visualizzazione del files

Per visualizzare un file quando siete tornati al sistema operativo, usate il comando TYPE e specificate il nome del file:

```
B > TYPE SAMPLE.TXT ↵
_____
_____   } visualizzazione
_____
```

Dovreste impratichirvi a creare files usando ED, e i suoi comandi. Troverete una descrizione completa di ED nel Capitolo 4. Tenete presente comunque che sono disponibili altri programmi di editing (e di trattamento di testi word processor) in grado di funzionare sotto CP/M o MP/M, e che questi programmi hanno comandi differenti e strutture di file differenti. Usate l'editor che si adatta di più alle vostre esigenze. Alcuni sono molto semplici, mentre altri sono più complessi e hanno un maggior numero di possibilità.

Se avete più di un programma di editing, ricordate quale editor avete usato per creare un file, e usate quell'editor per modificarlo.

IL CAMBIAMENTO DI NOME DEI FILES (REN)

Il comando REN è usato per cambiare il nome di un file. Viene scritto come:

```
REN nuovonome = vecchionome
```

Usate sempre i nomi completi. Per esempio:

```
REN FILE2.TXT = OLD.TXT
```

oppure:

```
REN BETTER.NAD = NAMES.NAD
```

Dopo l'esecuzione di REN, 'OLD' non esisterà più. Sarà stato sostituito da 'FILE2'. Analogamente 'BETTER' sarà stato sostituito da 'NAMES'.

Ricordate di usare sempre nomi di files completi, compresi il punto e l'estensione.

IL CARICAMENTO DI UN NUOVO DISCHETTO (RILANCIANDO IL SISTEMA)

Vorremmo subito fare copie di dischetti e sostituire i dischetti nell'unità B. Per fare questo, dapprima dobbiamo imparare come eseguire una "partenza a caldo".

Quando viene inserito un dischetto, il CP/M legge automaticamente la direttrice nella memoria del calcolatore e "registra" il dischetto. Se cambiate il dischetto in B e poi immediatamente cercate di scrivere sul

nuovo dischetto in B (ad esempio per creare un file) non riuscirete a farlo. Questo procedimento non funzionerà a causa di una caratteristica (delle prime versioni del CP/M) che ha lo scopo di proteggere i dischetti da cancellazioni involontarie.

Pertanto, dopo aver sostituito un dischetto, dovete autorizzare specificamente il CP/M a scrivere su di esso. Questo viene fatto rilanciando il CP/M e viene chiamato *partenza a caldo.*

Vediamo come funziona. Tenete il Dischetto di Sistema nell'unità A e inserite un nuovo dischetto nell'unità B. (Assumiamo per esempio che abbiate precedentemente usato l'unità B per fare una copia). Adesso tornate al termine e prendete ↑ C per "rilanciare" il sistema (fare una "partenza a caldo").

Una partenza a caldo (↑ C) predispone il nuovo calcolatore al dischetto. Il calcolatore conserva le informazioni sul dischetto costruendone una "mappa" nella sua memoria. Quando togliete un dischetto precedente e ne inserite uno nuovo, la mappa del dischetto precedente è ancora nel calcolatore. A causa della caratteristica (menzionata precedentemente) del CP/M che non vi permette di scrivere su un dischetto se ce n'era un altro prima nella stessa unità (per evitare errori accidentali), dovete digitare ↑ C per autorizzare il CP/M a scrivere sul nuovo dischetto. Si realizza una partenza a caldo (↑ C) per *cambiare la mappa* in modo che il calcolatore possa scrivere informazioni sul nuovo dischetto.

Se state soltanto *leggendo* da un nuovo dischetto (cioé guardando i files e il loro contenuto) non è necessario fare una partenza a caldo per introdurre un nuovo dischetto. Se state scrivendo (cioé creando un file, creando una copia, o mandando dati a un file) su un nuovo dischetto in un'unità che aveva contenuto prima un altro dischetto, *dovete* rilanciare il sistema per creare una nuova mappa per il nuovo dischetto. Una partenza a caldo è richiesta ogni volta che un nuovo dischetto viene inserito con l'intenzione di scrivervi sopra a meno che le nostre istruzioni specifichino diversamente (soltanto in operazioni speciali di copiatura).

Si dovrebbe tenere presente che l'MP/M non permette di cambiare i dischetti e di inserire il dischetto nuovo a meno che non si faccia un *reset del disco* (descritto più avanti). Inoltre, con il CDOS Cromemco, la caratteristica che richiede una partenza a caldo è stata eliminata, e non dovete più rilanciare il sistema ogni volta che cambiate i dischetti in un'unità.

Dopo aver rilanciato il sistema (↑ C), dovreste ottenere il prompt di sistema (A >). A questo punto siete liberi di accedere ai files in scrittura sia nell'unità A che nell'unità B o in altre unità se ne avete.

Potete passare dall'unità A all'unità B digitando 'B:' come comando:

```
A > B: ↵
B >
```

Ora siete nell'unità B. Usate il comando DIR per determinare quali files avete nel dischetto dell'unità B:

```
B > DIR ↵
NOT FOUND
B >
```

Il messaggio "NOT FOUND" significa che DIR non ha trovato alcun file sul dischetto. Questo perché abbiamo un dischetto bianco in B. Quando ci si riferisce a un file, il CP/M assume (a meno che non gli si specifichi diversamente) che il file sia nell'unità corrente. A questo punto, per riferirvi a un file nell'unità A, potete o ritornare all'unità A o specificare l'unità 'A' nel nome del file. Ad esempio se volete far riferimento a PIP.COM nell'unità A, dovreste dire;

```
A:PIP.COM
```

Il comando DIR può essere usato per trovare un file specifico. Per far ciò dovete digitare il nome del file dopo il comando DIR (separato da uno spazio):

```
B > DIR A:PIP.COM ↵
A:PIP      COM
B >
```

Un altro modo di trovar PIP.COM sull'unità A è di tornare all'unità A e poi eseguire il comando DIR:

```
B > A: ↵
A > DIR PIP.COM ↵
A:PIP      COM
A >
```

Se passate a un'unità a dischetti che non contiene un dischetto, il sistema si "sospenderà" (la luce sull'unità del disco si accende, poi non accade nulla e la tastiera non funziona più) fino a quando non fate una partenza a freddo (si veda "Accendete, inserite il dischetto di sistema, e partite" all'inizio di questo capitolo).

Se avete soltanto un'unità a dischi, non avete scelta - dovete togliere il Dischetto di sistema per inserire un nuovo dischetto. Poiché questo è più complicato, guardate le istruzioni al Capitolo 2. Con il CP/M, non potete fare copie di files se non avete più di una unità. Comunque, programmi speciali sul vostro calcolatore possono permettervi di copiare un intero dischetto.

La copia di un file

Vi accadrà sempre di voler fare copie di files. In effetti, appena avete creato o ottenuto un nuovo file o programma, dovreste farne una copia da conservare separatamente (nel caso accada qualche infortunio al dischetto originale).
Il procedimento corretto è il seguente:

- Quando ottenete un nuovo programma o file, fatene una copia prima di fare qualsiasi altra cosa. Mettete in salvo il dischetto originale e lavorate soltanto sulla copia.
- Quando create o moltiplicate un file, fate una copia del file prima di lasciare il sistema. Etichettate in modo chiaro. Indicate la data. Mettetelo in salvo.

Per eseguire le operazioni di copiatura si usa il comando PIP. PIP significa "Peripheral Interchange Program". (Programma di scambio tra periferiche). È un programma (scritto in linguaggio macchina) che eseguite digitando 'PIP'. Per eseguirlo, PIP deve essere un file già esistente sul vostro dischetto con il nome PIP.COM (solitamente residente sul Dischetto di Sistema).

Se avete seguito gli esempi, allora avete un file di testo SAMPLE.TXT nell'unità B. Se è un file prezioso, dovreste copiarlo su un nuovo dischetto bianco. Questo sarà spiegato più avanti.

Ora cerchiamo di usare PIP. Dapprima dovete passare all'unità A, perché PIP.COM è nell'unità A:

```
B > A: ↵
A >
```

Ora potete eseguire PIP:

```
A > PIP ↵
*
```

L'asterisco (*) è il prompt del programma PIP, e vi dice che PIP è caricato e funzionante. Potete ora digitare un'espressione PIP che realiz-

za l'operazione di copiatura. Espressioni PIP semplici prendono questa forma:

u:nomecopia = u:nomeoriginale

Gli argomenti 'nomecopia' e 'nomeoriginale' sono effettivi nomi di files e "u" è per la lettera dell'unità. PIP copia sempre da un file originale a un file copia. Pertanto il file originale deve già esistere; PIP crea il file copia con il nome fornito da voi. Per esempio:

```
* A:COPY.TXT = B:SAMPLE.TXT↵
```

Questa espressione dice a PIP di fare una copia di SAMPLE.TXT, che è sull'unità B, di chiamare questa nuova copia COPY.TXT e di metterla sull'unità A.

Poiché siete già nell'unità A, potete abbreviare l'espressione precedente:

```
* COPY.TXT = B:SAMPLE.TXT ↵
```

PIP assume che il file di copia debba essere creato sull'unità *corrente*, (cioé quella in cui siete in questo momento). Poiché SAMPLE.TXT è nell'unità B, dovete specificare la lettera dell'unità. A questo punto, usate sempre le lettere dell'unità quando fate pratica nell'uso di PIP, in modo da familiarizzarvi con la copiatura da unità a unità.

Quando PIP ha finito di copiare SAMPLE.TXT nel nuovo COPY.TXT, risponde con l'asterisco (*), il prompt di PIP. Premete il tasto Ritorno Carrello (RETURN o CR) per uscire dal programma PIP e tornare al sistema:

```
* ↵
A >
```

Indipendentemente dall'unità sulla quale e dalla quale copiate, all'uscita da PIP siete ancora nell'unità in cui eravate quando avete eseguito il comando. Poiché abbiamo eseguito PIP dall'unità A, siamo ancora in A.

Se volete eseguire soltanto un'operazione di PIP, potete usare un comando più breve e digitare l'espressione PIP e il comando PIP in una linea.

```
A > PIP A:COPY.TXT = B:SAMPLE.TXT ↵
A >
```

Quando PIP termina la copiatura, appare il prompt di sistema.

Notate coma le luce sull'unità B si accende e poi si spegne, e la luce sull'unità A si accende e poi si spegne quando si accede a ciascun disco. PIP copia il file in segmenti chiamati *blocchi*. Se il file è grande, deve tornare al file originale per leggere altri blocchi.

Ora avete una copia SAMPLE.TXT chiamata COPY.TXT nell'unità A. Facciamo una copia di COPY.TXT e mettiamola nell'unità B:

```
A > PIP B: = A: COPY.TXT ↵
A >
```

Non abbiamo dovuto specificare il nome della copia ('B:') perché vogliamo che la copia di COPY.TXT abbia lo stesso nome dell'originale. Probabilmente vorrete fare operazioni di copiatura di questo tipo più spesso di ogni altra. Il comando indicato sopra è equivalente a:

```
A > PIP B:COPY.TXT = A:COPY.TXT ↵
A >
```

Entrambi i comandi creano un nuovo COPY.TXT sull'unità B che è la copia di COPY.TXT esistente sull'unità A. Non dovete specificare un nome di file per la copia se volete che abbia lo stesso nome dell'originale. Dovete invece specificare l'unità e l'unità deve essere differente dall'unità del file originale se il nome del nuovo file è identico a quello originale.

Questo perché non potete avere due files su un dischetto con lo stesso nome. Se cercate di fare la copia di un file senza specificare o un nuovo nome di file o un'unità differente, otterrete la frase 'INVALID FORMAT', seguito dalla parte dell'espressione che ha causato l'errore. Se ottenete questo errore, premete il tasto RUBOUT (o DELETE) per cancellare l'errore.

La maggior parte delle operazioni di copiatura richiedono trasferimenti da dischetto a dischetto (o da disco a dischetto e viceversa), perché volete fare copie di salvataggio dei files sui dischi o sui dischetti. Poiché siete sul punto di imparare il comando ERA (cancellazione), dovreste imparare come fare la copia dell'intero dischetto (nel caso facciate un errore con il comando ERA).

LA COPIATURA DI UN INTERO DISCHETTO

Avete appena imparato come copiare il file COPY.TXT sull'unità A nel file COPY.TXT sull'unità B:

```
A > PIP B: = A:COPY.TXT ↵
A >
```

Avete appena fatto una copia di *salvataggio* di COPY.TXT.

Per copiare un intero dischetto con un solo comando dovete usare un *nome di file generico* (un espressione che dica al calcolatore di "eseguire quest'istruzione su ogni file il cui nome sia sostituibile a quello generico"). Il nome di file generico da usare per tutti i files di un dischetto '*.*'. Per esempio:

```
A > PIP B: = A: *.* ↵
```

Il simbolo '*' indica un nome qualsiasi nel suo campo. Questo comando copia tutti i files sostituibili a '*.*' (tutti i files sul dischetto) dell'unità A in nuovi files con lo stesso nome sull'unità B. (Tenete presente che con l'MP/M e le nuove versioni del CP/M, '*.*' corrisponderanno tutti i files nell'area corrente dell'utente).

Quando si copia un intero dischetto sull'unità B, è meglio per ora avere in B un dischetto vuoto. Altrimenti potrebbero verificarsi due problemi:

1. Se B contiene già dei files, deve avere abbastanza spazio disponibile per sistemare tutti quelli di A. Un dischetto standard può memorizzare fino a 270K bytes, compresa la direttrice. Se eseguite DIR, vi dirà quanto spazio sul dischetto è già stato usato).
2. Se B ha già un file con lo stesso nome di uno che deve essere copiato in B, l'operazione di copiatura si fermerà.

Poiché siete nell'unità A (nell'esempio precedente), potete abbreviare il comando precedente:

```
A > PIP B: = *.* ↵
```

Entrambi i comandi cercano nell'unità A (l'unità corrente in questo esempio) i files che corrispondono a '*.*', e creano copie dei files sull'unità B usando gli stessi nomi. Se esiste già un file sull'unità B con lo stesso nome, viene cancellato.

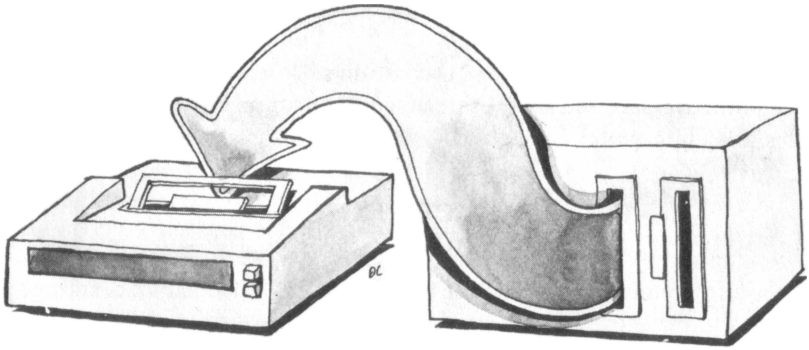
Ricordate che i nomi dei files possono avere dieci caratteri alla sinistra

del punto e tre caratteri alla destra. Il simbolo '*' corrisponde a qualsiasi sequenza di dieci caratteri alla sinistra del punto, e il simbolo '.*' corrisponde a qualsiasi sequenza di tre caratteri a destra del punto.

Quando usate il comando precedente, ricordate che tutti i files (soltanto i files) sul dischetto saranno copiati. Il programma CP/M non è un file. È memorizzato su due tracce riservate sul disco. Se il CP/M è l'unica cosa sul dischetto, il comando DIR mostrerà un 'Dischetto Vuoto'. Questo perché il CP/M è un programma speciale che non viene memorizzato come file. PIP copia soltanto files. Se deve essere copiato anche il CP/M, si deve usare un comando speciale, cioé SYGEN (descritto più avanti).

Un consiglio pratico

Ogni volta che modificate un file con l'editor, è preferibile usare la copia la volta successiva che usate il file: la copia sarà caricata molto più velocemente dal programma editor che il file originale. Questo è dovuto alla tecnica usata di disporre i dati memorizzati su disco in blocchi. Sul dischetto originale il file è disperso in blocchi non adiacenti. Sulla copia è compattato in blocchi adiacenti e quindi i dati sono reperiti molto più rapidamente.



LA STAMPANTE DI UN FILE

Dovreste avere due copie di COPY.TXT (SAMPLE.TXT e COPY.TXT sull'unità B), ma prima di giocare con il comando ERA, dovreste anche imparare come mandare un file alla stampante. Ci sono due modi semplici.

Il modo più semplice è di premere il tasto CTRL e il tasto P contemporaneamente. Ora digitate una "interlinea". Osservate che la vostra stampante (se è accesa) salterà una linea. Ora, qualsiasi cosa digitiate alla tastiera e qualsiasi cosa appaia sullo schermo sarà anche stampato sulla carta. Provate digitando e usando il comando DIR per verificare il funzionamento corretto della stampante.

Poi usate il comando TYPE:

```
A > TYPE SAMPLE.TXT ↵

_____
_____
_____   } video
_____

A >
```

Sullo schermo e sulla stampante, il comando TYPE ha stampato il vostro file. Ora premete ancora contemporaneamente CTRL e P. Così avete disabilitato l'operazione di stampa. Usate di nuovo il comando

TYPE e il file appare soltanto sullo schermo del terminale (ad una velocità molto maggiore).

Questo metodo è semplice ma scomodo se volete stampare molti files. Per stampare parecchi files, usate un comando di PIP che invia uno o più files alla stampante:

```
A > PIP LST: = COPY.TXT ↵
```

Questo comando manda il file COPY.TXT al "dispositivo di stampa" che è la stampante. 'LST:' è sempre usato per "dispositivo di stampa". Se non funziona, controllate il comando STAT nel Capitolo 2.

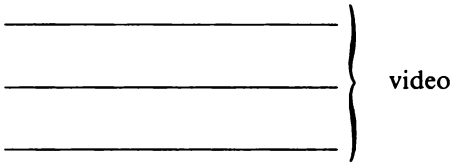
Potete mandare tutti i files dell'unità A alla stampante sostituendo '*.*' a 'COPY.TXT'. Per stampare tutti i files dell'unità B dall'unità A, usate il seguente comando:

```
A > PIP LST: = B *.* ↵
```

LA CANCELLAZIONE DEI FILES

Ora che avete COPY.TXT sull'unità A, e COPY.TXT e SAMPLE.TXT sull'unità B, potete permettervi di cancellarne uno. Verificate sempre prima la direttrice per vedere quello che avete:

```
A > DIR ↵
```




```
A > B: ↵
B > DIR ↵
SAMPLE   TXT
COPY     TXT
B >
```

Usate questo comando per cancellare il file COPY.TXT:

```
B > ERA COPY.TXT ↵
```

Se il file non esiste, otterrete il messaggio di errore "File not Found"

"File non trovato"). Se volete cancellare un file su un altro dischetto, potete specificare la lettura dell'unità con il nome del file:

```
B > ERA A: COPY.TXT ↵
```

Questo comando cancella il file COPY.TXT sull'unità A.

Per cancellare *tutti* i files di un dischetto, usate un nome di file generico per tutti i files:

```
B > ERA *.* ↵
```

Questo comando cancellerà tutti i files sull'unità B. Nell'MP/M e nelle nuove versioni del CP/M, il nome di file generico '*.*' verrà associato soltanto a tutti i files nell'*area utente corrente.* (Vedi :Capitolo 2 per una discussione delle aree utenti e di come vengono associati i nomi dei files).

LA COMPRENSIONE DEL CP/M

Il meccanismo interno

Avete appena imparato come caricare un sistema CP/M (o MP/M), creare dei files, cambiarne i nomi e cancellarli, copiare files e dischetti, e stampare files; ma cos'é accaduto effettivamente?

Il CP/M reagisce a una molteplicità di istruzioni: "cerca in questo dischetto un file con questo nome, visualizzalo, fanne una copia, ecc." Guardiamo più da vicino due di quelle operazioni: la stampa di un file e la copiatura.

Quando dite al CP/M 'TYPE SAMPLE.TXT', state eseguendo una serie di istruzioni riunite sotto il nome di 'TYPE'. Il sistema operativo accetta quello che avete digitato e lo legge quando premete il tasto RETURN. Legge 'TYPE' e va a cercare le istruzioni del programma TYPE. Poi legge 'SAMPLE.TXT' e va a cercare un file con quel nome. Cerca SAMPLE.TXT soltanto nell'unità corrente, perché non gli avete detto di cercare in un'altra unità. Quando trova SAMPLE.TXT il sistema operativo incomincia a mandare parti di questo al "dispositivo di console". Il file viene mandato blocco a blocco, fino a quando l'intero file è stato mandato. Poiché il vostro "dispositivo di console" è il terminale, ricevete il file sullo schermo. Se abilitate la stampante a ripetere (eco) qualsiasia cosa viene mandata al "dispositivo di console", allora esce anche su stampante.

Il CP/M è un programma complesso che esegue programmi di servizio più semplici. Il sistema riserva un'area di memoria dentro la memoria

interna del sistema (nell'involucro) per memorizzare temporaneamente dei programmi ed eseguirli. Per esempio, per copiare un file dovete eseguire il programma PIP. Quando digitate PIP, il sistema carica il programma PIP nella sua memoria interna e incomincia ad eseguirlo. PIP è chiamato comando transitorio o programma transitorio. È un programma (scritto in linguaggio macchina) che funziona come un comando tranne che non fa parte del "nucleo" del CP/M e deve esistere sul vostro dischetto sotto forma di un file con estensione '.COM' (PIP-.COM). Altri comandi, come STAT, SUBMIT, SYSGEN, ecc. sono anch'essi comandi transitori.

I comandi transitori

I comandi transitori (files di comando) sono in effetti programmi scritti in linguaggio macchina (linguaggio assemblatore). Dopo essere stato "assemblato" e provato, un programma in linguaggio macchina è stato caricato (usando il comando LOAD) nella memoria interna del sistema (chiamata TPA - Transient Program Area, cioé area dei programmi transistori). Il comando di caricamento ha anche fornito l'estensione '.COM', e il programma è diventato un comando transitorio. Ora potete eseguire il programma soltanto digitando il suo nome primario (senza l'estensione '.COM').

Per esempio PIP.COM è un comando transitorio. Eseguitelo digitando:

```
A > PIP ↵
```

Potete aggiungere al dischetto i vostri comandi (acquistati), come un elaboratore di testi, o un linguaggio come il BASIC (CBASIC, MICROSOFT BASIC, ecc.). Per esempio, WORDSTAR, un elaboratore di testi della Micropro International di San Rafael, California, ha due comandi transitori: WSU.COM e WSMSG.COM. WORDSTAR è eseguito digitando:

```
A > WSU ↵
```

WORDSTAR perciò diventa un'altra possibilità nel sistema CP/M (o MP/M). Un altro esempio potrebbe essere il Microsoft BASIC, che ha il comando transitorio MBASIC.COM. Per eseguire il sistema Microsoft BASIC, dovete soltanto digitare:

```
A > MBASIC ↵
```

Confrontate sempre i manuali forniti con il nuovo software per le istruzioni effettive di esecuzione.

Lo spegnimento del sistema

Quando siete pronti a spegnere il sistema, prima estraete il dischetto dei files (nell'unità B) e poi il Dischetto di Sistema (nell'unità A). Non spegnete il sistema con i dischetti ancora dentro all'unità, perché potreste cancellarli.

Dopo aver estratto i dischetti, potete tranquillamente spegnere il vostro sistema di elaborazione spegnendo i dispositivi accessori e poi il calcolatore stesso.

NOTA: non spegnete il sistema senza aver prima fatto le copie di salvataggio di ogni nuovo file che avete creato.

UNA LISTA DI CONTROLLO UTENTE

La seguente lista di controllo utente riassume le precauzioni e i procedimenti che dovreste sempre osservare. Sono molto importanti. Prendete tempo per imparare le istruzioni di questa lista prima di usare il calcolatore.

LISTA DI CONTROLLO UTENTE

L'ACCENSIONE DEL SISTEMA

Assicuratevi che:

☐ I dischetti siano fuori dalle unità dischi prima di applicare tensione.

☐ Il dischetto di sistema corretto sia disponibile.

☐ Uno o più dischetti bianchi siano disponibili.

☐ Tutti i cavi siano connessi in modo corretto.

☐ La stampante e il terminale siano correttamente predisposti.

L'USO DEL SISTEMA

È importante che:

☐ Facciate una copia di tutti i dischetti che usate.

☐ Salvate frequentemente il vostro file sul disco durante l'editing.

☐ Etichettiate immediatamente i dischetti con titolo, data e contenuti, usando un pennarello.

L'USO DI UN NUOVO PROGRAMMA

Dovreste:

☐ Fare una copia prima di usare il programma.

☐ Mettere via l'originale del nuovo programma in un posto sicuro.

AL TERMINE DEL LAVORO SUL SISTEMA

È importante che:

☐ Abbiate una copia di salvataggio di tutti i files creati.

☐ Togliete i dischetti dalle unità.

SOMMARIO

Ora avete imparato come accendere e spegnere il sistema, come far partire il CP/M e come usare i comandi e le possibilità di base del CP/M, come DIR, REN, ERA, PIP, ED, insieme alle funzioni speciali, come DEL e CTRL-C.

Avete anche imparato come stampare un file sulla stampante o sul video, e come fare copie dei vostri files e del CP/M.

Potreste sorprendervi di sapere che ora conoscete abbastanza del CP/M per eseguire la maggior parte dei programmi applicativi senza problemi. Comunque, se volete imparare di più sul vostro sistema operativo e sulle sue risorse, procedete nella lettura.

CAPITOLO 2

LE CARATTERISTICHE DEL CP/M E DELL'MP/M

INTRODUZIONE

Questo capitolo vi insegnerà tutti i comandi del CP/M, compresi ERA, REN, STAT, DIR, e i caratteri di controllo. Saranno trattati anche l'assemblaggio, il caricamento, la stampa, l'esecuzione, la correzione, il salvataggio dei programmi (ASM, LOAD, DUMP, DDT, SAVE), e l'esecuzione di un file di comandi (SUBMIT e XSUB). Verrà fornito un sommario delle possibilità di controllo dei comandi disponibili nel CP/M. Si esaminerà in dettaglio ogni comando e il modo di usarlo. Non è necessario a questo punto ricordare i comandi, ma dovreste esaminarli perché possono dimostrarsi subito utili.

Anche se siete soltanto un utente casuale del CP/M, dovreste imparare:

- I cinque caratteri di controllo usati più spesso (descritti nella sezione seguente).
- Come cancellare i files con ERA.
- Come cambiare i nomi dei files con REN.
- Come conoscere l'area del vostro file con STAT.
- Come copiare il vostro sistema CP/M con SYSGEN.

È utile, sebbene non assolutamente necessario, leggere questo capitolo una volta per intero. Tuttavia, quando usate il CP/M, dovreste leggere di nuovo questo capitolo se volete trarre pienamente vantaggio dalle risorse che il CP/M e l'MP/M offrono.

Il materiale presentato in questo capitolo vi metterà in grado di usare tutti i comandi del CP/M (e la maggior parte dei comandi dell'MP/M). Probabilmente farete riferimento frequentemente a questo capitolo, fino a quando sarete completamente familiarizzati con le azioni e le convenzioni di comando del CP/M. Troverete poi che il Capitolo 6 è un'utile guida di riferimento.

La descrizione dei comandi è basata prima di tutto sulla versione 1.4 del CP/M. La versione 2.2 del CP/M ha pochi miglioramenti, mentre la

versione 1.0 dell'MP/M ne ha molti. Poiché la maggior parte degli utenti ha la versione 1.4, focalizzeremo la nostra discussione e gli esempi del CP/M su quella particolare versione, e descriveremo la versione 2.2 e l'MP/M successivamente in una sezione speciale in questo capitolo. Se usate la versione 2.2 del CP/M o l'MP/M, dovreste leggere questa sezione prima. I comàndi modificati della versione 2.2 del CP/M e dell'MP/M sono anche inclusi nel sommario e nel Capitolo 6.

Dapprima saranno descritte le convenzioni usate dal CP/M: i caratteri di controllo, i comandi interni, i comandi transitori e i comandi dei files. Poi verranno presentati ciascuno dei comandi interni: DIR, TYPE, REN, ERA, SAVE; e i comandi transitori: SYSGEN, PIP, ED, STAT, ASM, LOAD, DUMP, DDT, SUBMIT, MOVCPM. Verranno descritte anche molte caratteristiche importanti del CP/M versione 2.2 e dell'MP/M.

COMANDI

Come convenzione per la descrizione del formato dei comandi in questo libro, ciascun comando verrà presentato in lettere MAIUSCOLE. Gli "argomenti" che devono essere forniti con i comandi verranno indicati in lettere minuscole. Se gli "argomenti" sono in *corsivo*, allora sono opzionali. Se due "argomenti" sono racchiusi tra parentesi { }, dovete scegliere tra essi. Queste convenzioni sono usate anche nel Capitolo 6.

Il simbolo ↵ indica il tasto RETURN a volte chiamato CR, che conferma il comando al sistema (cioé provoca l'esecuzione del comando) e muove anche il cursore alla linea successiva (genera un ritorno carrello/interlinea). Il simbolo ↵ indica il tasto CTRL (control) ed è usato per indicare di premere il tasto CTRL mentre si preme un altro tasto (per esempio ↵ X significa "premi il tasto CTRL e il tasto X").

Incominciamo a introdurre i caratteri di controllo. Essi vengono descritti nella Fig. 2.1.

I caratteri di controllo possono essere usati durante la digitazione di una qualunque linea di comando CP/M. Per esempio, supponiamo che incominciate digitendo:

```
A > PIP B: NEW.NCD =
```

Tuttavia in realtà volevate digitare NAD, non NCD. Se premete il tasto DEL tre volte, potete cancellare gli ultimi tre caratteri:

```
A > PIP B:NEW.NCD = = DC
```

Funzione	Tasto (i) da digitare	Operazione
Cancella l'ultimo carattere digitato:	RUBOUT (chiamato anche DELETE o CTRL-H•	Il cursore ripete il carattere (eco)
Cancella l'intera linea:	CTRL-U	Il cursore si muove alla linea successiva e stampa '#' per segnalare un nuovo comando.
	CTRL-X •	Il cursore si muove indietro all'inizio della linea, cancellando la linea
Ristampa la linea di comando corrente:	CTRL-R	Stampa una linea pulita. Usato dopo le correzioni
Trasmette (esegue) la linea di comando:	RETURN (talvolta chiamato CR) o CTRL-M o CTRL-J (LINE FEED)	Il comando corrente viene eseguito. Il cursore si sposta alla linea successiva per il nuovo comando
Consente una linea di comando lunga (più lunga della lunghezza della linea del terminale)	CTRL-E	Il cursore si sposta alla linea successiva senza trasmettere o eseguire il comando
Rilancia il sistema (partenza a caldo):	CTRL-C	Rilancia CP/M. Interrompe i processi in esecuzione
Rilancia il sistema (partenza a caldo) per permettere di scrivere su un nuovo dischetto:	CTRL-C	Vi permette di scrivere su un nuovo dischetto sostituendo quello precedente
Abortisce un processo (programma) MP/M in esecuzione:	CTRL-C ••	Abortisce il processo e rilancia l'MP/M
Termina le operazioni PIP (copiature)	RETURN (CR) o CTRL-M	Termina PIP. Restituisce il controllo al CP/M o MP/M
• indica soltanto CP/M versione 2.2 e MP/M •• indica soltanto MP/M		

Figura 2.1: Caratteri di controllo (continua)

Funzione	Tasto (i) da digitare	Operazione
Termina le operazioni DDT (debugger):	G0 seguito da RETURN	Restituisce il controllo al CP/M o MP/M
Termina le operazioni ED (programma editor):	E seguito da RETURN	Salva il testo nel file sorgente e nel buffer
Termina l'operazione ED di inserzione:	CTRL-Z	Restituisce il controllo a ED
Distacca un programma (processo) da un terminale:	CTRL-D••	Restituisce il controllo all'MP/M (il programma funziona staccato dal terminale)
Invia alla stampante tutto quello che appare sul terminale:	CTRL-P	La stampante stampa quello che digitate e quello che appare sullo schermo CRT
Interrompe la trasmissione alla stampante:	CTRL-P	Controllo On/Off per l'eco della stampante
Ferma la visualizzazione rapida sullo schermo, per una facile lettura:	CTRL-S	Interrompe lo schermo fino a quando premete di nuovo ↑ S
Fa ripartire lo schermo	CTRL-S	Controllo On/off per l'interruzione dello schermo
• indica soltanto CP/M versione 2.2 e MP/M •• indica soltanto MP/M		

Figura 2.1: Caratteri di controllo

e compilare il comando tranne che per RETURN:

A > PIP B:NEW.NCD = = DCAD = A:OLD.NAD

Sembra confuso, ma provate sul calcolatore. Usate CTRL-R, e otterrete una linea nuova pulita:

A > PIP B:NEW.NAD = A:OLD.NAD ↵

La linea di comando digitata adesso è corretta. Potete digitare un RETURN e il comando sarà eseguito.

rubout/delete	cancella e ristampa l'ultimo carattere
CTRL-U o CTRL-X	cancella la linea
CTRL-R	ristampa la linea
CTRL-E	continua sulla linea successiva
CTRL-C	rilancia il CP/M

(Nota: sono anche disponibili CTRL-P e CTRL-S per il controllo della stampante).

Figura 2.2: Riassunto dei controlli di editing

Similmente, se digitate un comando errato, potete cancellare tutta la linea con un CTRL-U (vedi Fig. 2.1). I cinque caratteri di controllo disponibili per l'input su tutte le versioni del CP/M sono riassunti nella Fig. 2.2. Fate in modo di familiarizzarvi con tutti questi caratteri.

Riferendoci di nuovo alla Fig. 2.1, notate che contiene la lista di tutti i caratteri di controllo disponibili, non solo per il CP/M ma per tutte le versioni del CP/M quando usate i comandi PIP, ED e DDT. È importante conoscere i comandi disponibili con il CP/M. Indipendentemente dalla versione (CP/M o MP/M), ci sono almeno cinque comandi interni:

TYPE
DIR
REN
ERA
SAVE

e molti comandi "transitori" standard, che devono essere presenti come files sul dischetto di sistema per poter essere eseguiti quando sono chiamati. Questi comandi transitori sono:

SYSGEN
ED
PIP
ASM
LOAD
DUMP
DDT
SUBMIT
MOVECPM*
STAT

Tutti questi comandi sono elencati nella Fig. 2.3. Ciascuno sarà descritto in dettaglio nel caso del capitolo.

Gestione dei dispositivi	Comando	Operazione
Stampa e modifica gli assegnamenti dei dispositivi:	STAT ↵ STAT assegnamenti dei dispositivi ↵	*NOTA: STAT VAL: ↵ stampa i possibili comandi STAT
Cambia l'unità a dischi corrente	u: ↵	u è il simbolo della nuova unità (A, B, C, D)
Copia da un disco all'altro:	PIP ↵ PIP destinazione = sorgente ↵	Programma di scambio tra le periferiche Nomi dei files destinazione
Stampa, perfora, copia, unisce e compie altre operazioni con i dispositivi:	PIP parametri	Vedere la trattazione di PIP
Trasmette files alla stampante: ••	SPOOL nome di file nome di file... ↵	
Interrompe e cancella la coda in trasmissione: ••	STOPSPLR ↵	
Stampa il numero di console (terminale): ••	CONSOLE ↵	
Abilita l'operatore di sistema a cambiare i dischi: ••	DSKRESET ↵	Il comando chiede agli altri utenti di abilitare un cambio di disco
Fa una copia del sistema CP/M (genera un nuovo dischetto di sistema):	SYSGEN ↵	Lancia il programma SYSGEN
Crea una versione diversa del sistema CP/M (lo riconfigura per una differenza e estensione di memoria):	MOVCPM ↵	Lancia il programma MOVCPM
Fa una copia del sistema MP/M o riconfigura il sistema: ••	Usate i comandi MPMLDR o SYSGEN soltanto dopo aver letto le informazioni importanti nella "Guida alle modifiche dell'MP/M" della documentazione della Digital Research's	
Stampa lo stato d'esecuzione del sistema MP/M: ••	MPMSTAT ↵	Stampa le informazioni sui processi

Figura 2.3: Comandi del CP/M (continua)

Gestione dei dispositivi	Comando	Operazione
Cambia l'area utente: •	USER n ↵	n è il numero di un'area utente - vedere la sezione "Aree Utente" in questo capitolo Senza n, USER stampa l'area utente corrente
Stampa e assegna la data e l'ora ••	TOD ↵ TOM mm/gg/aa hh:mm:ss ↵	Stampa l'ora e la data Assegna l'ora e la data
Gestione dei files	**Formato del comando**	**Operazione**
Creazione di un file su un disco:	ED nomefile ↵	Il programma editor del CP/M crea un file di testo Potete usare anche un qualsiasi altro programma editor
Cambia nome a un file:	REN nomenuovo=nomevecchio ↵	Cambia il nome di un file
Cancella un file:	ERA { nomefile / maschera } ↵	Cerca il nome del file o quelli che si adattano alla maschera
Copia un file:	PIP nomenuovacopia=nomevecchiacopia ↵	Copia un file dandogli un nome
Copia da un'unità all'altra (copia singola):	PIP u: nomenuovacopia=u: nomevecchiacopia (dove u è la lettera dell'unità) (vedere descrizione di PIP per le abbreviazioni)	
Esegue molte operazioni di copiatura:	PIP ↵ *nomenuovacopia=nomevecchiacopia ↵ *u: nomenuovacopia=u: nomevecchiacopia ↵ * ↵	Usate RETURN per terminare PIP

Figura 2.3: Comandi del CP/M (continua)

Gestione dei files	Formato del comando	Operazione
Stampa il contenuto di un file di testo:	TYPE nomefile	Il contenuto del file viene stampato sul video o sulla stampante usando ↑ P
Modifica il contenuto di un file di testo:	ED nomefile	Il programma ED permette di modificare i files di testo
Elenca i nomi di files nella direttrice:	DIR { *nomefile* / *maschera* }	DIR da solo stampa tutti i nomi dei files
Stampa l'estensione del file e del disco:	STAT { u:nomefile / u:maschera }	Stampa l'estensione dei files e lo spazio occupato
	STAT*u:*	Stampa lo spazio libero sul disco corrente o, opzionalmente, sul disco nell'unità u
Stampa gli attributi del file • (indicatori):	STAT *u*: nomefile	Stampa gli attributi come R/O per sola lettura o SYS per file di sistema
Crea un programma in linguaggio assembler:	ED nomefile	Il programma editor del CP/M crea files di testo. Un qualsiasi programma editor basato sul CP/M può essere usato per creare files di testo

Gestione dei programmi	Comando	Operazione
Traduce un programma in linguaggio assembler:	ASM nomefile	Crea un "file oggetto" in linguaggio macchina "HEX"
Produce un programma rilocabile: ••	GENMOD nomefile .HEX nomefile .PRL $bytes ↵	Produce un file. PRL da un file. HEX, con la memoria supplementare opzionale espressa in cifre esadecimali da da$bytes (vedere Capitolo 3)

Figura 2.3: Comandi del CP/M (continua)

Gestione dei programmi	Comando	Operazione
Crea un nuovo comando transitorio:	LOAD nomefile	Crea un file di comandi eseguibile con un'estensione .COM da un "file oggetto" in linguaggio macchina "Hex"
Esegue un comando transitorio o programma:	programma ↵	"programma" è il nome di un file con estensione .COM senza il '.COM'
Stampa un "file oggetto" (un file in linguaggio macchina "hex"):	DUMP nomefile ↵	Usato per i file esadecimali
Salva una copia di un comando transitorio o di un programma (dopo la creazione o l'esecuzione):	SAVE p nomefile ↵	p è il numero di pagine. Una pagina corrisponde a 256 bytes
Consente la messa a punto di un programma (usando il debugger del CP/M e dell'MP/M):	DDT nomefile RDT nomefile ••	RDT è il nome del debugger rilocabile dell'MP/M
Esegue un file di comandi (o programmi):	SUBMIT file par1 par2 ...	par significa parametro. Questo comando cerca un file contenente molti comandi, sostituisce i parametri ed esegue i comandi. Il sistema stampa i comandi in esecuzione (a meno che usiate • XSUB). Potete abortire l'operazione in qualsiasi momento usando RUBOUT. • XSUB, se inserito come primo comando del file, esegue i comandi e accetta dati per i programmi (se i programmi sono predisposti per l'ingresso dei dati in un buffer).
Assegna l'ora di esecuzione del programma:	SCHED mm/gg/aa hh:mm	mm/gg/aa è la data e hh:mm è l'ora
• indica soltanto CP/M versione 2.2 e MP/M •• indica soltanto MP/M		

Figura 2.3: Comandi del CP/M

Per usare efficacemente le risorse del CP/M e i suoi comandi standard, occorre capire due concetti:

- la differenza tra comandi interni e comandi transitori;
- le convenzioni sui nomi dei files.

Vediamo di esaminarli.

COMANDI INTERNI E COMANDI TRANSITORI

Come sapete, quando vedete il prompt di sistema del CP/M (una lettera che indica l'unità a dischetto seguita da una parentesi angolare destra, ad esempio 'A >'), potete digitare un qualsiasi comando interno e il calcolatore risponderà immediatamente. Simbolicamente tutti i comandi prendono questa forma:

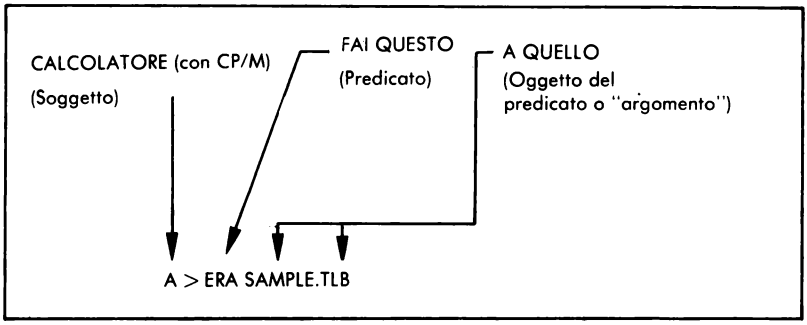


Figura 2.4: "Calcolatore, cancella il file SAMPLE.TLB."

Alcuni comandi sono semplici e alcuni sono complessi. Tutti i comandi sono in effetti programmi in linguaggio assembler (scritti nel linguaggio che il calcolatore "capisce"). Alcuni sono comandi "interni" (non elencati nella direttrice) e alcuni sono comandi "transitori" (elencati nella direttrice). Potete sempre eseguire i cinque comandi "interni" (DIR, ERA, REN, SAVE, TYPE) perché sono inseriti nel programma che costituisce il sistema operativo del CP/M. Potete eseguire comandi transitori soltanto se esistono come files di comando (. COM) sul vostro disco. Un comando transitorio è in effetti un programma in linguaggio assembler che può essere copiato, cancellato e spostato e oltre a ciò essere anche eseguito come un comando. Potete crearvi i vostri comandi transitori (programmi) se sapete come programmare un calcolatore. Il nome del file che contiene il comando transitorio ha l'estensione '. COM' (ad esempio PIP.COM), ma non dovete digitare 'COM' quando lo eseguite come comando. Se volete copiare, cancellare o spostare il file,

dovete tuttavia specificare l'intero nome del file con l'estensione 'COM'.

Per sapere se il comando transitorio che vi serve è sul vostro disco, usate il comando DIR. I comandi transitori standard forniti con il CP/M sono: ASM, ED, DUMP, LOAD, PIP, MOVCPM, STAT, SYSGEN, SUBMIT. I comandi MP/M sono *tutti* comandi transitori o "programmi residenti" come descritto nella sezione speciale "CP/M versione 2.2 e MP/M".

Sebbene siano descritti tutti i comandi transitori, ne userete probabilmente soltanto alcuni. Una tipica direttrice di sistema è indicata nella Fig. 2.5.

Direttrice di sistema	Richiesto	Fortemente consigliato	Utile	Opzionale
(CP/M esiste ma è invisibile a DIR)	X			
SYSGEN	X			
PIP	X			
STAT		X		
ED			X	
MOVCPM				X
ASM				X
LOAD				X
DUMP				X
DDT				X
SAVE				X
SUBMIT				X
XSUB				X
WORD PROCESSOR			X	
PROGRAMMI APPLICATIVI		X		

NOTE:

1. Per precauzione, non memorizzare dati o files di testo sul dischetto di sistema.
2. Inoltre, usate la protezione contro la riscrittura sul dischetto di sistema.

Figura 2.5: Una tipica direttrice del dischetto di sistema

I NOMI DEI FILES

Ogni direttrice di sistema contiene un nome di file per tutti i files sul dischetto. Un esempio del formato di un nome di file è NOME.TTT, dove NOME può avere fino a otto caratteri e .TTT è il tipo di estensione.

I nomi di files possono contenere lettere, numeri e caratteri speciali. Non possono però contenere i seguenti simboli:

< > . , : = ; * ? [ ]

per esempio:

PROG/22.BAK	è corretto
PROG=22.BAK	è scorretto

Estensioni

I nomi di file normalmente hanno un'estensione (tre caratteri a destra del punto). Queste estensioni distinguono differenti tipi di files. L'estensione è richiesta per molti tipi di file; altre estensioni sono richieste per convenienza (vedere Fig. 2.6).

Quando usate un nome di file come argomento di un comando, dovete digitare il nome per intero, compresa l'estensione, se esiste. L'unica eccezione è quando usate un file come comando transitorio (questo file deve avere l'estensione .COM internamente, anche se non è necessario digitare l'estensione per eseguire un comando).

Maschere dei nomi di file

A volte volete che un comando operi su più di un file o su tutti i files contemporaneamente. Se il comando lo permette, il suo formato indica che una *maschera dei nomi di file* (abbreviata in *maschera*) può essere sostituita al posto di un *nome di file* effettivo come argomento del comando. Per esempio, il formato del comando ERA è:

ERA { Nome-del-file / Maschera }

Questo significa che dovete scegliere fra un nome di file effettivo e una maschera come argomento di ERA.

Una *mashera dei nomi di file* è un gruppo di caratteri usato per riferirsi a più files contemporaneamente. I caratteri possono essere lettere, cifre, un punto e due simboli speciali '*' e '?'. Questi caratteri e simboli possono essere usati per selezionare convenientemente i nomi di molti files da sottoporre al comando. Pertanto è importante scegliere i caratteri che sono comuni a tutti i nomi di files.

Estensione	Tipo	Esempio
COM	Richiesta per un file contenente un comando transitorio (programma).	PIP.COM LOAD.COM
ASM	Richiesta per i files contenenti i sorgenti testi) in linguaggio assembler, usati con il comando ASM.	PROG1.ASM PATCH.ASM
PRN	Richiesta per il file di "listing" del programma in linguaggio assembler.	PROG1.PRN PATCH.PRN
PRL	Richiesta per i programmi rilocabili dell'MP/M.	RDT.PRL
HEX	Richiesta per i file contenenti programmi formato "hex" (linguaggio macchina), pronti per il comando LOAD.	PROG1.HEX PATCH.HEX
RSP	Richiesta per i "programmi di sistema residenti" dell'MP/M.	SPOOL.RSP
BAS	Richiesta per i files contenenti i sorgenti (testi) dei programmi in BASIC.	PROGBAS.INT
INT	Richiesta per i files intermedi eseguibili dei programmi BASIC (già compilati).	PROGBAS.INT
BAK	Creata da ED (editor) come copia di salvataggio di un file prima che sia modificato.	LETTER.BAK
$$$	Files temporanei (scratch) creati e normalmente cancellati da ED e da altri programmi.	LETTER.$$$
SUB	Files di testo con comandi interni o transistori del CP/M o programmi, da eseguirsi in modo batch dal programma SUBMIT.	TRANSFORM.SUB

Figura 2.6: Tipi di estensioni

Provate usando il simbolo '?'. Esso corrisponde a un carattere qualsiasi, ma *soltanto uno per ogni* '?' e *nell'esatta posizione* in cui è posto il simbolo '?'. Ecco alcuni esempi:

Questo	Corrisponde a questi nomi	Ma non a questi
S?MPL?	SAMPLE SIMPLE SIMPLY	SIMPL SAMPLEY
A?B?C	AABBC ACBCC	AABBCC ABCCC

Né '*' '?' possono corrispondere a un punto. Ecco alcuni esempi di più di un simbolo nelle maschere dei nomi di files:

Questo	Corrisponde a questi nomi	Ma non a questi
T???Y.*	TEDDY.COM	TINY.COM
	TARBY.ASM	TARABY.ASM
	TILLY	TONY
?????????.???		
o	qualsiasi nome di file	
**		

Ricordate che nel Capitolo 1 abbiamo usato una maschera per copiare l'intero dischetto. Il comando era:

A > PIP B: = A: *.* [V] ↵

dove '*.*' rappresenta "tutti i files".
NOTA: usare il simbolo '*' all'inizio del nome di file '*AB.*' è pericoloso! Cercate di non mescolare '?' e '*' nelle maschere dei nodi di files. Le maschere sono comunemente usate con il comando PIP per fare copie di molti o tutti i files contemporaneamente (o per mandare più di un file a un dispositivo).

SPAZI

Il CP/M standard richiede che un comando sia seguito almeno da uno spazio prima di ogni argomento. Nell'esempio precedente:

A > PIP B: = A: *.* [V] ↵

PIP deve essere seguito da uno o più spazi. (Può anche essere preceduto da uno spazio). L'argomento 'B:' può essere seguito da uno spazio ma questo è opzionale.

Similmente, uno spazio può essere usato opzionalmente dopo '=' o dopo 'A:'. In alcune versioni del CP/M (come North Star), *tutti* gli spazi sono opzionali. Quando si usa una versione "standard" del CP/M, comunque, ricordate che gli spazi sono obbligatori dopo un comando.

Spazi addizionali possono essere usati per separare sottoelementi di un comando. Ecco un esempio che usa spazi addizionali:

```
A > PIP B: = A: *.* [V] ↵
```

COMANDI INTERNI

Introduzione

Ricordate che nel Capitolo 1 abbiamo presentato brevemente tutti i comandi interni standard del CP/M (ad eccezione di SAVE). Forniremo ora esempi specifici del loro uso.

DIR (Direttrice)

Il comando di direttrice è usato per stampare una lista di tutti i files presenti sul dischetto. Per fare una lista di tutti i files sul dischetto nell'unità A, digitate:

```
A > DIR ↵
```

Per fare una lista dei files sul dischetto nell'unità B, digitate:

```
A > DIR B: ↵
```

(Questo metodo è il più veloce e rimanete nell'unità A). Oppure, potete digitare la sequenza:

```
A > B: ↵
B > DIR ↵
```

(Questo metodo è più lungo, ma rimanete nell'unità B dopo avere eseguito l'ultimo comando). Per cercare un file specifico sul dichetto nell'unità B, digitate:

```
A > DIR B:SPECIFIC.NAD ↵
```

Per fare una lista di tutti i files con l'estensione NAD sul dischetto nell'unità A, digitate:

```
A > DIR * .NAD ↵
```

Usate frequentemente il comando DIR per verificare quali files ci sono sul dischetto. Bisognerebbe anche verificare la direttrice dopo aver creato o cancellato un file.

Con il CP/M 2.2, la lista della direttrice è su 4 colonne. Ecco un esempio:

```
A > DIR ↵
A : MOVCPM  COM : ASM      COM : DDT      COM : DUMP     COM
A : ED      COM : LOAD     COM : PIP      COM : STAT     COM
A : SUBMIT  COM : SYSGEN   COM : XSUB     COM : DISKFED  LIB
A : DUMP    ASM : SINGLE   ASM : COPY     COM : LIST     COM
A : FORMAT  COM : SAVEUSER COM : USER     ASM : MEMR     COM
A : FILECOPY COM : SETDRIVE COM : READ-ME DOC : CONFIG   COM
```

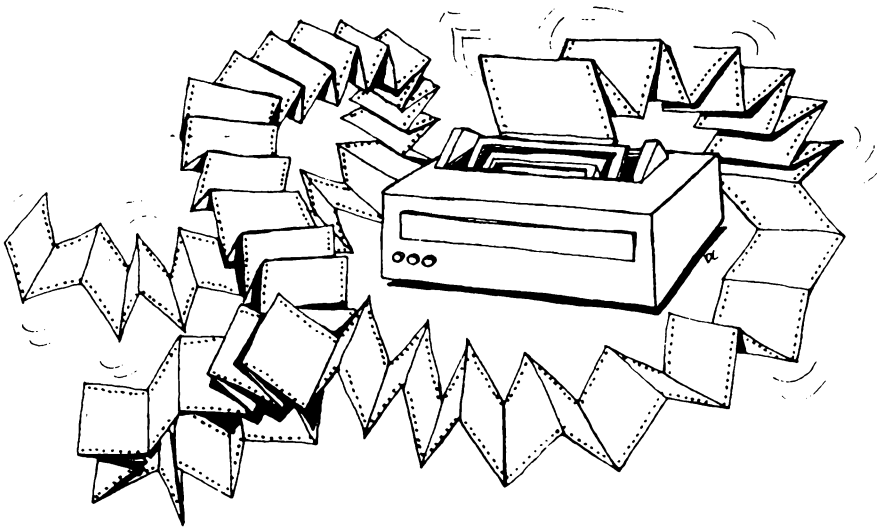
TYPE

Il comando TYPE può essere usato per stampare un qualsiasi file ASCII sullo schermo. Questo è un modo veloce e conveniente di esaminare un file di testo. Il formato è:

```
TYPE u:nome.est
```

dove il nome u dell'unità è opzionale. Se volete che il file sia stampato sulla stampante, premete un CTRL-P prima di dare il comando. Usando la stampante, comunque, la stampa è più lenta.

Il comando TYPE può essere usato per verificare qualsiasi testi abbiate creato. Non potendo fornire le prestazioni di un programma di word processing (gestione dei testi) o di un programma di liste di indirizzi, TYPE stampa il file come è effettivamente memorizzato nella memoria del calcolatore. TYPE solitamente è una via rapida per esaminare un file di testo che sia stato accidentalmente danneggiato. È un valido strumento per un esame veloce sullo schermo di un file appena creato.



REN (Rename)

Questo comando vi permette di cambiare il nome di un file. Il formato è:

REN nuovo = vecchio

Per esempio, supponiamo che FILE11.TXT sia sul dischetto nell'unità A. Se digitate:

A > REN FILE23.TXT = FILE11.TXT ↵

FILE11 prenderà il nome FILE23, e sul dischetto nell'unità A ci sarà soltanto FILE23. Il nome FILE11 sarà stato eliminato. Il file stesso, comunque, rimane intatto.

La specificazione del drive è opzionale. Per esempio l'istruzione:

A > REN B:FILE.TXT = FILE.BAK ↵

è equivalente a:

A > REN B:FILE.TXT = B:FILE.BAK ↵

Si noti che entrambi i files devono risiedere sulla stessa unità. La specificazione dell'unità può apparire sia a sinistra che a destra del segno uguale, e il nuovo nome di file non deve ancora esistere.

Una buona norma quando si lavora su un file è di inserire la data nel nome. Per esempio il file PUB può essere chiamato PUBMAG21. La vosta successiva si può modificare il nome del file in PUBMAG23. Alla fine di un lavoro sul calcolatore, modificate sempre il nome del file su cui avete lavorato al fine di:

1. Dargli un nome di identificazione.
2. Evitare possibili confusioni.
3. Inserire la data.

ERA (Erase)

ERA è usato per cancellare un file. Il suo formato è:

```
ERA u:nome
```

Dove u è il nome opzionale del drive. Un esempio di un comando ERA è:

```
A > ERA PROG.TXT ↵
```

dove PROG.TXT viene cancellato.

NOTA: il comando ERA può essere pericoloso! Usatelo con cautela!

ERA può anche essere usato per cancellare più di un file per volta. Per esempio, se volete cancellare tutti i files con l'estensione '.ASM' nel nome, dovete sostituire un simbolo generico al posto del nome del file:

```
A > ERA *.ASM ↵
```

In questo caso, la maschera è '*.ASM', e corrisponde a tutti i files il cui nome termina con '.ASM'. Il simbolo '*' corrisponde a un qualsiasi numero di caratteri, compresa l'assenza di caratteri. Tenete presente che può esserci soltanto un punto in ogni nome di file. Gli ultimi tre caratteri del nome (ASM) corrispondono soltanto alle lettere 'ASM' che appaiono in quelle esatte posizioni (le prime tre posizioni a destra dell'unico punto). Pertanto, in, quest'esempio, '*.ASM' corrisponde a 'ESEMPIO.ASM', 'UN ALTRO.ASM', '1.ASM', e '.ASM'. Non corrisponde a 'ESEMPIO', 'UN ALTRO.PRG', 'ASM.COM', o 'ASM' (notate che 'ASM' non è un'estensione, perché non occupa le tre posizioni a destra del punto).

Non usate il comando ERA per far pratica sulle maschere del files - usate invece il comando DIR. DIR fa solo una lista dei files che corrispondono alle maschere.

In pratica, vorrete spesso creare più copie di un file, come VERSIONE 1, VERSIONE 2, VERSIONE 3 o COPIA 1, COPIA 2. Per evitare confusione, ricordate di cancellare tutte le copie non necessarie prima di lasciare il terminale.

SAVE

Il comando SAVE è usato per memorizzare informazioni dalla TPA (memoria principale). Questo comando è più complesso e sarà descritto in dettaglio più avanti sempre in questo capitolo.

I COMANDI TRANSITORI

Introduzione

I dieci comandi transitori standard del CP/M sono: SYSGEN, PIP, ED, STAT, ASM, LOAD, DDT, SUBMIT, e MOVCPM. Tutti sono forniti con il CP/M come files con l'estensione .COM. Saranno descritti separatamente.

SYSGEN

Ricordate, dal Capitolo 1, che SYSGEN è usato per copiare CP/M da un disco all'altro. In particolare, è usato per copiare un dischetto di sistema. Quando ricevete la prima volta un dischetto di sistema e lo usate per costruire il vostro sistema CP/M, dovreste fare immediatamente una copia del sistema e riporre il dischetto di sistema in un posto sicuro. Potete usare il comando transitorio PIP per trasferire tutti i file '.COM' (e '.SYS') su un nuovo dischetto (vedi Capitolo 3), ma dovete anche copiare il "sistema" che risiede su tracce riservate del dischetto, per rendere il nuovo dischetto un "Dischetto di Sistema". Per far questo si deve usare il comando transitorio (programma) SYSGEN.

Il programma SYSGEN rende un dischetto ordinario dischetto di sistema (con il "sistema" residente su tracce riservate). Per far questo, SYSGEN porta "il sistema" nella memoria dal dischetto originale (*nella memoria* significa che viene portato nell'area di esecuzione dei programmi del calcolatore chiamata TPA cioé Transient Program Area), e poi lo scrive sul dischetto. SYSGEN porta "il sistema" nella memoria e lo sposta su un nuovo dischetto; oppure, SYSGEN trasferisce tutto ciò che vi è già in memoria su un altro dischetto.

La sequenza per usare SYSGEN è:

```
A >SYSGEN ↵
  SYSGEN VERSION n.n
  SOURCE DRIVE NAME (OR RETURN TO SKIP) A
  (se il CP/M è stato copiato nella memoria usando,
  per esempio, MOVCPM, potete digitare RETURN)
  SOURCE ON THEN TYPE RETURN ↵
  (con ciò si intende che vi si lascia il tempo di inserire
  un dischetto con il CP/M nel drive appropriato)
  FUNCTION COMPLETE
  (il CP/M ora è in memoria, pronto per essere scritto
  su un disco)
  DESTINATION DRIVE NAME (OR RETURN TO
  REBOOT) B
  (inserite il nuovo disco in B)
  DESTINATION ON B THEN TYPE RETURN ↵
  (il CP/M è scritto su B)
  FUNCTION COMPLETE
  DESTINATION DRIVE NAME (OR RETURN TO
  REBOOT) ↵
  (a questo punto potreste digitare B e fare un'altra
  copia. In questo caso finiamo premendo
  RETURN. ↵)
A >
```

Ricordate che il dischetto nell'unità B ora contiene soltanto il CP/M. Non contiene files (assumendo che fosse un dischetto vuoto). Il CP/M può essere copiato su un dischetto che conteneva già dei files; questo fatto non danneggia i files. Se il dischetto sull'unità B è vuoto, dovrete ora copiare i files separatamente. Se volete copiare tutti i files dal dischetto nell'unità A a quello dell'unità B, digitate:

```
A > PIP B: = A: *.* [V] ↵
```

Altrimenti, potete copiare i files desiderati uno alla volta, come occorre. Il formato di questa operazione è spiegato nel Capitolo 3.

A questo punto, dovreste provare il nuovo dischetto di sistema. Utenti MP/M: è importante notare qui che l'esempio riportato prima, che usa il comando PIP per copiare molti files in una volta, copia soltanto i files nell'area utente corrente. Il dischetto di sistema fornito può essere copiato usando questa forma di PIP, perché tutti i files necessari sono

nell'area utente zero, che è l'area "utente corrente" se non *usate mai* il comando USER. (Raccomandiamo di non usare l'area utente o il comando USER se non avete un sistema multiutente) - (vedi "Miglioramenti nel CP/M versione 2.2" nel Capitolo 3).

SYSGEN può essere usato per fare una copia del sistema che può essere lasciata nella TPA (Transient Program Area) e *non* trasferita su un nuovo dischetto. Per far questo, premete subito "RETURN" (per uscire) quando appare il messaggio 'DESTINATION DRIVE NAME (OR RETURN TO REBOOT)'.

Potete usare SYSGEN anche in congiunzione con MOVCPM per configurare una nuova versione del vostro sistema con uno spazio di memoria diverso. In questo caso, quando appare il primo messaggio 'SOURCE DRIVE NAME (OR RETURN TO SKIP)' premete subito RETUNR (↵) e SYSGEN assume che il sistema sia già caricato nella memoria (TPA). Quando usate MOVCPM, potete caricare il sistema nella memoria in preparazione di questo tipo di SYSGEN. Questo è descritto nel Capitolo 5.

PIP

PIP è il programma di trasferimento dei files, spiegato in dettaglio nel Capitolo 3.

ED

ED è l'editor, presentato in dettaglio nel Capitolo 4.

STAT

L'uso di STAT

STAT è usato per stampare lo stato o cambiare l'assegnamento dei dispositivi. Il comando STAT è un modo semplice per conoscere lo spazio disponibile sul disco. Può anche essere usato per vedere e modificare gli assegnamenti dei dispositivi.

Il semplice comando 'STAT' stampa le dimensioni dello spazio disponibile sul disco e lo spazio assegnato ai files. La più semplice forma del comando STAT è:

```
A > STAT ↵
A:R/W, SPACE: 144K
A >
```

Questa stampa mostra che rimangono 144K bytes sul dischetto dell'u-

nità A, cioé, 144K bytes che non sono utilizzati. La maggior parte dei dischetti a otto pollici contiene soltanto 224K bytes di spazio per i files. 'R/W' dice che il dischetto può essere *scritto* (cioé, potete creare nuovi files, riscriverli o cancellare vecchi files). 'R/W' significa "read-write" (lettura-scrittura) all'opposto di 'R/O', poiché è "write-protected" (protetto contro la scrittura).

Byte e records

Un *byte* è una locazione di otto bit in memoria (un bit è un "1" o uno "0"). 128 bytes formano un *record* (non necessariamente un "record di dati", ma un record di file del CP/M). Questa è anche l'ampiezza di un settore sul disco. 8 records corrispondono a 1024 bytes (cioé 1K). Si usano questi valori particolari perché possono essere espressi come potenze di 2 (numeri binari), cioé, possono essere rappresentati da un corrispondente numero di bits (n bits rappresentato fino a 2^n).

Un record di 128 bytes è una convenzione utile per leggere e scrivere sezioni di un file. I files esistono sul dischetto sottoforma di "blocchi" di 16K chiamati "estensioni". Queste "estensioni" non sono contigue (vicine una all'altra) sul dischetto. Ciascuna contiene sempre l'indirizzo di partenza della successiva, in altre parole, sono "concatenate". Non dovete trovarle voi; il Basic Disk Operating System (BDOS), una parte dell'intero sistema, gestisce l'allocazione dello spazio del file usando i blocchi di controllo del file (verrà descritto nel Capitolo 5). L'allocazione dello spazio è dinamica-il sistema assegna nuovo spazio al file quando voi (o il vostro programma) scrivete records nel file. Non avete mai bisogno di specificare una lunghezza massima.

L'assegnamento dei dispositivi con STAT

Il comando STAT fornisce informazioni utili sui dischi (dischetti), la memoria transitoria o "scratch-pad" (chiamata Area dei Programmi Transitori, o TPA, dall'inglese Transient Program Area), e gli assegnamenti dei dispositivi del sistema. Vi dà anche la possibilità di *cambiare* gli assegnamenti dei device e di proteggere le unità a dischetti contro la scrittura. Nella versione 2.2 del CP/M e nei sistemi MP/M, potete anche mettere degli indicatori sui *files* (come read-only), ricevere informazioni di stato addizionali sull'ampiezza di un file e sulla capacità di un disco e ricevere informazioni sulle aree utente.

La forma seguente di STAT può essere usate per stampare una lista dei possibili assegnamenti dei dispositivi con i nomi generici (logici) CON:, RDR:, PUN: e LST:

```
A > STAT VAL: ↵
```

Il sistema risponde stampando:

```
CON:    =TTY:    CRT:    BAT    UC1
RDR:    =TTY:    PTR:    UR1    UR2
PUN:    =TTY:    PTP:    UP1    UP2
LST:    =TTY:    CRT:    LPT    UL1
```

Vicino ad ogni nome logico di dispositivo (CON, RDR, PUN, LST,), STAT dà una lista dei possibili nomi fisici per quel dispositivo. Questi nomi corrispondo a quelli del sistema Intel MDS-800.

Il nome fisico di dispositivo può essere usato per un dispositivo che compie la stessa funzione di base; ad esempio, il dispositivo PTP: (perforatore di nastro di carta) potrebbe in effetti essere l'operazione di "registrazione" (scrittura) di un registratore a cassette.

Gli assegnamenti effettivi dei dispositivi possono essere stampati in ogni momento digitando questo comando:

```
A > STAT DEV: ↵
CON: = CRT:
RDR: = UR1:
PUN: = PTP:
LST: = TTY:
A >
```

Questo semplice esempio mostra che il dispositivo CON: è un CRT (schermo, dall'inglese Cathode Ray Tube), il dispositivo RDR: è un lettore definito dall'utente (numero 1), il dispositivo PUN: un perforatore di nastro di carte e il dispositivo LST: è una teletype.

Può essere che vogliate modificare occasionalmente i quattro nomi fisici di device. Questi assegnamenti possono essere cambiati usando il comando STAT seguente:

STAT log: = disp:, log: = disp:,...

dove *log*: è il nome logico del dispositivo (CON:, RDR:, PUN:, o LST:) e *disp*: è il nome fisico del dispositivo (PTP:, CRT:, UR1, ecc.). Per esempio il comando

```
A > STAT LST: = LPT: ↵
```

cambia il dispositivo LST: in una stampante a linee (LPT:), cosicché tutte le operazioni di trasferimento a LST: andranno ora alla stampante (a meno che il sistema sia rilanciato).

Si noti che il dispositivo CON: deve essere un dispositivo di ingresso uscita che possa sia mandare che ricevere dati (ad esempio un terminale con uno schermo e una tastiera). Il dispositivo RDR: deve almeno poter mandare dati (ingresso), e i dispositivi PUN: e LST: devono essere in grado di ricevere dati (uscite).

CP/M versione 1.4 e ultime versioni

Nel CP/M versione 1.4 e nelle versioni successive, un semplice comando STAT stampa il numero di bytes rimanendo nell'unità corrente. Potete ottenere queste informazioni anche per le altre unità usando questo formato:

```
A > STAT B: ↵
BYTES REMAINING ON B: 192K
B: R/O
A >
```

La stampa mostra che il dischetto dell'unità B è 'R/O', che significa "read only" (sola lettura). Potete soltanto leggere questo dischetto; ogni tentativo di scrivere genererà il messaggio di errore:

```
BDOS ERR ON B: READ ONLY
```

Quando ottenete questo messaggio nella versione 1.4, dovete soltanto premere un tasto sulla tastiera del terminale (ad es. RETURN), e il dischetto verrà riportato a 'R/W', che significa "read/write" (lettura-/scrittura). Ora potete leggere dal dischetto e scrivervi.

Potete mettere il dischetto in R/O (read-only) eseguendo la forma seguente del comando STAT:

```
STAT u: = R/O
```

dove u: è una qualsiasi unità a dischi.

Per stampare l'estensione di un file, usate questa forma di STAT:

STAT u:{nomefile / maschera}

dove u: è una lettera opzionale di unità per files che non sono nell'unità a dischi corrente. Se specificate un nome di file, STAT stamperà le informazioni per quel file. Se specificate una maschera di un nome di file, STAT cercherà i files che corrispondono alla maschera e li stamperà in ordine alfabetico con le informazioni di estensione. La stampa riportata sotto è un esempio:

```
A > STAT B:*.TXT ↵
RECS BYTS EXT D: FILENAME.TYP
8    1K   1  B:SAMPLE:TXT
4    1K   1  B:QUOTE:TXT
16   2K   1  B:CHAP1.TXT
A >
```

Il campo RECS indica quanti records di 128-byte sono stati assegnati al file (fino a questo momento), il campo BYTS indica quante migliaia di bytes sono state assegnate al file (1K corrisponde a 1024 bytes, e 128 volte RECS è uguale al numero di bytes assegnati, per cui BYTS è uguale al risultato di 128 volte RECS per 1024). Il campo BYTS è l'indicazione più accurata dello spazio occupato.

Il campo EX indica il numero di "estensioni" di 16K rimanenti nel file (il numero in BYTS divisero per 16). Nella versione 1.4 questo può corrispondere al numero di punti di ingresso nella direttrice di un disco per lo stesso file, perché un punto di ingresso nella direttrice (un blocco di controllo del file) può indirizzare soltanto fino a 16K. Il CP/M crea automaticamente più punti di ingresso e blocchi di controllo dei files quando il file è esteso.

I blocchi e i records nel CP/M

Il CP/M lavora sempre con un minimo di 8 record quando assegna lo spazio. Sui dischetti standard IBM, un record ha 128 bytes. Questo è il motivo per cui la più piccola quantità di spazio assegnata dal CP/M è 1K, come indicato nell'esempio di STAT appena riportato.

I dischetti in doppia densità, invece, possono avere records di 256

bytes. In questo caso, la più piccola quantità di spazio assegnata dal CP/M è 2K, anche se il file può usare soltanto una piccola porzione di quello spazio. Se si usa un disco rigido, la più piccola quantità di spazio può essere 4K.

STAT nel CP/M versione 2.2 e nelle versioni successive

Il programma STAT nelle versioni 2.0 e 2.2 del CP/M ha un certo numero di miglioramenti, comprese stampe più complesse. Se digitate semplicemente 'STAT ↵', la stampante corrisponderà a quella delle versioni precedenti del CP/M. Ecco un esempio:

```
A > STAT *.* ↵
Recs   Bytes   Ext   Acc
  64      8K     1   R/W A:ASM.COM
   8      1K     1   R/W A:BOOTHD.COM
  20      3K     1   R/W A:CONFIG.COM
  22      3K     1   R/W A:COPY.COM
   6      1K     1   R/W A:SAVEUSER.COM
```

La forma 'STAT VAL: ↵', invece, produce la stampa seguente:

```
Temp R/O Disk      : u: = R/O
Set Indicator      : u:nomefile.lst $R/O $R/W $SYS
                     $DIR
Disk Status        : DSK: u:DSK:
User Status        : USR:
Iobyte Assing      :

CON:   =TTY:   CRT:   BAT:   UC1:
RDR:   =TTY:   PTR:   UR1:   UR2:
PUN:   =TTY:   PTP:   UP1:   UP2:
LST:   =TTY:   CRT:   LPT:   UL1:
```

Si noti che le ultime quattro linee sono identiche a quelle stampate nel CP/M versione 1.4. Le linee sopra di esse mostrano i possibili comandi

STAT e le cose che fanno. Per assegnare l'attributo R/O all'intero *disco*, usate la versione del comando STAT 'u: = R/O' (come nel CP/M versione 1.4), dove 'u:' è un indicatore di unità (A:, B:,..., Y:). Per assegnare gli attributi $R/O, $R/W, $SYS o $DIR di un *file*, usate il formato di STAT descritto nella sezione sul CP/M versione 2.2 e sull'MP/M in questo capitolo. Per stampare lo stato del disco (dischetto) corrente o di quello in un'altra unità, usate il formato 'STAT DSK:' o 'STAT u:DSK'. Per stampare l'area utente corrente (e altre aree utenti presenti nel sistema), usate il formato 'STAT USR:' (un esempio sarà mostrato più avanti in questa sezione).

Nella versione 2.2 potete aggiungere l'argomento $S al comando STAT per stampare l'estensione dei files usando il seguente formato:

STAT u: { nomefile / maschera } $S

$S è un campo opzionale che provoca la stampa dell'estensione. Potete specificare sia un nome di file (compresa l'estensione), sia una maschera (nome di file generico per molti nomi di files). Opzionalmente potete specificare un'unità *u:* per stampare files residenti su un dischetto in un'altra unità. Ecco alcuni esempi:

```
A > STAT PIP.COM $S ↵

  Size    Recs    Bytes    Ext          Acc
    55      55      12K      1    R/O A:PIP.COM
```

Il campo 'Size' (spazio) dice quanti records (unità di 128 bytes) sono assegnati al file, ma questo è uno spazio "virtuale", perché il file potrebbe non usare ancora tutto lo spazio. Il campo 'Recs' somma il numero di records in ciascuna estensione (un'estensione è un blocco di 16K). Se il file è stato costruito sequenzialmente, questi due campi dovrebbero essere identici (e corrispondere al campo 'RECS' nella versione 1.4 del CP/M).

Il campo 'Bytes' vi dà l'unica indicazione accurata per i files ad accesso casuale - il numero effettivo di bytes assegnati al file. Questa indicazione corrisponde esattamente ai campi 'Size' e 'Recs' dei files sequenziali. I files ad accesso casuale crescono quando vi si scrivono i dati, perciò il campo 'Bytes' dà lo spazio in bytes assegnato in un certo momento, e il campo 'Recs' somma il numero di records in ciascuna estensione - sebbene un'estensione possa avere "buchi" non assegnati che non sono

stati ancora riempiti con dati. Il campo 'Size' indica in modo più accurato il "numero logico di records" in un file.

Il campo 'Ext' dice quante estensioni (blocchi di 16K) sono assegnati al file. Diversamente dalla versione 1.4, un punto d'ingresso nella direttrice (blocco di controllo del file) di un disco può indirizzare fino a 128K bytes (8 estensioni logiche) invece che soltanto 16K (un'estensione logica). Le indicazioni stampate, invece, corrispondono alla stampa della versione 1.4 per rendere compatibili le due versioni.

Il campo 'Acc' dice quale tipo di accesso è permesso - R/O (sola lettura) o R/W (lettura-scrittura). Questi tipi di accesso corrispondono agli attributi di file $R/O e $R/W descritti nella sezione sul CP/M versione 2.2 e sull'MP/M (in questo capitolo). Potete cambiare gli attributi di un file fornendo un attributo come argomento del comando STAT (al posto di '$S' nella definizione riportata sopra). Per esempio:

```
A > STAT PROG.ASM $R/O ↵
A > STAT FILE.TXT $R/W ↵
```

Il primo comando STAT assegna l'attributo $R/O (sola lettura) al file PROG.ASM. Il secondo comando STAT assegna l'attributo $R/W (lettura-scrittura) al file FILE.TXT.

Il formato generale di questo comando è:

```
                ⎧ nomefile ⎫   ⎧ $R/O ⎫
STAT u:         ⎪          ⎪   ⎪ $R/W ⎪
                ⎩ maschera ⎭   ⎩ $SYS ⎭
                               ⎩ $DIR ⎭
```

Il seguente è un altro esempio, dove ED.COM è un file a sola lettura, PIP.COM è a sola lettura e *di sistema* ($SYS) e TEXT.NAD è un file ad accesso casuale che ha l'attributo di lettura-scrittura.

```
 Size     Recs     Bytes    Ext    Acc
   48       48        6K     1    R/O  A:ED.COM
   55       55       12K     1    R/O  (A:PIP.COM)
65536      128        2K     2    R/W  A:TEXT.NAD
```

Si noti che PIP.COM è tra parentesi per indicare che l'attributo $SYS (sistema). I files non in parentesi hanno l'attributo automatico $DIR (direttrice). Gli attributi sono descritti più avanti nel corso del capitolo nella sezione sul CP/M 2.2 e MP/M.

Per stampare informazioni sul disco (dischetto) corrente (o in un'altra unità), usate la forma 'STAT u: DSK', dove u è un indicatore opzionale di unità (A, B, C, ..., P) per un'unità non corrente. Ecco un esempio di stampa:

```
A > STAT DSK: ↵
A:    Drive Characteristics
3888: 128 Byte Record Capacity
486:  Kilobyte Drive Capacity
128:  32 Byte Directory Entries
128:  Checked Directory Entries
128:  Records/Extent
16:   Records/Block
52:   Sectors/Track
2:    Reserved Tracks
```

Questo esempio di stampa si riferisce a un dischetto a doppia densità nell'unità corrente (l'unità A in questo caso). La capacità totale in record (128 bytes per record) è di 3888 records (approssimativamente 486K bytes). La capacità del drive in migliaia di bytes (Kilobytes) è indicata in 486 (senza tenere conto delle aree riservate del dischetto). Il numero di punti d'ingresso di 32 bytes nella direttrice corrisponde al numero effettivo di blocchi di controllo dei files (FCB) memorizzati sul dischetto. Il numero di punti d'ingresso verificati ("checked") è solitamente identico al numero dei punti d'ingresso della direttrice per i supporti rimovibili (come i dischetti), perché il sistema deve verificare i punti d'ingresso per rilevare un cambiamento di dischetti (questo numero è solitamente zero per i dischi non rimovibili). Il sistema verifica i punti d'ingresso a meno che non sia intervenuta una partenza a caldo (↑C) o una partenza a freddo (reset di sistema o "bootstrap a freddo"). Il numero di records per estensione ('Records/Extent') indicato nella stampa dà la capacità di indirizzamento di ciascun punto d'ingresso della direttrice; cioé quanti records (segmenti di 128 bytes) possono

essere indirizzati da un punto d'ingresso della direttrice (blocco di controllo sul disco). La stampa stabilisce che 128 records possono essere indirizzati da un singolo punto d'ingresso della direttrice (128 records corrispondono a 128 volte 128 bytes, o approssimativamente 16K bytes). Il numero di records per blocco ('Records/Block') indicato nella stampa dice il numero di records assegnato ad ogni settore (16 records corrispondono a 2048 bytes, o 2K bytes per blocco). Questa indicazione è seguita dal numero di settori fisici ("blocchi") per traccia ('Sectors/Track'), e dal numero di tracce riservate sul disco (dischetto). Il nostro esempio indica 52 settori (blocchi per traccia) e due tracce riservate al sistema. Tenete presente che se avete un disco grande utilizzato da molte unità a disco logiche, il numero di tracce riservato sarà grande, perché il sistema usa queste tracce per stabilire quali aree del disco possono essere utilizzate da ciascuna unità a dischi logica.

Per stampare le informazioni sulle aree utente, usate la forma 'STAT USR: ↵'. Il seguente è un esempio di stampa:

```
A > STAT USR ↵
Active User: 0
Active Files: 0 1 3 1 0
```

Il termine 'Active User' in questo esempio è effettivamente il numero dell'area utente in cui siete correntemente "in questo momento". Nell'MP/M, l'area utente corrente è stampata insieme al prompt di sistema (ad es., '0A>' indica l'unità corrente A e l'area utente 0); nel CP/M versione 2.2, dovete determinare l'area utente usando questa forma di STAT. Nel nostro esempio, l'area utente corrente è 0 (zero) (ed è l'area utente in cui si è automaticamente dopo una partenza a freddo o il rilancio del sistema). Il termine 'Active Files' indica il numero delle aree utenti che contengono attualmente dei files (nell'unità a dischi o dischetti corrente). Se vi spostate su una di queste aree utente (usando il comando USER), troverete almeno un file nell'area utente. Aree utente senza files non sono indicate nella riga 'Active files'. Le aree utente (e il comando USER) sono descritte più precisamente in questo capitolo nella sezione sul CP/M versione 2.2 e sull'MP/M.

L'ESECUZIONE DI UN FILE DI COMANDI (SUBMIT E XSUB)

Introduzione

Talvolta è utile o necessario eseguire una sequenza di comandi CP/M come se fossero istruzioni in un programma. Per esempio, se una sequen-

za è usata frequentemente da un operatore, sarebbe conveniente dare un nome alla sequenza ed eseguirla con un singolo comando, proprio come un programma.

Il CP/M fornisce il comando transitorio SUBMIT per eseguire convenientemente una sequenza di più comandi. Il comando SUBMIT si aspetta di trovare file con l'estensione '.SUB' che contenga linee di comandi effettivi che comprendono *argomenti* da sostituire con *valori* al momento dell'esecuzione. Il file '.SUB' viene creato proprio come un file di testo, usando ED o qualsiasi altro programma editor, e contiene linee di comandi come se fossero digitate al terminale. Per esempio, SAMPLE.SUB potrebbe contenere queste linee:

```
DIR $1:$2 ↵
PIP A: = 1$:2$ ↵
```

'$1' e '$2' sono argomenti. Sono come variabili in un programma - saranno sostituiti da valori effettivi quando li fornite al momento dell'esecuzione (cioé, quando usate il comando SUBMIT). Qui, l'argomento '$1' è sostituito dalla lettera che indica l'unità a dischi, e l'argomento '$2' è sostituito da un nome completo di file (compresa l'estensione, se esiste).

In questo esempio, il file SAMPLE.SUB viene eseguito usando il seguente comando SUBMIT:

```
A > SUBMIT SAMPLE B FILE1.TXT ↵
```

Il programma SUBMIT prima cerca SAMPLE.SUB e poi incomincia ad eseguire i comandi. Per eseguire DIR, sostituisce il valore 'B' al posto di '$1' e 'FILE1.TXT' al posto di '$2' e stampa il nome del file nella direttrice dell'unità B. Poi esegue PIP per copiare B:FILE1.TXT sul disco nell'unità A, usando lo stesso nome.

Il comando SUBMIT assume la forma seguente:

```
SUBMIT nomefile v1 v2 v3...
```

dove *v1* è il valore da sostituire al posto di '$1' ovunque nel file '.SUB' e *v2* sostituisce '$2' ovunque nel file '.SUB', ecc. SUBMIT applica l'estensione '.SUB' al nome di file fornito, pertanto non dovete specificare l'estensione '.SUB'.

Quando usate argomenti ($1, $2, ...), dovete usare un simbolo di dollaro seguito da un numero intero. Il numero deve incominciare da 1

per il primo argomento, 2 per l'argomento successivo, 3 per il successivo e così via. Poiché usate il simbolo del dollaro per indicare gli argomenti, dovete usare due simboli di dollaro ($$) per avere un normale simbolo di dollaro in un file '.SUB' (due simboli di dollaro diventano uno in un file '.SUB' mentre un simbolo di dollaro con un numero diventa un *argomento*). Potete anche inserire il carattere ↑ per indicare la combinazione CTRL-tasto (per esmpio ↑ C o ↑ Z) in un file '.SUB' - usate la freccia verso l'alto invece di CTRL (Control). Questo è necessario perché nella maggior parte dei casi non potete usare alcuna sequenza CTRL-tasto in un programma editor mentre costruite il file 'SUB'.
NOTA: indipendentemente dall'unità che scegliete per eseguire l'operazione SUBMIT, il file '.SUB' non sarà trattato fino a quando non inserite il dischetto nell'unità A (o usate l'unità logica A) e rilanciate (partenza a caldo) il sistema (un ↑ C è sufficiente). Potete specificare un'altra unità per il file '.SUB' in un comando SUBMIT mettendo davanti al nome del file l'indicatore dell'unità, ma l'operazione non avrà luogo fino a quando il dischetto con il file '.SUB' non sarà nell'unità A.

Potete abortire un'operazione SUBMIT in esecuzione premendo il tasto RUBOUT (DELETE). SUBMIT crea automaticamente un file temporaneo '$$$.SUB' per contenere i comandi del file '.SUB'; questo file viene cancellato quando SUBMIT è terminato, o se il sistema trova un errore mentre è in funzione l'operazione SUBMIT, o se voi abortite l'operazione SUBMIT premendo RUBOUT (DELETE). Se, per qualche ragione, questo file $$$.SUB esiste ancora dopo che si è verificato un errore, rilanciando il sistema (partenza a caldo) verranno eseguiti i comandi in $$$.SUB (invece di attendere i comandi che voi digitate). Se questo accade, abortite l'operazione SUBMIT (premendo ROBOUT) e cancellate il file $$$.SUB.
NOTA: per i programmatori che scrivono programmi facendo uso di SUBMIT: se il programma assume la funzione CCP (Console Command Processor) di leggere e interpretare i caratteri che provengono dalla console e gli errori di sistema, deve anche cancellare il file $$$.SUB creato da SUBMIT. Inoltre, se eseguite il programma tramite un'operazione SUBMIT, dovete assicurarvi che il file $$$.SUB o viene cancellato (se non volete continuare l'operazione di SUBMIT) o conservato per l'uso futuro (cambiandogli il nome). Ogni nuovo comando SUBMIT sostituirà il file $$$.SUB esistente. Potete naturalmente inserire il comando SUBMIT in un file '.SUB' proprio come "un altro comando CP/M". Questo vi permette di "concatenare" un altro insieme di comandi.

SUBMIT con XSUB

Nel CP/M versione 1.4, SUBMIT crea un file temporaneo $$$.SUB

dal file '.SUB' che voi fornite, e poi il CCP (Console Command Processor, gestore dei comandi di console) esegue ciascuna linea di $$$.SUB come se fosse un comando digitato. (Il CCP è parte del sistema che legge ed esegue ciò che digitate come comando).

Nel CP/M versione 2.2 c'é una possibilità in più: il programma XSUB (comando transitorio). Il programma XSUB permette di inserire dati di input (cioé comandi) in programmi (diversi da CCP) che fanno uso dell'operazione "input bufferizzata" del CP/M. Non occorre che sappiate come usare l'input bufferizzato; dovete solo sapare se il programma che volete eseguire e a cui volete fornire dati di input, usa l'input bufferizzato. I programmi ED, DDT e PIP sono di questo tipo. Pertanto, per esempio, potete usare comandi di ED in un file '.SUB' come input al programma ED e automaticamente realizzare una complessa operazione ED ripetitiva con un comando SUBMIT.

Ecco un esempio di una tale operazione. Il file DOTHIS.SUB contiene le righe seguenti, compresa la 'XSUB':

```
XSUB                        Esegue XSUB
DIR $1.*                    Stampa i files
ED $1.$2                    Usa ED sul file specificato
#A                          Riempie il buffer
B                           Va all'inizio del buffer
I Copyright 1980, Sybex     Inserisce l'indicazione
                            di copyright
E                           Termina ED
PIP 1$.OLD = 1$.BAK         Fa una copia del vecchio file
                            quando il controllo torna
                            al CP/M dopo la fine di ED
DIR $1.*                    Stampa di nuovo tutti i files
A > SUBMIT DOTHIS SAMPLE TXT ↵
```

In questo esempio, SUBMIT dapprima dice a CCP di eseguire XSUB. Il programma XSUB si colloca nell'area direttamente sotto il CCP e resta attivo fino alla prossima partenza a freddo (rilancio del sistema o

"bootstrap" a freddo). Mentre XSUB è attivo, stampa il messaggio 'XSUB ACTIVE' sopra al prompt di sistema; e fino a quando vi sono comandi nel file DOTHIS.SUB, XSUB li esegue (a meno che non intervenga un ↑ C).

Quindi, XSUB esegue il comando DIR per stampare i files associati a SAMPLE. 'SAMPLE' viene sostituito a '1$' e 'TXT' viene sostituito a '2$' in DOTHIS.SUB. XSUB esegue il programma ED sul file SAMPLE.TXT. XSUB continua a fornire input al programma ED (usando l'input bufferizzato di console): dapprima il comando #A per dire a ED di trasferire l'intero file (65535 righe) nel buffer di edit, poi il comando B per spostare il puntatore di carattere di ED all'inizio del buffer, quindi il comando I per inserire il testo "Copyright 1980, Sybex" all'inizio del buffer e infine il comando E per salvare il risultato e terminare il programma ED. (I comandi di ED sono descritti nel Capitolo 4).

Il programma XSUB resta attivo dopo la fine di ED, per eseguire il comando PIP che copia SAMPLE.BAK nel nuovo file SAMPLE.OLD (facendo una copia di backup di SAMPLE.BAK). XSUB poi esegue il comando DIR per stampare tutti i files associati con il nome SAMPLE.

Quando il file di comandi è vuoto, il CCP subentra e attende nuovi comandi digitati al terminale. Comunque, il programma XSUB resta attivo (il messaggio 'XSUB ACTIVE' appare ancora dopo ciascuna esecuzione di un comando) a meno che venga eseguito un programma che intenzionalmente ricopra l'area occupata da XSUB (cioé, direttamente sotto il CCP, o finché venga fatta una partenza a freddo o rilancio del sistema). Mentre XSUB è attivo, il comando SUBMIT può essere usato per eseguire altri files '.SUB'. *Non occorre* che gli altri files '.SUB' abbiano 'XSUB' *senza* XSUB per manterli compatibili con le precedenti versioni del CP/M ed eseguire XSUB come comando quando volete che sia attivo nel momento in cui lanciate '.SUB'.

ASSEMBLAGGIO (ASM), CARICAMENTO (LOAD) E DUMP

Introduzione

Nel mondo del linguaggio assembler, ASM, LOAD e DUMP sono termini usuali per operazioni che permettono ai programmi scritti in linguaggio assembler di funzionare come comandi. Potete usare i comandi ASM e LOAD per trasformare un programma sorgente in linguaggio assembler (scritto in un file di testo) in un comando transitorio realizzato da voi e potete usare DUMP per stampare il contenuto del comando transitorio. Il termine "dump" è usato spesso per descrivere

operazioni di copiatura su larga scala (ad esempio, l'operazione di "copiatura giornaliera" o "daily dump" sui piccoli e sui grandi calcolatori) e l'azione di mandare il contenuto di un programma alla stampante. Il comando DUMP del CP/M "scarica" soltanto l'area di memoria di un programma (in esadecimale) sul terminale video. Per mandare un file alla stampante o per fare operazioni di copiatura su larga scala, dovete usare il comando transitorio PIP.

Per scrivere programmi in linguaggio assembler per un sistema CP/M (o MP/M), dovete conoscere il linguaggio assembler del calcolatore. Il CP/M standard funziona sui microcalcolatori che usano i microprocesori Intel 8080, 8085 o 8086 oppure Zilog Z80 o Z8000. Altri venditori di software offrono il CP/M su calcolatori basati sul 6502 come l'Apple e il Pet.

Assemblaggio

Il comando ASM esegue l'Assembler 8080 della Digital Research, residente nel file ASM.COM. Questo programma traduce un file sorgente in linguaggio "assembler" (scritto nel linguaggio assembler dell'8080) in un file in *linguaggio macchina*. (Vedere Fig. 2.7). Un file in linguaggio macchina contiene istruzioni in *bianario* - il linguaggio del microprocessore; comunque, la maggior parte delle stampe del file saranno in notazione *esadecimale.*

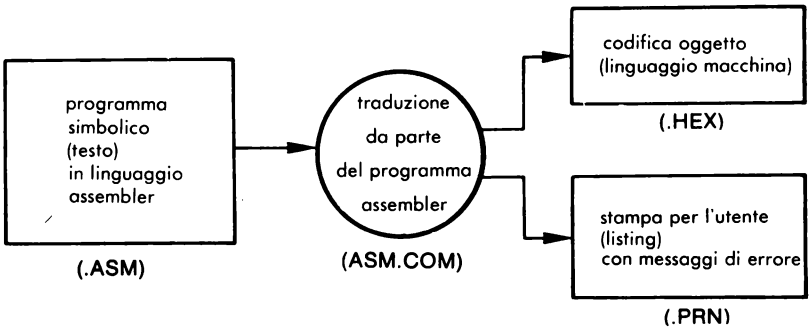


Figura 2.7: L'operazione di assemblaggio

Ci sono altri programmi assembler che possono fornire linguaggi assembler per altri microprocessori, e ci sono assembler più potenti anche per l'8080. Questi assembler potrebbero esistere tutti come files '.COM' nel vostro sistema e voi potete eseguirne uno qualsiasi nello stesso modo in cui eseguite ASM.COM.

Il programma assembler traduce un file sorgente che ha un'estensione '.ASM' (ad esempio, PROG.ASM). L'assembler usa poi l'estensione '.HEX' per indicare il file tradotto in linguaggio macchina (es. PROG.HEX). L'assembler usa anche un'estensione 'PRN' (es. PROG.PRN) per creare un file stampabile. Potete mandare questo file 'PRN' alla stampante per ottenere la stampa del programma. Il file '.PRN' contiene le righe del file sorgente '.ASM', insieme con i messaggi di errore e il codice macchina risultante (nella notazione esadecimale standard Intel).

Il file in codice macchina. ".HEX" prodotto da ASM è ora pronto per essere caricato nel sistema (usando il comando LOAD) e diventa un programma transitorio che viene eseguito come un comando.

Per eseguire ASM, potete usare una di queste forme:

1. ASM nomefile
2. ASM nomefile *.shp*

In entrambe le forme, dovete soltanto specificare il nome primario del file sorgente. Il file deve avere un'estensione '.ASM' - il programma ASM si aspetta di trovare un'estensione '.ASM' per ogni nome di file specificato. ASM non tradurrà un file che non abbia un'estensione '.ASM'.

La prima forma (1.) semplicemente esegue il programma ASM per tradurre il files indicato da "nomefile". ASM assume che il file sia sull'unità corrente e che deve mettere i files '.HEX' e '.PRN' su quell'unità. Per esempio:

```
A > ASM PROG ↵
```

Questo comando esegue ASM (che si ritiene sia sull'unità A come ASM.COM) per tradurre il file PROG.ASM (che si ritiene sia anch'esso sull'unità A). ASM produrrà i files PROG.HEX e PROG.PRN e li metterà sull'unità A.

La seconda forma (2.) vi permette di specificare unità a dischi diverse da quella corrente se il file sorgente è su un'altra unità, se volete mettere i files '.HEX' o 'PRN' su un'altra unità o se volete dire ad ASM di saltare l'operazione e di creare il file '.HEX' o 'PRN'.

In questo formato, *.shp* consiste di tre lettere precedute da un punto che seguono il nome del file. La *s* può essere una qualsiasi delle lettere da A a P. La *h* può essere una delle lettere da A a P, oppure Z; la *p* può essere una delle lettere da A a P, oppure X oppure Z. La *s* è una lettera che indica quale unità (A, B, ..., Y) contiene il disco con il file sorgente. La *h* è una lettera che indica quale unità (A, B, ...Y) dovrebbe ricevere il file

'.HEX' o, se si usa la lettera 'Z', che dice a ASM di non creare il file '.HEX' e generare soltanto il file 'PRN'. La *p* è una lettera che indica quale unità (A, B, ..., W, Y) dovrebbe ricevere il file '.PRN'. Se si usa una 'X' il file. '.PRN' viene soltanto visualizzato sul terminale o, se si usa una 'Z', ASM non crea il file '.PRN' ma soltanto il file '.HEX'.

Ecco alcuni esempi della seconda forma:

```
A > ASM PROG1.ABZ ↵
```

Questo comando assembla il file PROG1.ASM, che è il file sorgente sull'unità A. Crea anche PROG1.HEX sull'unità B (e non crea PROG1.PRN).

```
A > ASM PROG2.BZX ↵
```

Questo comando assembla il file PROG2.ASM, che è il file sorgente sull'unità B e manda il file PROG2.PRN sullo schermo del terminale (senza creare PROG2.HEX).

NOTA: i messaggi di errore di ASM possono prendere la forma di una riga sorgente del linguaggio assembler, in cui una lettera indica un codice di errore. (Questi codici sono spiegati nella documentazione dell'assembler). Gli errori si correggono modificando il programma sotto il debugger DDT (o un altro debugger) o correggendo il file sorgente e riassemblando. Questi errori ASM sono anche indicati nel file '.PRN'. Altri errori che possono verificarsi sono mostrati in Fig. 2.8.

Caricamento

Quando avete un file '.HEX' prodotto da ASM (o un altro assembler) potete trasformarlo in un file '.COM' (un comando transitorio eseguibile) "caricandolo" nel sistema per mezzo di LOAD (vedi Fig. 2.9).

Il comando LOAD è esso stesso un comando transitorio, con il nome LOAD.COM. Lo si esegue digitando 'LOAD' seguito dal nome di file di un file di tipo '.HEX', come nell'esempio seguente:

```
A > LOAD PROG ↵
```

Questo comando cerca PROG.HEX sull'unità corrente, crea un'immagine di memoria di questo file (nel formato esadecimale) e chiama il nuovo file PROG.COM. Questo nuovo programma può essere eseguito digitando 'PROG ↵'.

NO SOURCE FILE PRESENT	ASM non trova il file sorgente o avete specificato un file che non ha l'estensione '.ASM'.
NO DIRECTORY SPACE	La direttrice del disco (dischetto) è piena. Cancellate i files non essenziali (o fatene delle copie) e riprovate.
SOURCE FILE NAME ERROR	Nome di file digitato in modo improprio nel comando ASM. Non potete usare le maschere dei nomi di file nei comandi ASM.
SOURCE FILE READ ERROR	Il file sorgente non può essere letto da ASM per qualche ragione. Usate il comando TYPE per trovare la riga scorretta nel file sorgente.
OUTPUT FILE WRITE ERROR	Un file di uscita ('.HEX' o '.PRN') non può essere scritto sul disco (dischetto); di solito perché il disco (dischetto) è pieno.
CANNOT CLOSE FILE	Un file di uscita. ('.HEX' o '.PRN') non può essere chiuso (aggiornato); verificate se il disco (dischetto) è protetto contro la scrittura (read only).

Figura 2.8: Errori dell'Assembler

Poiché LOAD crea un file '.COM', dovete eseguire LOAD su un file '.HEX' soltanto una volta. Il file '.HEX' deve contenere un "formato esadecimale" Intel corretto, che viene creato da ASM o da un qualsiasi assembler simile.

Potete caricare files anche da un'altra unità specificando l'unità come parte del nome del file. Per esempio:

```
A > LOAD B:GAMES ↵
```

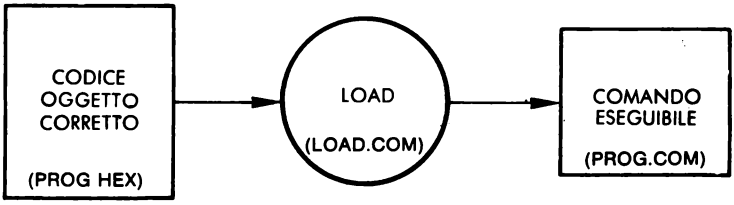


Figura 2.9: Caricamento del Codice Oggetto

Questo comando carica il file GAMES.HEX nell'unità B e crea GAMES.COM sull'unità A (l'unità corrente).

Poiché potete usare PIP per trasferire files in "formato esadecimale" da un dispositivo di lettura, potete anche caricare un file che era stato assemblato qualche tempo prima e che era stato recentemente trasferito sul vostro sistema per mezzo di PIP.

NOTA: per i buffers del linguaggio assembler: il file '.HEX' deve avere records esadecimali validi che incominciano a 100H ('H' significa esadecimale cioé l'indirizzo 100_{16}). 100H è l'inizio della TPA (Transient Program Area, cioé area dei programi transitori). Gli indirizzi del file '.HEX' devono essere in ordine ascendente (i vuoti di memoria vengono riempiti con zero da LOAD quando legge i records esadecimali). Pertanto, LOAD si usa soltanto per creare i files '.COM' che si eseguono in TPA. I programmi che *non* occuperanno la TPA (programmi speciali) possono essere caricati usando DDT (o un altro debugger).

Dump

Il comando DUMP visualizza il contenuto di un file sul terminale in forma esadecimale. Ogni file può essere visualizzato con DUMP, poiché tutti i dati sono valori che possono anche essere rappresentati nella forma esadecimale (base 16). Tutti i numeri esadecimali terminano con 'H' in questo libro, come nei mnemonici Intel. La 'H' sostituisce l'indice 16 (ad esempio, 10H è 10_{16}).

In un comando DUMP dovete dare l'intero nome di file (compresa l'estensione):

```
A > DUMP B:PROG.HEX ↵
```

L'esempio riportato sopra visualizza il contenuto di PROG. HEX (che è sull'unità B).

ESECUZIONE, CORREZIONE (DDT) E SALVATAGGIO (SAVE) DEI PROGRAMMI

Esecuzione

Una volta che un programma '.HEX' è caricato con LOAD, che crea un file '.COM', potete eseguire il programma digitando il nome del file come un comando (senza l'estensione '.COM'). Per esempio:

```
A > PROG ↵
```

Si assume che PROG.COM sia sull'unità A.

Potete eseguire il programma anche su un'altra unità:

```
A > B: ↵

B > A:PROG ↵
```

Quando eseguite un programma o un comando transitorio, questo viene portato nella memoria principale del calcolatore (memoria "scratch-pad" o TPA).

Correzione (debugging)

I "Bugs" (cimici) sono errori in un programma. Gli errori indicati da ASM (o un altro assembler), PIP o LOAD o gli errori "run-time" (errori in fase di esecuzione) possono essere corretti, di solito, usando un programma "debugging" come DDT. DDT.COM viene fornito con la versione standard del CP/M (o MP/M). Il DDT porta un qualsiasi file nella memoria principale ed esegue le operazioni disponibili come comandi DDT. DDT può essere usato per correggere gli errori o per portare un file nella memoria principale al fine di salvarne un'immagine di memoria (usando SAVE). Per eseguire DDT su un file, usate la forma seguente:

```
        ⎧ nomefile.HEX  ⎫
DDT     ⎨               ⎬
        ⎩ nomefile.COM  ⎭
```

Potete specificare una lettera di unità come parte del nome del file (se il file è su un'altra unità).

In entrambi i casi, DDT sostituisce CCP (Esecutore dei comandi di Console) come "sistema operativo" che legge il comando linea per linea. Il CCP è descritto nel Capitolo 5. Quando il programma DDT sostituisce il sistema operativo CCP, il programma DDT assume il compito di leggere la riga di comando. (Come il programma ED, DDT ha il suo proprio insieme di comandi).

Se specificate un nome di file (del tipo '.HEX' o '.COM') con DDT, il debugger porta il programma nella TPA (cancellando qualsiasi cosa ci fosse prima). Se non specificate un nome di file, DDT occuperà la TPA e attenderà un comando DDT 'I' per mettere un file nella TPA.

Quando viene eseguito, DDT stampa un messaggio di conferma (che può essere differente per ciascuna installazione) e poi il suo prompt: '-'. Ora potete digitare i comandi DDT (usate la documentazione fornita con DDT). Nelle ultime versioni del CP/M, DDT stampa anche i valori 'NEXT' e 'PC' che sono trattati con il comando DDT 'R' avanti.

Qui saranno trattati molti comandi DDT: il comando I (input), il comando R (lettura) e il comando GO (terminazione del debugger e ritorno al sistema operativo).

Il comando I inserisce un file '.HEX' o '.COM' nella TPA. Essenzialmente è la stessa operazione che viene realizzata quando fornite un nome di file di tipo '.HEX' o '.COM' con il comando DDT. Potete anche aggiungere un altro file che già esiste nella TPA usando questo comando.

Il comando R legge il file nella TPA e stampa un "messaggio di caricamento" che consiste di:

```
NEXT PC
nnnH pc
```

Questa stampa avviene automaticamente nelle ultime versioni del DDT. Il numero sotto la colonna NEXT è il prossimo indirizzo che esegue il programma caricato. Usate questo valore per calcolatore il numero di "pagine" (blocchi di 256 bytes) da usare in un comando SAVE.

Il comando GO termina DDT, ma lascia il programma nella TPA, cosicché potete usare il comando SAVE per salvare un'immagine di memoria del file. Digitate:

```
GO↵
```

Salvataggio

Il comando SAVE prende una o più "pagine" della TPA (memoria principale) e le mette su un disco come file con il nome che specificate. (Si tenga presente che una "pagina" corrisponde a 256 bytes). (Nell'MP/M questa operazione è realizzata *all'interno* del programma debugger-DDT o SID).

Potete usare il comando SAVE per creare un file che sia l'immagine di memoria di qualsiasi cosa vi è in quel momento nel TPA (il programma eseguito per ultimo). Se usate il DDT su un programma e questo incomincia a funzionare correttamente (o anche se *parte* di esso funziona correttamente) vorrete salvarlo con SAVE in un file che possa essere copiato, corretto, o eseguito.

NOTA: nella versione 1.4 non potete realizzare due operazioni consecutive di SAVE sullo stesso contenuto della TPA, perché il primo SAVE causa un'operazione sulla direttrice che cambia alcune aree dalla TPA. Nella versione 2.2, invece, questo problema è stato corretto - potete realizzare due operazioni consecutive di SAVE sullo stesso contenuto della TPA.

Il comando SAVE prende questa forma

```
SAVE p nomefile
```

Fornite il numero di pagine di memoria in p (un numero decimale). p viene calcolato usando DDT e il comando R del DDT per conoscere l'indirizzo sotto la colonna NEXT - il primo indirizzo (il più alto) che segue il programma nella TPA. Questo indirizzo è effettivamente l'ultimo indirizzo della TPA più *uno*, pertanto, se sottraete uno da questo valore stampato (operando in aritmetica esadecimale) avrete l'ultimo indirizzo della TPA (la fine del programma) che potete trasformare nel numero di pagine in decimale.

Un metodo semplice per trasformare l'indirizzo NEXT nel valore di p è: se l'indirizzo NEXT termina con due zeri (ad esempio 1200H) sottraete 1H ('H' significa esadecimale) ottenendo 11FFH, poi ignorate le ultime due cifre ('FF') e convertite 11H in decimale ($1 \times 16^0 = 1$, e $1 \times 16 = 16$, quindi $11H = 17_{10}$). Se l'indirizzo NEXT *non termina con due zeri, non* sottraetegli 1H; semplicemente convertite le prime due cifre (ad esempio 1205H diventa 12H, che corrisponde a 18 pagine).

Ecco un semplice esempio:

(Valore sotto NEXT nella stampa del comando R del DDT):

```
3FFH
```

NOTA: 'F' è il valore decimale '15' in notazione esadecimale. 'H' significa esadecimale e *non* è una cifra.

Per trasformarlo nel numero di *pagine* (blocchi di 256 bytes), ignorate le prime due cifre 'FF' e usate la cifra '3'. Questo è il "byte di ordine più alto", e può essere espresso come 3H. 3H convertito in decimale è 3_{10}. Pertanto usate 3 come valore decimale delle pagine nel comando SAVE.

Ecco un esempio più difficile:

(Valore sotto NEXT nella stampa del comando R del DDT):

```
1D00H
  -1H
------
1CFFH
```

NOTA: D è il valore decimale '13'. Sottraete 1H perché '1D00H' termina con due zeri. D (13) diventa C (12).

Per trasformare nel numero di pagine, usate il byte di ordine più alto '1C' e convertitelo in decimale:

$$\begin{aligned} CH &= 12 \text{ volte } 16^0 = 12 \\ +\ 10H &= 1 \text{ volta } 16^1 = 16 \\ &= \ldots\ldots\ldots\ldots\ldots\ldots 28 \end{aligned}$$

NOTA: il valore decimale 28 corrisponde al numero di pagine di memoria tra gli indirizzi 100H (inizio della TPA) e 1CFFH.

Ecco alcuni esempi di comandi SAVE:

A SAVE 4 PROG. COM ↵	Salva la memoria da 100H a 4FFH e la mette in PROG. COM sull'unità A. Il primo indirizzo dopo 4FFH è 500H.
B SAVE 10 KLUDGE.COM ↵	Salva la memoria da 100H a 0AFFH nel file KLUDGE. COM sull'unità B. Il primo indirizzo dopo 0AFFH è 0B00H. AH corrisponde a 10 decimale.
A SAVE 40 B: WORKS.COM ↵	Salva la memoria da 100H a 28FFH nel file WORKS.COM sull'unità B. Il primo indirizzo dopo 28FFH è 2900H. 28H è (2 volte 16)+8 = 40 decimale.

CP/M VERSIONE 2.2 E MP/M

Un'Introduzione all'MP/M

L'MP/M è un sistema operativo progettato per il time - sharing (funzionamento dei programmi a suddivisione di tempo). Un sistema time-sharing può eseguire molti *processi* o programmi contemporaneamente. Comunque, la simultaneità è soltanto apparente. In effetti, il calcolatore esegue un programma utente, e poi un altro in rapida successione cosicché ciascun utente ha l'impressione di essere solo a usare il calcolatore.

Pertanto, quando l'MP/M viene usato da un solo terminale, il sistema essenzialmente si comporta come un sistema CP/M a utente singolo. L'MP/M comunque offre delle possibilità addizionali permettendo all'utente singolo di eseguire molti programmi contemporaneamente. Per esempio, i programmi possono essere agganciati alla console e sganciati da essa, permettendo all'utente singolo di interagire con la console mentre altri programmi sono in esecuzione.
NOTA: se prevedete di usare l'MP/M come utente singolo nella configurazione di programma singolo, non avete bisogno di conoscere l'MP/M. La lettura del testo sul CP/M versione 2.2 coprirà le vostre esigenze.

Uno dei problemi che un sistema time-sharing deve risolvere è la *gestione dei programmi* (*scheduling*). Si deve fare in modo che ciascun programma funzioni a turno sul processore. La tecnica di scheduling *round-robin* è la più semplice tra quelle usate. Con questa tecnica, ciascun programma ottiene un'uguale porzione del tempo del calcolatore a turno. Questa tecnica è illustrata da un puntatore ruotante nella Fig. 2.10.

Per rendere efficiente l'uso del calcolatore, comunque, alcuni programmi (o processi) dovrebbero funzionare prima di altri. In ogni buon sistema di scheduling, ciascun processo dovrebbe essere fornito di un

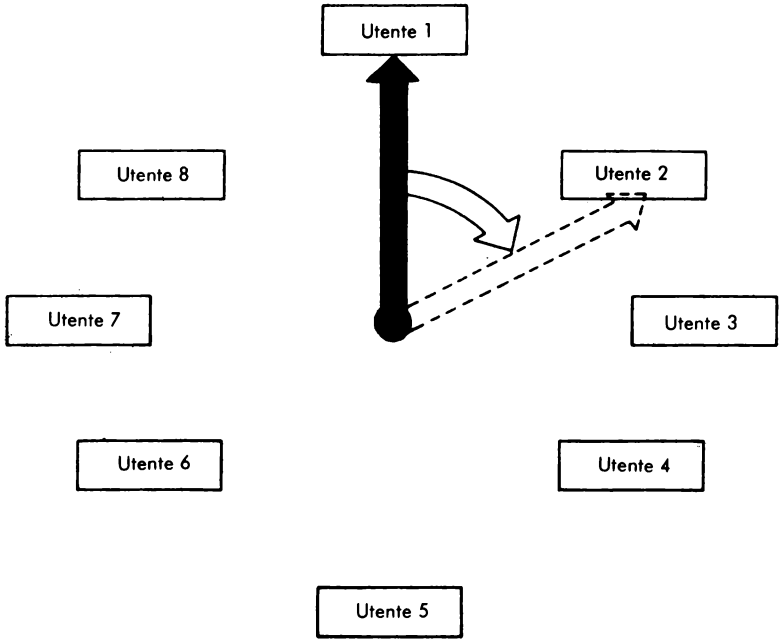


Figura 2.10: Scheduling Round Robin

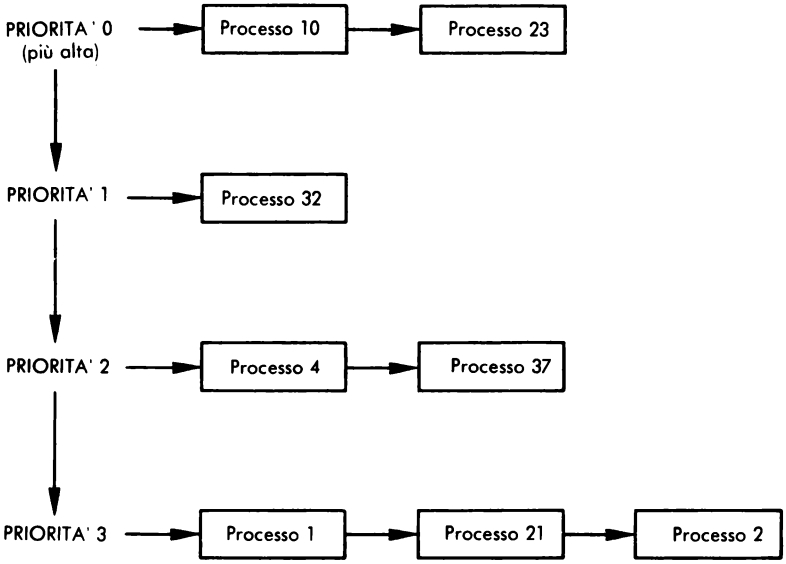


Figura 2.11: Una lista di priorità a 4 livelli

livello di priorità che determina quando dovrà funzionare. Per esempio, quando dei dati diventano disponibili sul disco, dovrebbero essere letti prontamente da un programma di trasferimento, altrimenti un lungo ritardo può provocare il movimento della testina dopo il punto corretto sul disco. Questo processo di trasferimento dovrebbe avere una priorità elevata. Al contrario, un processo che stampa su una stampante lenta può essere ritardato di alcuni millisecondi o anche secondi, senza una grande perdita. Pertanto, il processo di stampa dovrebbe in generale avere una priorità bassa.

Per fissare delle priorità, solitamente si distinguono due classi di processi: quelli che realizzano funzioni di input/output (operazioni di I/O), e quelli che realizzano calcoli (operazioni di CPU).

Mentre un processo è in esecuzione sulla CPU (il processore), può richiedere un'operazione di input o output. Questo tipo di operazione è molto lenta rispetto agli standard della CPU, poiché tipicamente richiede alcuni millisecondi, e un'istruzione della CPU viene eseguita in microsecondi. Quando viene richiesta un'operazione di I/O, il processo in esecuzione generalmente viene *bloccato* e diventa *sospeso*. Quindi viene iniziata un'operazione di I/O. L'I/O procederà contemporaneamente alla CPU. Quando l'operazione di I/O sarà completata, risveglierà il processo originale sospeso. Il processo di CPU allora riassume la sua posizione nella lista di scheduling. È diventato di nuovo *attivo*.

Una lista di priorità di processi a più livelli è mostrata in figure 2.11.

Poiché la priorità più alta usualmente viene indicata con 0, la priorità effettiva diminuisce al crescere del numero della priorità.

Nella Fig. 2.11, prima viene eseguito il processo 10 poi il processo 23.

Mentre il processo 23 è in esecuzione, un nuovo processo potrebbe entrare alla priorità 0 (vedere Fig. 2.12). In questo caso, il prossimo processo a entrare in esecuzione sarà il 40. Quando tutti i processi a livello 0 avranno completato l'esecuzione, verranno eseguiti quelli al livello 1 e così via.

Non è stata data qui una definizione formale di processo perché ciascun sistema operativo usa una definizione in qualche modo differente. Un *processo* può essere un programma o un sottoprogramma o un'altra porzione di programma. Così, un utente può attivare una molteplicità di processi.

Nell'esempio in Fig. 2.11, sono mostrati soltanto i *processi attivi*. Una lista simile viene mantenuta per i *processi sospesi* che sono stati bloccati. Quando viene risvegliato, un processo sospeso diventa attivo ed entra nella lista dei processi attivi.

Il sistema di scheduling che è stato descritto viene generalmente usato per il processo. Però può esservi competizione per altre scarse risorse all'interno del sistema, come il disco o la stampante.

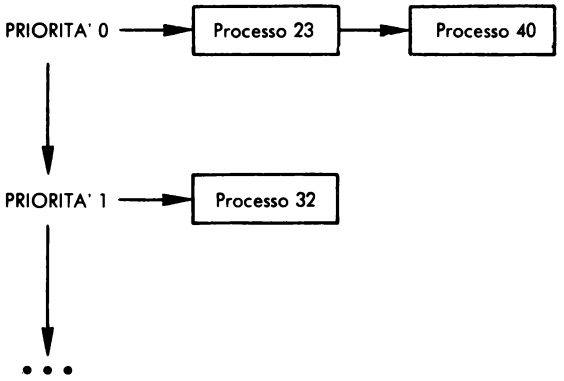


Figura 2.12: Un nuovo processo è entrato alla priorità 0

Generalmente, i processi si accodano per la stampante e sono serviti sulla base del metodo arrivato primo servito. Questo metodo viene chiamato FIFO (First-In First-Out). Il programma di scheduling corrispondente viene chiamato *spooler*.

Similmente vengono di solito accodate le richieste per il disco. Però un

programma cerca di ottimizzare l'accesso al disco leggendo quanti più settori è possibile muovendo la testina il meno possibile.

Il risultato è che i processi in coda di attesa per il disco possono essere serviti in un ordine qualsiasi, se il blocco di cui hanno bisogno viene a trovarsi sotto la testina del disco.

La suddivisione di risorse comuni su processi simultanei richiede meccanismi addizionali:

— I processi devono essere sincronizzati. Questo viene realizzato con segnali (flags) e interruzioni.
— I processi devono operare in un ordine ragionevole quando sono in competizione per un comune dispositivo. Questa è la funzione del programma di scheduling per quel dispositivo.
— Si deve disporre di meccanismi di protezione in modo che il malfunzionamento di un processo non danneggi gli altri.
— Lo spazio di memoria deve essere assegnato in modo efficiente affinché i processi occupino lo spazio di memoria soltanto quando ne hanno bisogno.

L'MP/M è un semplice sistema time-sharing che fornisce un buon insieme di possibilità. Però, per permettere che un sistema time-sharing risieda in una quantità limitata di memoria, si è provveduto a molte risorse in modo semplice:

— L'MP/M esegue fino a otto processi.
— Ciascun processo ha il **suo** proprio segmento di memoria fissato (48K).
— I files possono avere uno di questi quattro attributi:
  — R/O (sola lettura);
  — R/W (lettura/scrittura);
  — SYS (sistema-il file non viene riportato nella direttrice);
  — DIR (direttrice-toglie l'attributo SYS).

Introduzione al CP/M 2.2

La nuova versione del CP/M ha molti miglioramenti opzionali progettati per una transizione semplice all'MP/M, un sistema multiutente. In un ambiente multiutente, dove molte persone usano la medesima macchina nello stesso tempo, occorre più protezione per i vostri files, perché dovete *condividere* le risorse del sistema - la stampante, le unità a disco (o dischetto) e lo stesso calcolatore (CPU). Se avete il CP/M versione 2.2, potete usare molte caratteristiche progettate in effetti per l'MP/M, e conservare la compatibilità, cioé i vostri files e programmi saranno utilizzabili su entrambi i sistemi.

Le nuove caratteristiche sono:

- Area utente e il comando USER per separare i vostri files dai files degli altri utenti.

- Attributi di files (assegnati usando STAT) per progettare i vostri files da cancellazioni accidentali o sovrascritture, compreso un attributo per memorizzare un file in modo segreto.

NOTA: se *non* usate queste caratteristiche, i vostri files e programmi saranno compatibili con le versioni precedenti del CP/M così come con il CP/M versione 2.2 e l'MP/M.

Oltre a queste caratteristiche, l'MP/M ne ha altre che non si trovano nel CP/M versione 2.2 e sono usate *soltanto* negli ambienti multiutente:

— Il comando CONSOLE per conoscere il numero della console corrente (terminale) in un sistema con più di un terminale.
— Il comando DSKRESET per regolare il cambiamento dei dischi in un sistema multiutente.
— Il comando GENMOD per produrre programmi rilocabili (essenziali in sistemi a multiprogrammazione).
— Il comando SPOOL per regolare il traffico verso la stampante (STOPSPRL viene usato per cancellare uno SPOOL).
— Il comando SCHED per fissare l'esecuzione dei programmi in un tempo successivo.
— Il comando TOD per stampare o assegnare l'ora e la data.
— La possibilità di sganciare un programma in esecuzione dal vostro terminale e riagganciarlo più tardi; da usarsi insieme a MPMSTAT per stampare le informazioni sul sistema.

Le caratteristiche dell'MP/M 1.0

L'MP/M richiede almeno 32K bytes di memoria. L'MP/M può gestire fino a 16 console e 16 dischi, e ciascuna unità a disco è in grado di contenere sino a 8 megabytes di informazione. L'MP/M permette fino a 8 segmenti di memoria di 48K.

Area utente e comando USER

Nel CP/M versione 2.2 e nell'MP/M, potete separare i files su un disco (dischetto) in "aree utente" numerate da 0 a 15. I files non sono separati in modo effettivo - si deve fare riferimento ad essi usando un "numero dell'area utente" che viene anteposto internamente al loro nome. Si assume che voi possiate scegliere un'area utente e mettere i vostri files nella stessa area su tutti i dischi (dischetti). La Fig. 2.13 dimostra questa collocazione.

Quando si accede inizialmente all'MP/M, appare su ciascuna console questo messaggio:

	Disco A	Disco B	Disco C
utente 0	files di sistema	files di sistema	files di sistema
utente 1			
utente 2 (voi)	i vostri files	i vostri files	i vostri files
utente 4			

Figura 2.13: Collocazione dei files nelle aree utente

```
MP/M
nA >
```

dove n è il numero di utente per quella console. n è 0 per la console 0, 1 per la console 1 ecc. Per esempio, la stampa sulla console 3 (assumendo che siano connesse 4 o più console) è:

```
MP/M
3A >
```

Ciascun utente nel campo da 0 a 15 può essere assegnato a qualsiasi console usando il comando USER descritto sotto.

La 'A' si riferisce all'unità a dischi A, e può essere cambiata a piacere come nel CP/M.

Come mettere un file nell'area utente

Quando siete "dentro" a una "area utente corrente", i files che create vengono messi dentro a quell'area utente e dovete essere "dentro" a quell'area utente per accedere ai files. Se avete bisogno di "ottenere" una copia di un file che è in un'altra area utente, usate il parametro G di PIP fornito dal CP/M versione 2.2 e nell'MP/M (descritto nel Capitolo 3).

Quando caricate per la prima volta il CP/M versione 2.2 o l'MP/M, la

vostra area utente corrente è l'area utente 0. Nell'MP/M, questo viene indicato dal prompt di sistema 'OA>' (per l'unità A, utente 0). Nel CP/M versione 2.2, potete usare la forma di STAT, 'STAT USR: ↵', per determinare il numero dell'area utente corrente.

Come passare a un'altra area utente

Se siete nell'area utente 0 (cioé, non usate il comando USER), i files e i programmi saranno compatibili con le versioni precedenti del CP/M e con l'area utente 0 dell'MP/M. Se desiderate passare a un'altra area utente, dovete usare il comando USER; i files creati in quell'area utente devono essere copiati nell'area utente 0 (usando il parametro G di PIP) per essere compatibili con le versioni precedenti del CP/M.

Il formato del comando USER è:

```
USER n
```

dove n è un argomento opzionale nell'MP/M ma necessario nel CP/M versione 2.2. In entrami i sistemi, se fornite un n nel campo da 0 a 15, il comando USER vi porterà in quell'area utente. Per esempio,

```
0A > USER 2 ↵
2A >
```

NOTA: questo è un esempio MP/M. Nel CP/M dovete usare 'STAT USR: ↵' per stampare l'area utente corrente. In entrambi i casi, l'utente viene portato nell'area utente 2 della stessa unità a dischi.

Quando vi portate in un'altra area utente, rimanete in quell'area utente fino a quando viene eseguito un altro comando USER o il sistema viene rilanciato (partenza a freddo). Dopo una partenza a freddo, siete sempre portati nell'area utente 0.

Punti da ricordare sulle aree utente

Un'area utente non diventa attiva fino a che non vi spostate su di essa. Si tenga presente invece che l'unità a dischi rimane attiva qualunque sia il punto in cui voi accedete - e le aree utente sono all'interno dell'unità a dischi logica. Le aree utente sono state in realtà progettate pensando ai dischi di grande capacità (possono essere usati i dischi rigidi). L'area utente cessa di esistere quando cancellate *tutti* i files, compresi quelli con gli attributi $R/O e $SYS (trattati più avanti in questo capitolo). Potete cancellare tutti i files usando il comando ERA con la maschera dei nomi

di file '*.*', che cancella tutti i files nell'area utente corrente eccettuati quelli con l'attributo $R/O (solo lettura).

Poiché il comando ERA cancella i files soltanto nell'area utente corrente, potete cancellare i files soltanto in un'area utente per volta, a meno che scriviate un programma che cancella tutto quello che vi è sul disco (dischetto). La forma 'ERA *.* ↵' cancella i files soltanto nell'area utente corrente (quella in cui siete correntemente "dentro"). Non cancella i files con l'attributo $R/O (sola lettura) fino a che non cambiate il loro attributo in $R/W (sola lettura) fino a che non cambiate il loro attributo in $R/W (lettura-scrittura).

Se usate il comando DIR per vedere se esistono ancora dei files nell'area utente correne, potreste non accorgervi dei files con l'attributo $SYS (chiamati "files di sistema"). Questi files non appaiono nella stampa di DIR, ma appaiono soltanto in parentesi in una stampa prodotta da 'STAT *.* $S ↵'. Usate il comando STAT per cambiare gli attributi di un file (vedere più avanti) per cancellare tutti i files e eliminare un'area utente.

Attributi dei files

Normalmente i vostri files sono già files di "direttrice" che possono essere *letti* o *scritti*. Opzionalmente, potete usare in modo che un file non venga elencato in una stampa di DIR sostituendo l'attributo di "direttrice" con un attributo di "sistema". Potete anche progettare un file da un'operazione di sovrascrittura o cancellazione se cambiate il suo attributo di "lettura-scrittura" in "sola lettura".

Nel CP/M versione 2.2 e nell'MP/M, ciascun file viene creato con l'attributo $DIR per indicare che è un "file di direttrice" che il comando DIR può trovare e visualizzare. Ciascun file viene anche creato con l'attributo $R/W (lettura-scrittura) per indicare che può essere *letto* e *scritto* (e *cancellato*).

Quando un file ha l'attributo $SYS (il contrario di $DIR), il comando DIR e il comando PIP non sono in grado di trovarlo. (PIP non può trovarlo a meno che non usiate il parametro R di PIP, descritto nel Capitolo 3). L'attributo $SYS è usato per "nascondere" i files DIR e PIP e così evitare che altri utenti del sistema conoscano il contenuto di un file o lo copino. Si noti comunque che il comando STAT può essere usato da altri utenti per vedere anche i files di "sistema", perciò questa protezione (da PIP e DIR) non è sufficiente. Dovete *anche* assegnare l'attributo $R/O (sola lettura) al file (il contrario di $R/W) e poi nascondere la documentazione del comando STAT. Essenzialmente, questi files sono soltanto protetti contro il *vostro accidentale* uso errato del sistema. Se poi usate in modo sbagliato i vostri dischi, questo è un altro problema.

NOTA: il comando ERA è in grado di trovare files di "sistema", ma non può cancellare i files a sola lettura. Dovete usare STAT per cambiare l'attributo $R/O in $R/W se volete cancellare un file $R/O.

Ecco alcuni esempi dell'uso di STAT per cambiare e visualizzare gli attributi

```
A > STAT SAMPLE.TXT $R/O ↵
```

Questo comando cambia SAMPLE.TXT in un file a sola lettura.

```
A > STAT B:TEMP $R/W ↵
```

Questo comando cambia SAMPLE.BAK in un file a sola lettura e di "sistema", uno di quelli non trovati da DIR:

```
A > STAT SAMPLE.BAK $R/O ↵
A > STAT SAMPLE.BAK $SYS ↵
```

Questo comando cambia il file TEMP sull'unità B in un file a lettura-scrittura (da sola lettura).

```
A> DIR SAMPLE.BAK ↵
NOT FOUND
A > STAT SAMPLE. * $S ↵

 Size    Recs    Bytes   Ext   Acc
  48      48      6K      1    R/O  A:SAMPLE.TXT
  48      48      6K      1    R/W  A:SAMPLE.JNC
  48      48      6K      1    R/O  (A:SAMPLE.BAK)
```

Il file SAMPLE.BAK è in parentesi perché ha l'attributo $SYS oltre all'attributo $R/O (indicato nella colonna 'Acc', che rappresenta il metodo di accesso al file). Gli altri files hanno l'attributo $DIR e SAMPLE.TXT ha l'attributo $R/O (sola lettura) mentre SAMPLE.JNC ha l'attributo $R/W (lettura-scrittura).

Se copiate un file a sola lettura o di "sistema", PIP crea la nuova copia con gli attributi automatici di lettura-scrittura ($R/W) e direttrice ($DIR). Dovete usare STAT per assegnare effettivamente i nuovi attributi.

Operazioni MP/M

In un sistema multiutente MP/M, ciascun utente ha un terminale e può operare sul sistema. Un sistema MP/M con un solo utente può essere identico a un sistema CP/M versione 2.2. Con più di un utente, invece, l'MP/M può apparire come un sistema CP/M separato per ogni utente. In qualche ambiente può essere necessario disporre di un "operatore di sistema" che gestisca tutte le risorse e operi sul sistema per gli utenti. Un operatore di sistema potrebbe compiere le operazioni seguenti:

- Cambio dei dischetti (usando il comando DSKRESET).
- Trasmissione dei files alla stampante o ad altri dispositivi (i comandi SPOOL e STOPSPRL).
- Lo scheduling dei programmi che devono essere eseguiti in un tempo successivo (comando SCHED).
- L'assegnazione dell'ora e della data (comando TOD).
- La stampa delle informazioni riguardanti il sistema (comando MPMSTAT).

DSKRESET (MP/M)

Anche se potete passare su qualsiasi unità a dischetti ed estrarre un dischetto, questo non sarebbe saggio in un sistema MP/M, dove altri utenti potrebbero in quel momento accedere ai files sul dischetto. L'MP/M ha il comando DSKRESET per informare gli altri utenti che qualcuno desidera cambiare un dischetto. DSKRESET, come gli altri comandi MP/M, è sul dischetto di sistema come file '.COM' o '.PRL' ("a pagina rilocabile") e può essere eseguito semplicemente digitando il suo nome primario:

```
0A > DSKRESET ↵
Confirm reset disk system (Y/N) ?Y
```

Il messaggio "Confirm reset disk system" appare su tutti i terminali agganciati al sistema. Ciascun utente può rispondere con una 'Y' (sì) per permettere un reset del disco.

Spooling (MP/M)

Quando più di un utente vuole mandare dei files alla stampante o quando un utente vuole mandare molti files, l'utente può fare uno *spooling* (mettere in fila, uno dopo l'altro, in una *coda*) dei files al dispositivo. Comunque, possono essere mandati soltanto files di testo

(ASCII) o files dati che contengono un testo ASCII. (Si tenga presente che i files sorgente, i listings, i files '.PRN' e ogni file creato da un text editor come ED o un programma di gestione dei testi, sono tutti files di testo).

Per mandare i files alla coda di spooling, usate il comando SPOOL, che prende la forma seguente:

SPOOL nomefile *nomefile nomefile ...*

Il primo nome di file è obbligatorio; gli altri sono opzionali. Il nome di file deve comprendere l'estensione, se esiste. Ecco un esempio:

```
0A > SPOOL PROG.PRN SAMPLE.TXT TEMP.LST ↵
```

Questo comando manda i files PROG.PRN, SAMPLE.TXT, e TEMP.LST al dispositivo LST: (solitamente una stampante).

Per interrompere un'operazione di spooling e vuotarne la coda, usate il comando STOPSPRL. Ecco un esempio:

```
0A > STOPSPLR ↵
```

L'MP/M è in grado di conservare l'ora e di fornire la data corretta se assegnate questi valori. Usando l'ora e la data, potete fare in modo che i programmi vengano eseguiti a un'ora specificata di una data specificata. Il sistema verifica costantemente l'ora e la data per gestire i programmi di questo genere.

Esecuzione a scheduling (MP/M)

Il programma SCHED può essere usato per fissare l'esecuzione di un altro programma. Il programma SCHED ha sia l'estensione '.PRL' che '.COM'. Quando lo eseguite come un comando, dovete fornire gli argomenti indicati nel formato seguente:

SCHED mm/gg/aa hh:mm programma

Fornite la data come mm/gg/aa, il tempo in ore (hh) (espresse da zero a ventiquattro) e minuti (mm) (espressi da zero a sessanta). Si assume che il programma abbia un nome con l'estensione '.COM' o '.PRL' (pagina rilocabile), ma non è necessario fornire l'estensione.

Ecco un esempio:

```
0A > SCHED 12/31/80 23:59 EIGHTY ↵
```

Il programma EIGHTY.COM (o EIGHTY.PRL) sarà eseguito il 31 Dicembre 1980 alle 23 e 59, se il sistema è in funzione e trova quella data. Si tenga presente che poiché l'indicatore di tempo può essere modificato in qualsiasi momento, chiunque potrebbe interferire e cancellare l'ora e la data. Tutti gli utenti pertanto devono cooperare perché si possa fare assegnamento su questa operazione.

Ora del giorno (MP/M)

Per stampare l'ora del giorno usate la forma semplice del comando TOD:

```
0A > TOD ↵
Sat 12/29/80 02:20:14
```

Per riassegnare l'ora e la data, usate questa forma del comando TOD:

```
0A > TOD 12/29/80 02.22.00 ↵
Strike any key to set time
Sat 12/29/80 02:22:00
0A >
```

NOTA: il programma TOD stampa il messaggio 'Strike any key'; potete quindi premere un qualsiasi tasto quando siete pronti.

Arresto di un programma (MP/M)

Per arrestare (o abortire) il programma correntemente agganciato alla console, digitate ↑ C. (Questo non ha effetto sui programmi sganciati).

Il comando ABORT abortisce un programma qualsiasi, anche se appartiene ad un'altra console.

Per esempio:

```
2B > ABORT LISTING 2 ↵
```

abortisce il programma LISTING appartenente alla console alla quale

viene digitato il comando (console 2).

È anche possibile digitarlo alla console 4:

```
4A > ABORT LISTING 2 ↵
```

per abortire il programma LISTING. Si noti che il nome della console può essere opzionalmente specificato dopo il nome del programma.

Agganciamento e sganciamento di un programma (MP/M)

Un programma può essere sganciato dalla console digitando ↑ D. Continuerà a funzionare in modo "invisibile" fino a quando non viene riagganciato alla console. Questo libera la console per l'esecuzione di un altro programma o per inserire un testo o la data.

La digitazione di ↑ D sgancia il programma dalla console, purché il programma verifichi lo stato della console, cioé legga il comando. Un programma può anche sganciarsi automaticamente eseguendo una chiamata di sganciamento XDOS.

Inversamente, un programma viene agganciato alla console con il comando ATTACH. Deve sempre essere riagganciata alla stessa console dalla quale era stato sganciato. Per esempio:

```
0A > ATTACH PROG ↵
```

NOTA: la digitazione di ↑ D quando il TMP (processo del messaggio di terminale) è in esecuzione sulla console, provoca l'attivazione del processo successivo, che sia pronto per essere eseguito e al più alto livello di priorità di quelli in attesa sulla console. Si noti anche che TMP è in esecuzione ogni volta che un comando può essere digitato o è in fase di esecuzione.

Console (MP/M)

Poiché un numero di utente non corrisponde necessariamente al numero della console, viene fornito il comando CONSOLE per esaminare il numero della console che si sta usando. Per esempio:

```
2A > CONSOLE ↵
CONSOLE = 1
```

L'esempio sopra indica che l'utente n. 2 sta usando la console n. 1.

Direttrice (MP/M)

Il comando DIR funziona nel modo solito e ha un'estensione, "l'opzione S". Per esempio:

```
0A > DIR *.* S ↵
```

inserisce nell'elenco tutti i files che hanno l'attributo di sistema.

Cancellazione (MP/M)

È disponibile la forma solita del comando ERA, accanto a una forma nuova. Il comando ERAQ può essere usato per cancellare un insieme di files che corrispondono a una maschera specifica. Per esempio:

```
1B > ERAQ PROG. * ↵
B:PROG COM? 4
B:PROG INT? 4
```

Stampa di un file (MP/M)

Per eseguire la stampa (su console) di un file, è disponibile il solito comando TYPE. Inoltre può essere usata la modalità di pausa. Quando viene utilizzata questa opzione, il comando:

```
0A > TYPE PROG.TXT P15 ↵
```

stamperà 15 righe di PROG.TXT e poi attenderà fino a quando viene digitato un ↵.

Caratteri di controllo dell'MP/M

L'MP/M fornisce le stesse funzioni del CP/M per il controllo della digitazione dei comandi, più alcune funzioni addizionali. I caratteri di controllo sono elencati nel Capitolo 6.

MPMSTAT (MP/M)

L'MP/M mette a disposizione uno speciale comando STAT per stampare in modo opportuno lo stato completo del sistema. Il comando è:

```
0A > MPMSTAT ↵
```

Una stampa tipica è riportata qui di seguito.

```
        ****** MP/M Status Display ******

Top of memory = FFFFH
Number of consoles = 02
Debugger breakpoint restart # = 06
Stack is swapped on BDOS calls
Z80 complementary registers managed by dispatcher
Ready process (es):
  MPMSTAT Idle
Process (es)     DQuing:
  [Sched]        Sched
  [ATTACH]       ATTACH
  [CLiQ]         cli
Process (es)     NQuing:
Delayed Process (es):
Polling Process (es):
  PIP
Process (es) Flag Waiting:
  01 - Tick
  02 - Clock
Flags(s) Set:
  03
Queue(s):
MPMSTAT  Sched      CliQ      ATTACH    MXParse
MXlist   [Tmp0 ]    MXDisk
Process (es) Attached to Console:
  [0] - MPMSTAT
  [1] - PIP
Process (es) Waiting for Consoles:
  [0] - TMP0       DIR
  [1] - TMP1
Memory Allocation:
  Base = 0000H   Size = 4000H   Allocated to PIP    [1]
  Base = 4000H   Size = 2000H   * Free *
  Base = 6000H   Size = 1100H   Allocated to DIR    [0]
```

NOTA: un'interpretazione dettagliata di questa stampa dello stato va oltre lo scopo di questo capitolo. La stampa è inclusa prima di tutto per rendere più completo il testo e può essere saltata durante una prima lettura.

Il significato semplificato della stampa è il seguente:

Ready Process(es): Questa lista riporta tutti i processi pronti in ordine di priorità. Il processo di priorità più alta è quello in esecuzione.

Process(es) DQuing: viene riportata ciascuna coda, con i processi che hanno eseguito un'operazione di lettura sulla coda. I processi sono elencati in ordine di priorità e sono in attesa di un messaggio scritto sulla coda.

Process(es) NQuing: Come sopra, tranne che i processi attendono un buffer per scrivere un messaggio sulla coda.

Delay Process(es): Elenca i processi ritardati per un certo tempo (espresso in quantità elementare di tempo).

Polling Process(es): Elenca i processi che interrogano un dispositivo di I/O in attesa che sia disponibile.

Process(es) Flag Waiting: Elenca i processi in attesa sul flag corrispondente.

Flag(s) Set: Elenca i flags che sono attivi.

Queus(s): Elenca le code nel sistema. I caratteri maiuscoli sono usati per quelle code a cui si può accedere per mezzo di un comando di console. 'MX' all'inizio di un nome di coda indica la mutua esclusione.

Process(es) Attached to Console: Elenca i processi e i corrispondenti numeri di console.

Process(es) Waiting for Console: Elenca i processi per console e priorità. I processi sono stati sganciati e ora sono in attesa della loro console per riprendere l'esecuzione.

Memory Allocation: Presenta una mappa di memoria che indica l'indirizzo base, l'estensione, il banco (se applicabile) e il processo residente insieme al numero di console.

Comandi addizionali dell'MP/M

L'MPM è fornito di tre comandi addizionali, apparentemente complessi, che sono usati soltanto dai programmatori in linguaggio assembler: GENMOD, GENHEX, e PRLCOM.

Sono elencati qui per completare il testo, ma possono essere saltati in una prima lettura.

GENMOD (MP/M)

Questo comando speciale trasforma FILE1 che contiene due files esadecimali concatenati (del tipo HEX), sfasati l'uno rispetto all'altro di 0100H bytes, in FILE2 che è rilocabile per pagine (del tipo PRL).

Il formato del comando è:

```
0A > GENMOD u: FILE1.HEX u:FILE2.PRL $DDDD ↵
```

dove $DDDD è un parametro opzionale che specifica, in esadecimale, l'ammontare addizionale di memoria richiesto dal programma.

GENHEX (MP/M)

Questo comando trasforma un file COM in un file HEX. Questo comando viene dato spesso prima di GENMOD. Si può anche specificare uno sfasamento degli indirizzi (offset). Per esempio:

```
0A > GENHEX B:FILE.COM 200 ↵
```

PRLCOM (MP/M)

Questo comando trasforma un file PRL in un file COM:

```
0A > PRLCOM B: FILE1.PRL A: FILE2.COM ↵
```

GENSYS (MP/M)

Questo comando viene usato per generare un sistema MP/M. Interroga l'utente per tutte le informazioni e i parametri richiesti e crea il file MPM.SYS. Il comando MPMLDR (descritto più avanti) può essere usato successivamente per caricare ed eseguire MPM.SYS.

Il dialogo è mostrato qui di seguito. Una linea indica una risposta dell'utente.

```
A > GENSYS ↵
        MP/M SYSTEM GENERATION

————————————————————————
————————————————————————
Top page of memory = ——
Number of consoles = —
Breakpoint RST # = —
Add system call user stacks (Y/N)? —
Z80CPU (Y/N)? —
Bank switched memory (Y/N)? —
Memory segment bases, (ff terminates list)
: ——
: ——
```

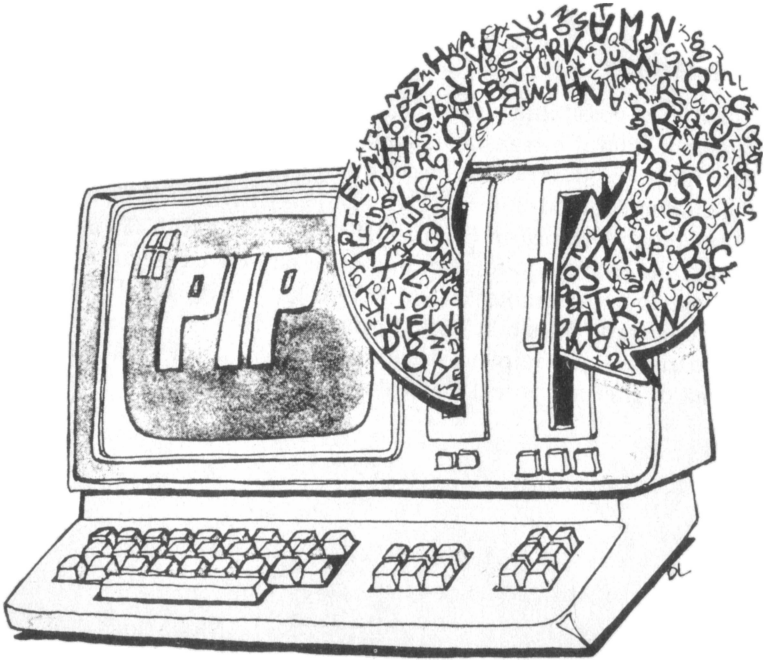
```
: ——
: ——
Select Resident System Processes: (Y/N)
ABORT       ? —
SPOOL       ? —
MPMSTAT     ? —
SCHED       ? —
```

NOTA: il dialogo riportato sopra varia leggermente se viene specificato "Bank Switched Memory".

SOMMARIO

Tutte le possibilità offerte dal CP/M, dai caratteri di controllo ai comandi interni e transitori, sono state presentate in questo capitolo. Ricordate che dovreste conoscere tutti i caratteri di controllo, quattro dei cinque comandi interni (DIR, TYPE, REN ed ERA), e quattro dei nove comandi transitori (PIP, ED, SYSGEN e STAT). Gli altri comandi vi saranno utili soltanto se prevedete di scrivere ed eseguire programmi in linguaggio assembler (ASM, LOAD, DUMP, DDT, SAVE), o se volete usare la possibilità offerta da SUBMIT.

Un riassunto completo di tutti questi comandi sarà presentato in forma di riferimento nel Capitolo 6.


PIP

CAPITOLO 3

GESTIONE DEI FILES CON PIP

INTRODUZIONE

Il comando PIP è stato introdotto molto succintamente nel Capitolo 1, dove è stato usato per copiare un file. L'uso principale di PIP è copiare, o più precisamente, trasferire dei files. Però può fare molto di più.

In questo capitolo imparerete tutte le possibilità offerte da PIP. Anche se probabilmente userete solo alcune di queste possibilità, è importante conoscere tutte le opzioni disponibili. Pertanto siete invitati a leggere questo capitolo completamente e poi a ritornare indietro e studiare in dettaglio le sezioni di specifico interesse per voi.

Potrete essere sorpresi dall'ampiezza di possibilità offerte da PIP. Per menzionarne solo alcune, imparerete:

- Come congiungere i files (concatenazione).
- Come stampare testi formattati (con l'uso delle tabulazioni).
- Come tagliare righe che sono troppo lunghe per lo schermo (opzione D).
- Come stampare un gruppo di files con un solo comando (usando una specificazione di PRN).
- Come stampare automaticamente un testo in pagine con un formato (opzione P).
- Come leggere un file fino a una parola specifica, senza usare l'assembler (opzione Q).

LA COMPRENSIONE DI PIP

Pip è un programma di trasferimento di files. Le lettere PIP significano "Peripheral Interchange Command", cioé "comando di interscambio tra periferiche". Come indica il suo nome, PIP permette il trasferimento di files tra due dispositivi qualsiasi. Finora abbiamo realizzato soltanto il trasferimento dei files da disco a disco. Impareremo ora come

usare le opzioni addizionali disponibili con PIP. In particolare, PIP può anche elaborare i files mentre li trasferisce.

Dapprima presenteremo una descrizione completa della funzione più importante di PIP: la copiatura dei files. Poi, studieremo le possibilità di PIP per il trasferimento dei files tra vari dispositivi connessi al sistema di calcolo.

LA COPIATURA DEI FILES

La copiatura di un file singolo

PIP può essere eseguito in due modi:

1. Come singolo comando
2. Come "Programma"

Ecco un esempio che usa PIP come singolo comando:

```
A > PIP B: COPY1.BAK = FILE1.TXT ↵
A >
```

Si assume che FILE1.TXT sia sull'unità corrente, A. Questo comando indirizza PIP a fare una copia di FILE1.TXT, a dare alla nuova copia il nome COPY1.BAK e a mettere COPY1.BAK sull'unità B. Poi si rientra nel CP/M con il prompt di sistema (A>). Questo è un modo veloce per copiare un solo file.

PIP può anche essere eseguito come "programma" ed essere usato per realizzare una sequenza di operazioni di copiatura:

```
A > PIP ↵
* B:COPY2.BAK = FILE2.TXT ↵
* A: = B:SAMPLE.BAS ↵
* A: = B:PROG.FOR ↵
* ↵
A >
```

Quando il programma PIP è in esecuzione, stampa il prompt '*' di PIP. A questo punto possono essere eseguiti i comandi di PIP. La prima riga nell'esempio fa una copia del file FILE2.TXT, che è sull'unità A, la chiama la copia COPY2.BAK e mette COPY2.BAK sull'unità B. La riga successiva fa una copia di SAMPLE.BAS che è sull'unità B, usa il nome originale (SAMPLE.BAS) per la copia e la mette sull'unità A. Anche l'espressione successiva copia un file dall'unità B all'unità A. Un sempli-

ce RETURN (ritorno carrello) termina PIP, facendo tornare al CP/M: sullo schermo appare il solito prompt 'A>' del CP/M.

Presenteremo ora le regole per realizzare questi trasferimenti. Per fare una copia di un file da un dischetto a un altro, usate questo tipo di espressione PIP:

u:copia = u:originale

dove 'u' è la lettura dell'unità, 'copia' è il nuovo nome per la copia del file e 'originale' è il nome del file originale. Le due u sopra possono riferirsi a una stessa unità o a due unità differenti. La 'u' a destra può essere omessa, poiché PIP assume che il file sia nell'unità corrente. Anche la 'u' a sinistra può essere omessa se si fornisce il nome 'copia'. Se volete copiare un file con lo stesso nome del file originale, usate la forma abbreviata:

u' = u:originale

dove u' è una lettera di unità diversa da u:. Il nome della copia può essere omesso - PIP assumerà che il nome del file nella nuova unità sia lo stesso di quello originale. Però questo funziona soltanto se l'unità del file copia è differente da quelle del file originale, poiché non potete avere due files con lo stesso nome sullo stesso dischetto. Per esempio:

```
* B: = A:TEST.INT ↵
```

copierà TEST.INT da A a B.

Studiamo qualche esempio. Supponiamo di essere nell'unità A. Supponiamo che FILE1.NAD sia nell'unità A e PROGRAM.TXT sia nell'unità B. I seguenti comandi PIP sono legali?

```
    A > PIP ↵
(1) * A: = B:PROGRAM.TXT ↵
(2) * B: = FILE1.NAD ↵
(3) * A:FILEREV.NAD = A:FILE1.NAD ↵
(4) * A:FILE1.TXT = FILE1.NAD ↵
(5) * B:FILE1.NAD = FILE1.NAD ↵
```

Sono tutti legali. In (2), si noti che il comando è equivalente a:

```
B: = A:FILE1.NAD
```

Ricordate che l'unità corrente può essere omessa. In questo caso PIP assume che sia l'unità corrente, cioé A. Si sarebbe potuto mettere la A in (3). FILE1.TXT in (4) *non* è lo stesso di FILE1.NAD. Questo è permesso. (5) potrebbe essere stato abbreviato come (2).

La copiatura di più files

Si usano comandi multipli di PIP per copiare parecchi files differenti. Per esempio, copiamo i tre files:

```
FILE1.NAD
LETTER.TXT
PROGRAM.INT
```

da B ad A:

```
A > PIP ↵
* A: = B:FILE1.NAD ↵
* A: = B:LETTER.TXT ↵
* A: = B:PROGRAM.INT ↵
* ↵
A >
```

In casi speciali, però, questo trasferimento può essere semplificato con l'uso dei *simboli di maschera* di PIP. Per facilitare la copiatura di molti files, PIP permette l'uso di due simboli speciali: '?' e '*'. Il '?' può essere usato in un nome di file e corrisponde ad ogni carattare che può apparire al suo posto. Per esempio:

```
FILE?.NAD
```

corrisponderà a:

```
FILE1.NAD
FILE2.NAD
FILE 3.NAD
```

ma non: FILE44.NAD (un carattere in più)

Copiamo ora i tre files:

FILE1.NAD
FILE2.NAD
FILE3.NAD

dall'unità B all'unità A. Il comando è il seguente:

```
A > PIP A: = B:FILE?.NAD ↵
```

questo comando realizza da solo i tre trasferimenti con l'uso del carattere speciale di maschera. La direttrice di B sarà esaminata da PIP fino a quando tutte le possibilità di corrispondenza saranno esaurite. Si noti che se c'era su B un altro file di nome

FILES.NAD

sarebbe stato trasferito anch'esso.

Il secondo carattere di maschera '*', è ancora più potente. Corrisponde a qualsiasi cosa nel suo campo, *indipendentemente dalla lunghezza*. Per esempio, supponiamo che B contenga:

FILE1.NAD
FILE12.NAD
LETTER.TXT
CBASIC.INT
FILE1.BAK

Allora i caratteri *.NAD corrisponderanno a:

FILE1.NAD
FILE12.NAD

e FILE1.* corrisponderà a:

FILE1.NAD
FILE1.BAK

È possibile anche scrivere '*.*', che corrisponde a tutti i files sul dischetto

nell'unità B. Questo sarà usato nella sezione successiva per copiare tutti i files.

Per esempio, se vogliamo copiare tutti i files di tipo COM dall'unità A all'unità B, possiamo digitare semplicemente:

```
A > PIP B: = *.COM ↵
```

e tutti i comandi saranno copiati uno dopo l'altro sull'unità B.

Abbiamo ora imparato come copiare un file singolo o un gruppo di files. Di seguito impareremo come copiare un intero dischetto.

La copiatura di tutti i files

Si noti che questa sezione è intitolata "la copiatura di tutti i files" e non "La copiatura di un intero dischetto". Questo perché il CP/M non è memorizzato sottoforma di file. Per copiare il CP/M, si deve usare un comando speciale, SYSGEN, (descritto nel Capitolo 2). Se un dischetto contiene soltanto files, allora copieremo tutti i files. Se il dischetto contiene anche il CP/M, invece, copieremo soltanto i files, non il CP/M.

Non potete sapere se il CP/M sia su un dischetto semplicemente esaminando la direttrice con il comando DIR; il CP/M non è un file e non viene elencato tra i files. Se volete verificare se il CP/M sia o no su un dischetto, dovete cercare di eseguire il CP/M da questo dischetto facendo, per esempio, un CNTRL-C.

Usiamo ora la possibilità di mascheratura di PIP per copiare tutti i files. Una *maschera dei nomi di files* può essere usata come nome di file soltanto per il file *originale*. Per esempio, se volete copiare tutti i files dal dischetto A al dischetto B, dovete digitare questo comando:

```
A > PIP B: = A: *.* ↵
```

Usate questa forma di PIP per fare copie dei dischetti:

```
PIP u': = u: *.*
```

dove *u'* è la lettera dell'unità che contiene il dischetto nuovo e *u* è la lettera dell'unità che contiene il dischetto vecchio. La maschera '*.*' corrisponderà a *tutti i nome di files*. *u'* deve essere diversa da *u*.

In pratica, quando si copiano tutti i files, è raccomandabile digitare:

```
A > PIP B: = A: *.* [V] ↵
```

'V' è un'opzione di PIP che specifica "verifica"; controlla che la copia sia identica all'originale. Questo è il comando migliore da usare per sicurezza. Però il procedimento di copiatura diventa molto più lungo con l'opzione [V], per cui molti utenti non lo usano a meno che il file sia molto importante.
NOTA: ricordate che il comando PIP:

```
B: = A: *.*
```

copierà tutti i files di A, ma soltanto i files. Se A contiene il "sistema", cioé il CP/M, questo non sarà copiato, poiché non è un file. Ricordate che il CP/M stesso può essere copiato con il comando SYSGEN.

La copiatura di un dischetto

Su molti calcolatori è disponibile un programma speciale per copiare l'*intero dischetto* ad alta velocità. Se si usa questa opzione, il secondo disco sarà una copia completa del primo, compreso il CP/M, se era sul primo disco. Di solito questo è il modo più veloce per copiare un dischetto completo.

D'altra parte, quando PIP copia un file, lo copia su "blocchi" o "settori" adiacenti sul dischetto. Di conseguenza, il file copiato sarà accessibile in modo molto più veloce dai programmi quali ad esempio un editor (o elaboratore di testi) o un interprete (BASIC).

Sostanzialmente, se si usa PIP per copiare un dischetto il risultato sarà-dei files più "puliti". Se si usa un programma per la copiatura dei dischi, invece, si risparmierà tempo sull'operazione.

Il CP/M versione 2.2 e l'MP/M

L'espressione '*.*' come maschera dei nomi di files potrebbe non essere sufficiente per la vostra installazione se avete *aree utente* separate (le aree utente sono descritte nel Capitolo 2). Se fate uso di queste aree, il vostro sistema dovrebbe avere un programma speciale per duplicare i dischetti. Se trovate un errore di "invalid format" (formato non valido), premete un tasto qualsiasi *eccettuato* il tasto RETURN per riottenere il prompt di PIP. Se non si trova a PIP, provate di nuovo. Se premete RETURN, PIP termina e rientra il sistema operativo con il suo promtp.

La copiatura su un dischetto nuovo

I due metodi

Se avete tre unità a dischi o più, inserite semplicemente un nuovo dischetto sull'unità C e copiate il file da B a C, cioé dalla seconda unità

alla terza. Se avete soltanto due unità, invece, il procedimento di copiatura è più complicato. Supponiamo che il disco di sistema sia nell'unità A e il file da copiare nell'unità B. Vogliamo copiarlo su un dischetto nuovo. Si possono usare due metodi: trasferirlo attraverso A e scambiare i dischetti.

Il trasferimento attraverso A

Questo metodo è sicuro ma lento, e funziona soltanto se il dischetto nell'unità A ha spazio sufficiente. (Vedremo più avanti come verificare lo spazio disponibile con i comandi STAT o DIR). Il metodo usato è molto semplice:

1. Si trasferisce il file da B ad A.
2. Si mette un nuovo dischetto in B.
3. Si trasferisce il file da A a B sul dischetto nuovo.

Ricordate che si deve eseguire un ↑ C prima del passo 3, in modo che il CP/M possa scrivere su un dischetto nuovo. Per esempio:

```
A > A: = B:FILE.INT ↵
(mettete un dischetto nuovo in B)
↑ C
A > B: = A:FILE.INT ↵
```

Poi verificate che il file sia sul dischetto B analizzando la direttrice (usando 'DIR'), e cancellate la copia in più su A ('ERA').

Sostituzione dei dischetti

Questo metodo trasferisce direttamente il file sul nuovo dischetto. Perché il trasferimento avvenga, sono necessarie tre condizioni contemporaneamente:

1. PIP deve essere in esecuzione.
2. Il dischetto contenente il file sorgente deve essere in un'unità.
3. Il dischetto destinazione deve essere in un'unità.

Si carica PIP dal dischetto nella memoria del calcolatore. Poi si toglie il dischetto di sistema e si esegue PIP in memoria. In altre parole una volta che è stato chiamto PIP, il programma PIP è caricato nella memoria del calcolatore e il dischetto di sistema può essere sostituito da un altro dischetto per il trasferimento.

Supponendo che il dischetto di sistema con PIP sia nell'unità A, il procedimento è il seguente:

1. Inserire un dischetto nuovo in B.
2. Premere CTRL-C per realizzare una "partenza a caldo". Questo permette al CP/M di riconoscere un nuovo dischetto in B e di scrivere su di esso.
3. Chiamare PIP digitando: 'PIP(CR)':

```
A > PIP ↵
* ↵
```

ora PIP è nella memoria, pronto per essere eseguito.

A questo punto potete togliere il dischetto di sistema per un *momento* e inserire il dischetto originale (quello che dovete copiare) nell'unità A. Può sembrar strano suggerire di togliere il dischetto di sistema sul quale risiede PIP. Ricordate comunque che quando un programma di tipo COM è in esecuzione, è caricato dal disco nella memoria del calcolatore.

PIP è stato chiamato digitando:

```
A > PIP ↵
```

Una copia di PIP è ora nella memoria del calcolatore. Non abbiamo più bisogno del dischetto e possiamo toglierlo fino a quando vogliamo uscire da PIP.

NOTA: non terminate PIP fino a che non avete rimesso il dischetto di sistema.

Non digitate un RETURN dopo un prompt, quando PIP è stato attivato:

```
* ↵      (NO!)
```

perché questo terminerebbe PIP. Ancora, non usate CTRL-C fino a che non avete rimesso il dischetto di sistema. Riferitevi a "Consigli pratici" nel Capitolo 7 per imparare una salvaguardia effettiva contro le uscite accidentali da PIP.

Ora inserite il dischetto da copiare nell'unità A. Al prompt '*' di PIP potete digitare un'espressione PIP. Tenete a mente che il dischetto originale è nell'unità A e che il dischetto nuovo è nell'unità B. Se copiate l'intero dischetto originale, dovete digitare questa espressione:

```
* B: = A: *.* ↵
```

O se preferite:

```
* B: = A: *.* [V] ↵
```

con l'opzione di verifica.

'*.*' è la maschera per i nomi di tutti i files (sotto CP/M versione 2.2 e MP/M, '*.* [V]' corrisponde a tutti i files soltanto nella *vostra area utente*). Questa espressione PIP copia tutti i files dall'unità A (dischetto originale) all'unità B usando gli stessi nomi per le copie dei files. Dopo l'esecuzione, avrete una copia di ogni file sul nuovo dischetto e i files avranno lo stesso nome di prima.

A questo punto siete pronti per tornare nel sistema. Prima di terminare PIP, togliete il dischetto originale dall'unità A e mettete nell'unità A il dischetto di sistema. Ora potete terminare PIP premendo semplicemente il tasto RETURN:

```
* ↵

A >
```

Se non si torna al sistema, verificate l'unità A. Se la luce sull'unità A è accesa, potete inserire il dischetto di sistema e si ritornerà al sistema. Se la luce è spenta, dovete inserire il dischetto di sistema e rilanciarlo, come descritto nel Capitolo 1.

Le installazioni che hanno soltanto due unità devono usare il metodo appena descritto:

1. Inserire un dischetto nuovo nell'unità B.
2. Eseguire PIP.
3. Togliere il dischetto di sistema.
4. Mettere il dischetto originale nell'unità A.
5. Eseguire le espressioni PIP per le operazioni di copiatura.
6. Quando si è terminato, togliere il dischetto originale.
7. Rimettere il dischetto di sistema.
8. Terminare PIP.

Ricordate di mettere il dischetto nuovo nell'unità B e il dischetto originale nell'unità A. Se in B vi era un altro dischetto, eseguite un CTRL-C prima di fare qualsiasi cosa altrimenti il CP/M rifiuterà di scrivere su un dischetto che non è "conosciuto". CTRL-C forzerà il CP/M a tener conto del dischetto nuovo. Si tenga presente che il metodo contrario non funziona. Se invece del dischetto di sistema avessimo messo un dischetto nuovo in A, il CP/M rifiuterebbe di scrivere su di esso perché non potrebbe conoscerlo.

Una volta che il dischetto di sistema è stato tolto, non potete più

eseguire un CTRL-C per cui non c'é il modo di registrare il nuovo dischetto in A. Comunque, questo è un consiglio pratico: fate una copia di CP/M e PIP sul vostro dischetto nuovo; poi potrete metterlo in qualsiasi unità e fare copie da esso convenientemente.

CDOS Cromemco

CDOS non richiede un CTRL-C per scrivere su un dischetto nuovo; pertanto il procedimento di copiatura può essere semplificato. Con il sistema nell'unità A e il dischetto da copiare nell'unità B, eseguite 'PIP ↵' come prima poi:

1. Togliete il dischetto di sistema.
2. Mettete il nuovo dischetto in A.
3. Realizzate il trasferimento.
4. Togliete il nuovo dischetto.
5. Rimettete il dischetto di sistema in A.
6. Premete "RETURN" per terminare PIP.

Il procedimento raccomandato

Fino a quando non avete sufficiente esperienza col CP/M, dovreste trasferire i files usando il primo metodo (copiando attraverso A), perché è il più sicuro. Ancora meglio, copiare il CP/M e PIP su tutti i dischetti in modo che non sia più necessario scambiarli. Se volete provare il secondo metodo di trasferimento di un file, ricordate che potete danneggiare il dischetto originale se uscite da PIP troppo presto premendo RETURN senza aver tolto il dischetto originale ricoprendo i files.

L'interruzione di un'operazione di copiatura

Se si preme un qualsiasi carattere sulla tastiera durante un trasferimento con PIP, questo normalmente viene abortito. PIP lo conferma stampando il messaggio 'ABORTED'.

LA COPIATURA SUI DISPOSITIVI

Introduzione

La stampa di un file è un esempio di un'operazione di trasferimento. Il file viene copiato dal disco sulla stampante. PIP fornisce delle funzionalità generiche di trasferimento e permette di trasferire un file non solo da disco a disco, ma tra vari dispositivi.

Queste possibilità generali saranno descritte nel seguito. Vogliamo imparare come trasferire tra due qualsiasi dispositivi (ragionevoli) che

possono essere attaccati a un calcolatore. Introdurremo nuove caratteristiche di PIP come la concatenazione, che può essere usata su tutti i trasferimenti di files, compreso quello da disco a disco. Anche se non prevedete di usare un lettore di schede, è importante leggere questa sezione in modo completo, perché si applica anche alla stampa e alla copiatura dei files.

Considereremo prima l'operazione usata più di frequente: la stampa.

La stampa di un file

Un file può essere stampato da PIP o da altri programmi. Se usate un (word processor elaboratore di testi) o un altro programma speciale che ha la "possibilità di stampa" (la possibilità di mandare un file alla stampante) potete usare quel programma per stampare i files creati e accessibili al programma. Per esempio, se create o aggiungete dati a un file con "nomi e indirizzi" usando un insieme di programmi realizzati a questo scopo, è possibile che vi sia anche un modo di stampare il file usando un programma speciale fornito come parte di questo stesso insieme di programmi.

Il vantaggio maggiore di un programma di stampa specializzato è che stampa il file in un formato specifico. Un programma di stampa per esempio può fornire automaticamente la realizzazione del formato, le tabulazioni, le spaziature, le impaginazioni e altre opzioni di stampa.

Anche il comando TYPE del CP/M può essere usato per stampare un file alfanumerico (vedi Capitolo 1). Per esempio:

```
A > ↑ P
A > TYPE FILE.TXT ↵
```

Il CTRL-A rende attiva la stampante se non lo era. Questo comando è semplice da usare e fornisce una stampa "grezza", in altre parole il file appare sulla stampante esattamente come era sul disco senza alcun nuovo formato.

Soltanto una possibilità viene fornita da TYPE; espande i caratteri di tabulazione (CTRL-I) contenuti nel file assumendo una posizione di tabulazione ogni otto colonne. Il comando TYPE è usato per una stampa veloce, per guardare l'inizio di un file o, più spesso per visualizzare un file velocemente sullo schermo invece che sulla stampante. Il terminale CRT può visualizzare un testo alla velocità di 9600 baud contro i 300 o 600 baud (approssimativi) di una stampante. Pertanto un'operazione di TYPE sullo schermo stampa il testo molto più velocemente che se fosse mandato sulla stampante.

I files possono essere stampati anche con PIP, come parte delle sue

possibilità generali di trasferimento. Sarà ora descritto questo procedimento.

Il trasferimento dei files

Un file può essere mandato a qualsiasi dispositivo in grado di riceverlo. Per esempio, un file può essere mandato a un'unità a dischi, a una stampante, a un monitor video, a un perforatore di nastro di carta e a un registratore a cassetta. Non può essere mandato a un lettore di schede o a una tastiera.

Una stampante senza tastiera può soltanto ricevere files. Una stampante con tastiera diventa un terminale e la tastiera può generare un file.

La Fig. 3.1 mostra come un utente può "leggere" da un dispositivo (input) e "scrivere" su un dispositvo (output). Il calcolatore esegue tutti i programmi e trasferisce tutte le informazioni.

Per stampare un file sulla stampante, il calcolatore prima legge il file dal disco (input) e poi lo trasferisce sulla stampante (output). La maggior parte dei programmi legge i files dal disco a blocchi (un settore per volta) perciò possono trasferire un file di grandi dimensioni usando soltanto una piccola quantità di memoria interna. Questo è mostrato nella Fig. 3.2.

I caratteri di un file possono anche essere inseriti da una tastiera o mandati allo schermo. Allo stesso modo possono essere mandati al disco in un trasferimento da disco a disco. Specifici programmi di trasferimento dei files possono essere disponibili per fornire queste possibilità. Spesso, programmi specializzati realizzano una di queste funzioni all'interno di un pacchetto applicativo. Comunque, il programma PIP (un comando) può essere usato come strumento generale per mandare una copia di un file a un dispositivo o ricevere informazioni da un dispositivo per metterle in un file. Molte espressioni PIP, insieme a parole chiave speciali usate per rappresentare i dispositivi, possono realizzare potenti e complesse operazioni di copiatura. Queste operazioni saranno descritte qui.

Per imparare le espressioni valide di PIP, bisogna indicare in modo proprio i dispositivi. Dapprima saranno esaminate le convenzioni di PIP per specificare i dispositivi.

La specificazione dei dispositivi

Le parole chiave usate nelle espressioni di PIP sono nomi di dispositivi sia "logici" che fisici. Un nome *fisico* di dispositivo è il nome effettivo di un dispositivo attaccato al sistema. Un nome *logico* di dispositivo è come un nome "generico" che può essere in effetti uno dei molti dispositivi assegnati al sistema. Per esempio, LST: è il nome del dispositivo di

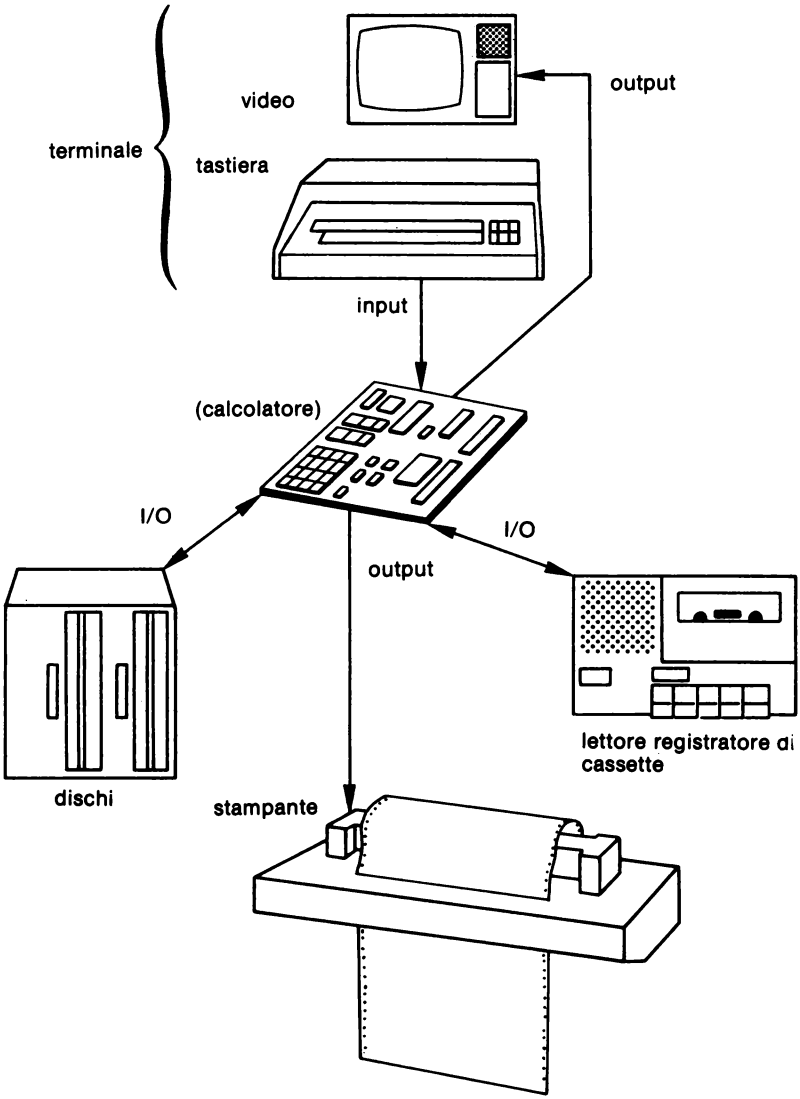


Figura 3.1: Gli elementi di un sistema.

stampa ("list"), che è di solito la stampante ma che potrebbe essere un altro dispositivo come una teletype (telescrivente) o un modem usato per comunicare su una linea telefonica. Non dovete conoscere il nome proprio del dispositivo (come Teletype o Hazeltine), ma il nome logico.

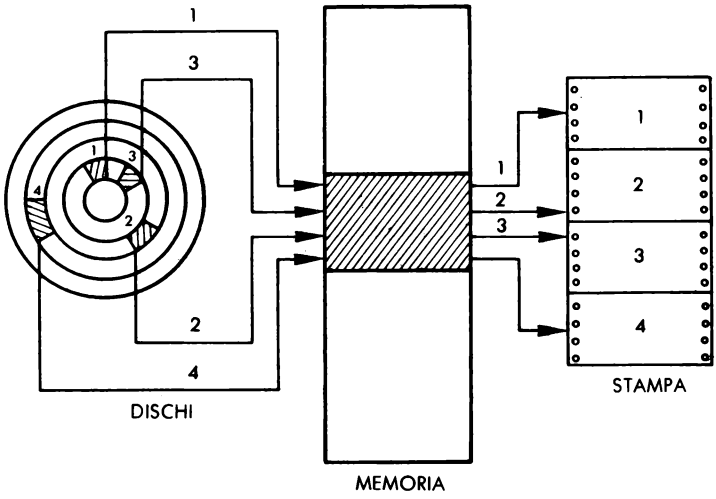


Figura 3.2: Trasferimento attraverso la memoria.

I nomi logici di dispositivi permessi in espressioni PIP sono:

CON: per "console" o terminale, comprendente stampante e video (input/output);
RDR: per il lettore di nastro di carta o di schede (solo input);
PUN: per il perforatore di nastro di carta o di schede (solo output);
LST: per il dispositivo di stampa come una stampante (solo output).

Notate che gli assegnamenti che farete per RDR: e PUN: probabilmente non saranno dispositivi per nastro di carta o schede perforate, poiché quei dispositivi stanno diventando rapidamente obsoleti. Sono usati più spesso per operazioni di ingresso e uscita di tipo "batch" (discusse nei Capitoli 2 e 4). Inoltre il dispositivo CON: usualmente è il terminale CRT (Cathode Ray Tube - Tubo a Raggi Catodici) con una tastiera, quindi è un dispositivo di input-output. Il dispositivo LST: è solitamente la stampante. Però sia il dispositivo di CON: che il dispositivo LST: potrebbero essere telescriventi o qualsiasi altra stampante fornita di una tastiera.

Potreste chiedervi come PIP conosca la velocità alla quale trasmettere i caratteri, poiché le stampanti operano a differenti velocità. Questo avviene perché il CP/M è sempre "confezionato" per un'installazione specifica. Quando il CP/M è stato "configurato" per la vostra installazione, sono stati inseriti programmi specifici per la vostra stampante, il vostro video CRT e il vostro controllore dei dischi. Se cambiate i dispositivi di input/output, dovete cambiare insieme il dischetto CP/M.

Il programma STAT (comando), descritto nel Capitolo 2, stampa gli assegnamenti dei dispositivi per ciascun nome logico. Potete anche cambiare questi assegnamenti usando il comando STAT.

Esempi pratici

Potete usare i nomi logici (descritti prima) nelle espressioni PIP. Per esempio:

```
A > PIP ↵
*CON: = SAMPLE.TXT ↵
*LST: = B:SAMPLE.BAK ↵
*PROG.BAS = RDR: ↵
*PUN: = PROG.BAS ↵
* ↵
A >
```

La prima espressione PIP manda una copia del file SAMPLE.TXT (che è nell'unità corrente [A]) al dispositivo di console (probabilmente il terminale video) (Fig. 3.3).

La seconda espressione trasferisce una copia del file SIMPLE.BAK nell'unità B al dispositivo corrente LST: (solitamente la stampante o la telescrivente) (Fig. 3.4).

La terza espressione PIP legge l'informazione proveniente dal dispositivo di lettura RDR: e crea il file PROG.BAS. Questo è di solito un programma su nastro di carta, schede perforate o cassetta che può essere letto nel sistema e memorizzato in un file su disco usando quest'espressione PIP (Fig. 3.5).

L'operazione opposta, di mandare una copia del file al perforatore del nastro di carta o di schede perforate o al registratore a cassetta, è l'ultima espressione PIP, che manda una copia di PROG.BAS al dispositivo PUN: (Fig. 3.6).

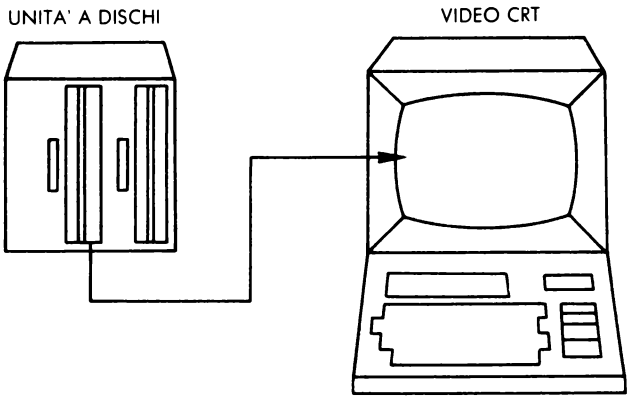


Figura 3.3: Trasferimento di un file alla console: CON: = SAMPLE.TXT

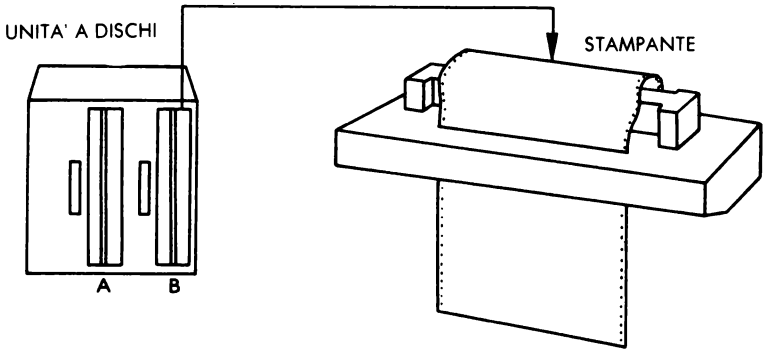


Figura 3.4: LST: = B: SIMPLE.BAK

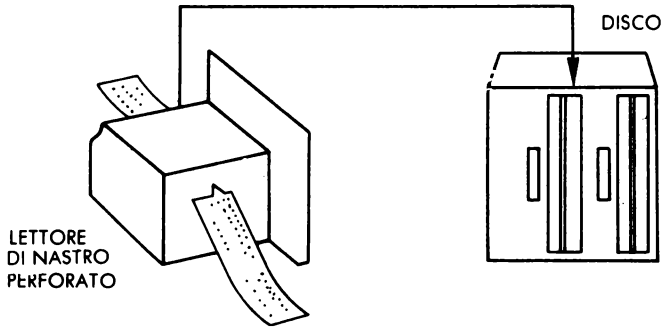


Figura 3.5: PROG.BAS = RDR:

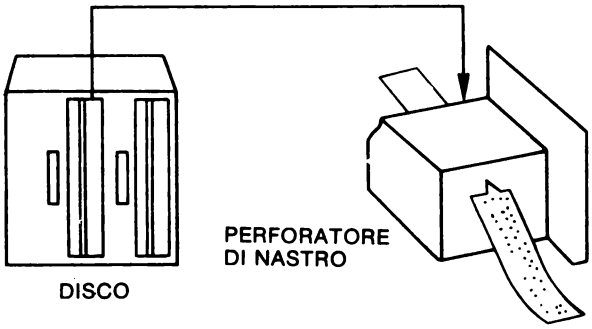


Figura 3.6: PUN: = PROG.BAS

Nomi fisici dei dispositivi

All'occorrenza, possono essere usati nelle espressioni PIP anche i nomi fisici dei dispositivi. I seguenti sono nomi fisici validi di dispositivi:

TTY:	per un console o terminale, un lettore, un perforatore o un dispositivo di stampa (telescrivente)
CTR:	per un console o un terminale, o un dispositivo di stampa (video)
PTR:	per un lettore di nastro di carta o di schede
PTP:	per un perforatore di nastro di carta o di schede
LPT	per un dispositivo di stampa (stampante)
UC1:	per un console o un terminale definito dall'utente
UR1:	per un lettore definito dall'utente
UR2:	per un secondo lettore definito dall'utente
UP1:	per un dispositivo di uscita (perforatore) definito dall'utente
UP2:	per un secondo dispositivo di uscita (perforatore) definito dall'utente
UL1:	per un dispositivo di stampa definito dall'utente

NOTA: BAT: non è incluso, poiché riassegna soltanto i valori per RDT: e LST:

OPERAZIONI SPECIALI DI COPIATURA

Introduzione

Abbiamo imparato ora come realizzare tute le operazioni semplici di copiatura. In questa sezione impareremo come realizzare operazioni più

complesse di trasferimento sui file di testo e i "files hex". PIP non è soltanto un semplice programma di "copiatura", ma un programma generale di trasferimento dotato di alcune opzioni di trattamento. Queste opzioni di trattamento saranno descritte ora in dettaglio.

Nomi dei dispositivi speciali

PIP fornisce nomi di dispositivi speciali che permettono un trattamento speciale di un file. Si possono usare i seguenti nomi addizionali di dispositivi quando di effettuano trasferimenti con PIP:

NUL: manda 40 "null" (codice ASCII 0) al dispositivo, solitamente un perforatore in uscita. Esempio (dove PROG.HEX viene mandato al perforatore):

```
*PUN: = PROG.HEX,NULL: ↵
```

EOF: manda un "end-of-file" (codice ASCII ↑ Z) al dispositivo (mandato automaticamente da PIP durante i trasferimenti dei file di testo in codice ASCII e necessario soltanto in casi speciali). Esempio:

```
*PUN: = NUL:,X.ASM,EOF:,NULL: ↵
```

Questo esempio manda 40 null al dispositivo di perforazione, seguiti da una copia del file X.ASM, seguita da un carattere di end-of-file (↑ Z) ed altri 40 null.

PRN: lo stesso di LST: (trasmette alla stampante), tranne che le tabulazioni sono espanse ogni otto caratteri, le righe sono numerate (come nel programma ED) e viene inserito un avanzamento a pagina nuova (form feeds) ogni 60 righe (per far avanzare la carta della stampante alla pagina successiva) con un posizionamento iniziale all'inizio della pagina. Esempio:

```
*PRN: = SAMPLE.TXT ↵
```

INP: codice speciale di dispositivo di input che può essere "collegato" allo stesso programma PIP (dovete scrivere un programma in linguaggio assembler e aggiungerlo a PIP). PIP riceve il carattere di input chiamando una locazione di memoria (103H) e memorizzando il dato a partire dalla locazione 109H (il bit di parità deve essere zero usate il parametro Z).

OUT: codice speciale di dispositivo di output che può essere "collegato" allo stesso programma PIP, come INP: descritto sopra. PIP chiama la locazione 106H e manda il dato nel registro C (carattere per carattere).
NOTA per programmatori in linguaggio assembler: le locazioni da 109H a 1FFH della memoria di PIP non sono utilizzate possono essere sostituite con codice per la gestione di dispositivi speciali (usate DDT il Debugger del CP/M fornito dalla Digital Research con il CP/M e l'MP/M). Esempio:

```
*MODEL.CLK = INP ↵
```

(L'input dal dispositivo speciale viene memorizzato nel file MODEL.CLK)

```
*OUT: = MODEL.CLK ↵
```

(Una copia di MODEL.CLK viene mandata al dispositivo speciale)

Files di testo (ASCII) trasmessi ai dispositivi

La maggior parte di files di dati e *tutti* i files di testo creati dai programmi "editor" o dai programmi di gestione di testi sono files di testo in formato ASCII. Gli altri files, come i files di comando (.COM), i files di programma intermedi del BASIC (.INT) e i files in linguaggio macchina (.HEX) sono effettivamente programmi scritti in un linguaggio ad alto livello (come il BASIC) o in un linguaggio assembler dove i codici binari rappresentano i numeri effettivi o istruzioni, *non* testi.

È importante conoscere la differenza tra i files per realizzare operazioni speciali di copiatura, come la traduzione di caratteri maiuscoli in caratteri minuscoli o la cancellazione di caratteri di un'altra copiatura o la copiatura di porzioni di un file. Si possono fare queste cose soltanto con i files di testo ASCII, perché PIP si aspetta un certo carattere (il carattere prodotto premendo CTRL e Z contemporaneamente cioé ↑ Z) alla fine del file in modo che può facilmente cercare i caratteri. Potete anche concatenare (congiungere) più files di testo ASCII usando PIP (descritto più avanti).

Certi dispositivi possono ricevere o mandare soltanto file di testo ASCII. Potete mandare soltanto file di testo alle stampanti e alla console, per esempio, ma altri dispositivi (RDR: e PUN:) possono mandare o ricevere qualsiasi tipo di file. Le informazioni in un file possono essere codificate in più modi. Per esempio, un file di testo normalmente è codificato nel formato ASCII, dove un codice a 8-bit (un "byte") viene

usato per rappresentare ciascun carattere, compreso il carattere speciale di controllo. Questo codice è riportato in Fig. 3.7.

NUMERO DEI BIT								0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
b_7	b_6	b_5	b_4	b_3	b_2	b_1	HEX 1 / HEX 0	0	1	2	3	4	5	6	7
			0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
			0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
			0	0	1	0	2	STX	DC2	"	2	B	R	b	r
			0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
			0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
			0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
			0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
			0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
			1	0	0	0	8	BS	CAN	(	8	H	X	h	x
			1	0	0	1	9	HT	EM	)	9	I	Y	i	y
			1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
			1	0	1	1	11	VT	ESC	+	;	K	[	k	{
			1	1	0	0	12	FF	FS	,	<	L	\	l	|
			1	1	0	1	13	CR	GS	-	=	M	]	m	}
			1	1	1	0	14	SO	RS	.	>	N	^	n	~
			1	1	1	1	15	SI	US	/	?	O	_	o	DEL

NUL — Nullo
SOH — Partenza del primo blocco
STH — Partenza del testo
ETX — Fine del testo
EOT — Fine della trasmissione
ENQ — Interrogazione
ACK — Avvisatore di ricevimento
BEL — Richiamo
BS — Ritorno indietro
HT — Tabulazione orizzontale
LF — Avanzamento di un'interlinea

VT — Tabulazione verticale
FF — Alimentazione di carta
CR — Ritorno carrello
SO — Codice semplice
SI — Codice normale

DLE — Uscita trasmissione
DC — Controllo della periferica
NAK — Avvisatore di ricevimento-negativo
SYN — Aspettativa sincronizzazione
ETB — Fine del blocco di trasmissione
CAN — Annullamento
EM — Fine del supporto
SUB — Rimpiazzamento
ESC — Cambiamento di codice
FS — Separatore di file
GS — Separatore di gruppo
RS — Separatore di record
US — Separatore di unità
SP — Spazio (black)
DEL — Cancellazione

Figura 3.7: Tabella di conversione ASCII

Invece, i programmi che sono stati trattati da un compilatore sono solitamente rappresentati in un codice più compatto, chiamato esadecimale, che usa soltanto 14 bit per rappresentare 16 simboli (Fig. 3.8).

DECIMALE	BINARIO	ESADECIMALE
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Figura 3.8: Tabella di conversione esadecimale

Quando si trasferisce un file a una stampante o a un video, è essenziale specificare se è esadecimale (due cifre per byte) o ASCII (un carattere per byte). PIP trasferisce un file fino a quando raggiunge il carattere di end-of file (fine del file) in un file di testo ASCII (↑ Z), o la effettiva fine del file negli altri files (tranne quando PIP trasferisce soltanto porzioni di un file).

La concatenazione dei files di testo

La concatenazione è un comando importante usato per raggruppare più files di testo in uno. Si tenga presente comunque che lo spazio disponibile su un disco deve essere sufficiente per contenere il file finale.

L'esempio più semplice è la concatenazione di due files:

```
A > PIP ↵
*BIG.TXT = PART1.TXT.PART2.TXT ↵
```

Potete anche congiungere più files di testo in una volta sola:

```
A > PIP ↵
*FINAL.ASM = SUB1.ASM,SUB2.ASM,TEMP.ASM ↵
*B:NEW.ZOT = A:OLD.ZAP,B:OLD.ZOT,A:NEW.ZAP ↵
* ↵
A >
```

In questo esempio, le copie dei files SUB1.ASM, SUB2.ASM e TEMP.ASM (tutti sull'unità corrente, l'unità A) sono congiunte in questo ordine (cioé, SUB1.ASM è il primo, SUB.2.ASM è il secondo, ecc.) in una copia chiamata FINAL.ASM. La seconda espressione PIP congiunge una copia di OLD.ZAP sull'unità A con OLD.ZOT sull'unità B e con NEW.ZAP sull'unità A, e la copia risultante viene messa sull'unità B chiamata NEW.ZOT.

PIP assume sempre che questi siano files di *testo*, ciascuno terminante con il carattere end-of-file (codice ASCII ↑ Z). Se sono files di testo, PIP non ha alcun problema a congiungerli (togliendo il carattere ↑ Z e mettendone uno alla fine della nuova copia per indicare la fine del file). Se *non* sono files di testo, invece, dovete leggere alla sezione successiva, "La concatenazione dei files non di testo".

Un altro modo di concatenare è di specificare il primo file come la *copia*. Per esempio:

```
A > PIP FIRST.TXT = FIRST.TXT, SECOND.TXT,
THIRD.TXT ↵
A >
```

Questo comando *non* cambia il contenuto iniziale di FIRST.TXT o degli altri file ma *appende* (aggiunge alla fine) a FIRST.TXT i files SECOND.TXT e THIRD.TXT. Il file finale FIRST.TXT conterrà all'inizio il vecchio contenuto del file iniziale FIRST.TXT.

La concatenazione può essere usata per stampare o esaminare più files in una volta, con un singolo comando. Per esempio:

```
A > PIP LPT: = FIRST.TXT,SECOND.TXT
```

stamperà i due files in sequenza sulla stampante.

Quando si usa il programma XFER della Cromemco invece di PIP, i caratteri di controllo sono differenti da PIP e devono essere imparati in modo specifico. In particolare, XFER richiede un carattere di controllo

per concatenare i files (altrimenti il CTRL-Z sarà lasciato alla fine di ciascun file).

Un consiglio pratico: quando concatenate i files, assicuratevi che ci sia spazio sufficiente sul dischetto per il file risultante.

La concatenazione dei files non di testo

Quando concatenate (congiungete) files non di testo, ciascun file *non* ha un carattere di end-of-file, pertanto PIP non copia dopo avere raggiunto la fine effettiva di un file (cioé, se non c'é un carattere di end-of-file, proprio la fine del file). Per forzare PIP a copiare il file successivo e a congiungerlo a quello precedente, dovete usare un "parametro di trasferimento" indicato qui come "X" (trattato in "Parametri nelle operazioni di copiatura" in questo capitolo). I parametri sono racchiusi tra parentesi ([ ]) e devono apparire nelle espressioni PIP dopo il file o il dispositivo a cui si applicano. Per esempio:

```
A > PIP FINAL.HEX = TEMP1 [X], TEMP2 [X],
TEMP3 ↵
```

Questo comando PIP concatena copie dei files TEMP1, TEMP2 e TEMP3 e chiama la copia risultante FINAL.HEX. Il parametro X forza PIP a trascurare la fine effettiva dei files TEMP1 e TEMP2, e realizzare la concatenazione (usato soltanto con i files non di testo).

La copiatura dei files hex

I files esadecimali sono creati generalmente da un *assembler*. Un assembler traduce un programma in linguaggio assembly in *codice macchina* (sequenza di numeri binari corrispondenti alle istruzioni) che è memorizzato in un file esadecimale.

L'assembler CP/M crea un file HEX, cioé un file con l'estensione HEX. L'estensione HEX ha un significato speciale per PIP: PIP assume che il file sia nel "formato esadecimale" Intel e PIP automaticamente verifica la correttezza del formato, dei valori esadecimali e del Checksums (controllo di correttezza). Pertanto l'estensione HEX deve essere usata con cautela.

Se volete fare una copia di un file tipo HEX, dovete usare un'espressione PIP con il parametro H o I (per il trasferimento di dati *esadecimali*). Quando usate il parametro H, PIP verifica tutti i dati per assicurare la correttezza del formato esadecimale Intel. Se c'é un errore nei dati (cioé non hanno il formato esadecimale corretto) PIP interroga il terminale per creare un'azione correttiva. Il parametro H elimina anche i caratteri

non essenziali tra i records esadecimali durante le operazioni di copiatura.

Il parametro I automaticamente comprende il parametro H (fa quello che fa H e più). Se usate il parametro I, PIP ignora i records ':00' nel file originale in formato esadecimale (esadecimale Intel) e verifica se vi è un formato esadecimale improprio.

Se copiate *da un dispositivo* a un file con l'estensione esplicita 'HEX' PIP verifica la correttezza del formato esadecimale Intel e il Checksum. In altre parole, se copiate da un lettore di nastro di carta a un nuovo file SAMPLE.HEX, non dovete usare il parametro H per verificare la correttezza del formato esadecimale (dovete invece usare il parametro I per eliminare i records ':00').

Se PIP rileva un formato non valido o un errore di Checksum, riporta l'errore sul terminale e attende l'azione correttiva.

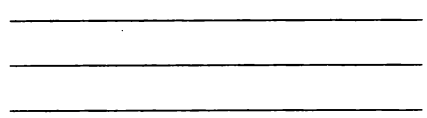
Se copiate nel nastro perforato, normalmente potete tirarlo indietro di circa 20 pollici e rilanciarlo. Quando il nastro è pronto, digitate un RETURN e PIP tenterà di copiare dal nastro.

NOTA: se il dispositivo è RDR: potete introdurre il carattere di fine file (↑ Z) dalla tastiera del terminale mentre PIP sta copiando. PIP legge dal dispositivo verificando la tastiera e attende che digitiate un ↑ Z per terminare l'operazione di copiatura.

Ecco alcuni esempi di espressioni di PIP che usano questa estensione:

```
*X.HEX = CON:, Y.HEX [I], PTR: ↵

Records
in formato
HEX
digitati al
terminale
```



```
↑Z
*
```

In questa espressione, PIP copia in X.HEX prendendo i dati dapprima dal dispositivo CON: (il terminale) fino a quando digitate un ↑Z. Poi, PIP copia da Y.HEX e ignora i records '.00". Infine, PIP copia dal lettore del nastro di carta PTR: fino a quando incontra un end-of-file (↑Z).

```
*PROG.X = KLUDGE.HEX [H] ↵
*
```

Questa espressione copia il file in formato esadecimale KLUDGE.HEX in PROG.X e verifica la correttezza esadecimale durante il trasferimento.

I PARAMETRI NELLE OPERAZIONI DI COPIATURA

I parametri

Descriveremo ora alcune delle opzioni di trattamento disponibili durante i trasferimenti. Anche se generalmente ne userete soltato alcune, è importante sapere che esistono. I parametri sono lettere speciali racchiuse in parentesi quadrate che seguono un nome di file, in un'espressione PIP e hanno effetto sulla copia di quel file. Potete specificare più di un parametro in un'espressione PIP. Alcuni parametri richiedono altre lettere o cifre. Queste sono tutte operazioni avanzate di copiatura e la loro conoscenza non è richiesta al normale utente di PIP.

B Trasferimento a blocchi. PIP mette i dati in un buffer fino a quando legge un carattere ASCII 'X-off' (↑ S) dal dispositivo. A questo punto PIP svuota il buffer su disco e torna ad attendere i dati. L'estensione del buffer dipende dall'estensione del sistema (vedere la documentazione fornita col sistema). Usate questo parametro per trasferire dati da un dispositivo a lettura continua come un registratore a cassetta. Esempio:

```
*SERVE.TXT = RDR: [B] ↵
```

Dn PIP cancella i caratteri che si trovano dopo la colonna 'n' (colonne verticali sul terminale) durante la copiatura dei files di testo. Usatelo per troncare le righe lunghe se mandate un file a un "dispositivo stretto", come una stampante a basso costo o un monitor a 40 colonne. Esempio:

```
*PRN: = LONG.TXT [D40] ↵
```

Questo comando può anche essere usato per "tagliare" i commenti da un programma se appaiono in una posizione specifica.

E Ristampa tutte le operazioni di copiatura sullo schermo del terminale nel momento in cui vengono realizzate. Esempio:

```
*COPY.TXT = SOURCE.TXT,S2.TXT,S3.TXT,S4.
 TXT [E] ↵
```

È utile nel caso di una sequenza di trasferimenti.

F PIP filtra i caratteri di posizionamento pagina dal file (cioé li elimina). Potete anche usare il parametro P per inserire nuovi caratteri di posizionamento pagina. Questo vi permette di "pulire" un file dopo averlo modificato, per una stampa impaginata.

H Trasferimento di dati esadecimali: PIP verifica se tutti i dati sono nel formato esadecimale Intel corretto. È richiesto un file .HEX.

I Ignora i records '.00' nel trasferimento di un file formato esadecimale Intel (comprende il parametro H). Richiede un file .HEX.

L Traduce tutti i caratteri maiuscoli in caratteri minuscoli.

N Inserisce il numero di riga in ogni riga copiata del nuovo file (incominciando dal numero1). Ogni numero di riga è seguito da un "due punti". Gli zeri iniziali (ad esempio 003) vengono trascurati, a meno che specifichiate il parametro 'N2'. 'N2' lascia gli zeri iniziali e inserisce un carattere di tabulazione dopo i numeri. Potete espandere gli spazi corrispondenti ai caratteri di tabulazione usando il parametro T. Quest'opzione è utile per i riferimenti nelle stampe.

O Trasferimento di un file oggetto (per i files non ASCII): PIP ignora la fine fisica del file durante la concatenazione (vedi "La concatenazione dei files").

Pn PIP inserisce il posizionamento a pagina nuova ogni 'n' linee (con un posizionamento iniziale a pagina nuova). Se n è 1 (o se non specificate 'n'), il salto a pagina nuova avviene ogni 60 righe. (Questa è chiamata specificazione di *"default"*.) Se usate anche il parametro F, PIP elimina i vecchi posizionamenti prima di inserire i nuovi. Questo è un modo conveniente di stampare in un certo formato di pagina.

Q stringa ↑ Z PIP interrompe la copiatura del dispositivo del file quando trova la 'stringa' di caratteri specificata (una 'stringa' è un gruppo di caratteri; ad esempio, STRING105%). Terminate la 'stringa' con un ↑ Z (CTRL e Z contemporaneamente). Si veda "la copiatura di porzioni di files" in questo stesso capitolo. Questo è un modo conveniente per stampare una porzione di un file.

S stringa ↑ Z PIP incomincia a copiare dal dispositivo o dal file quando trova la 'stringa' di caratteri specificata. Terminate la stringa con un ↑ Z. Si veda "la copiatura di porzioni di file". Questo è un modo conveniente per stampare una porzione di un file incominciando in una certa posizione.

Tn Espande con gli spazi i caratteri di tabulazione, posizionandoli ad ogni colonna ennesima, durante il trasferimento dei files di testo.

Potete inserire una tabulazione in un file di testo usando ↑ I; questo parametro espande le tabulazioni rispetto alla loro solita quantità fissa di colonne (colonne verticali sullo schermo del terminale).

U Traduce tutti i caratteri minuscoli in caratteri maiuscoli durante la copiatura dei files di testo.

V PIP verifica che i dati siano stati copiati correttamente rileggendo la nuova copia del file (il file destinazione non può essere un dispositivo) e stampando un messaggio se la copiatura ha avuto successo. Si dovrebbe usare questo parametro ogni volta che si fa una copia di sicurezza importante.

Z Mette a zero il bit di parità sui caratteri ASCII di input. Usate questo parametro in particolare se inserite i caratteri dallo pseudo dispositivo INP:

Ecco alcuni esempi di espressioni PIP con parametri.

```
* LST: = SAMPLE.TXT [NT8P60] ↵
```

Quest'espressione manda il file SAMPLE.TXT al dispositivo di stampa (LST:), con i numeri di riga ('N'), le tabulazioni espanse ogni otto colonne ('T8') e il posizionamento a pagina nuova ogni 60 righe ('P60'). Il dispositivo PRN: assume automaticamente questi parametri; se il dispositivo di stampa era stato assegnato a PRN: l'esempio sopra poteva essere descritto:

```
*PRN: = SAMPLE.TXT ↵
```

Potete modificare le "assunzioni" (parametri di default) di PRN: fornendo i vostri parametri:

```
*PRN: = SAMPLE.TXT [P59] ↵
```

Questa espressione manda SAMPLE.TXT a PRN: con i soliti parametri di default (cioé NT8), ad *eccezione* del parametro P che è cambiato in 59 righe. Ecco un altro esempio:

```
*LPT: = PROG.ASM [NT8U] ↵
```

Questa espressione manda PROG.ASM al dispositivo di stampa con i numeri di riga ('N'), le tabulazioni ogni otto colonne ('T8'), e i caratteri minuscoli tradotti in maiuscoli ('U').

La copiatura di porzioni di files

Inevitabilmente, vi accadrà qualche volta di interrompere un lungo listato sulla stampante o per motivi accidentali (per aver premuto un carattere sulla tastiera) o a causa della stampante (fine della carta o altri problemi meccanici). Vorrete allora ricominciare la stampa nel punto in cui era stata interrotta, piuttosto che stampare di nuovo tutto il file. Questo è il più semplice esempio di un trasferimento parziale. PIP fornisce la comoda opzione di stampare e trasferire porzioni di files.

Potete istruire PIP a copiare soltanto *porzioni* di files di testo specificando stringhe di caratteri di partenza e interruzione. (Ricordate che una "stringa" è una sequenza di caratteri). Usate il parametro S per specificare una stringa di partenza (cioé, dove PIP deve incominciare a copiare). E il parametro Q per specificare una stringa di interruzione (cioé dove PIP deve terminare di copiare). PIP cercherà automaticamente ciascuna stringa di caratteri.

Dovete terminare entrambe le stringhe con il carattere ↑Z (premendo CTRL e Z contemporaneamente). Ecco un esempio che copia il file SAMPLE.TXT dall'inizio alla stringa 'Extra':

```
A > PIP ↵
* NEWSAMPLE.TXT = SAMPLE.TXT [QExtra ↑Z] ↵
* ↵
A >
```

Quest'operazione di PIP termina quando incontra la stringa 'Extra'. Si noti che 'Extra' è in caratteri maiuscoli e minuscoli e che eseguiamo PIP come un programma, non come un comando di una riga. Se avessimo digitato:

```
A > PIP NEWSAMPLE.TXT = SAMPLE.TXT [QExtra ↑ Z] ↵
```

PIP avrebbe tradotto automaticamente la stringa 'Extra' in 'EXTRA'. SE ESEGUITE PIP COME UN COMANDO DI UNA RIGA, TRADURRA' SEMPRE AUTOMATICAMENTE LE STRINGHE IN CARATTERI MAIUSCOLI. Se eseguite PIP come un programma e digitate un'espressione di PIP, la stringa rimarrà come l'avete digitata.

Ecco un altro esempio dei parametri S e Q:

```
A > PIP ↵
* EXTRA.TXT = SAMPLE.TXT [SExtra ↑ZQUn altro extra ↑Z]↵
* ↵
A >
```

Questa operazione di PIP incomincia a copiare SAMPLE.TXT quando trova la stringa 'Extra' e *termina* di copiarlo quando trova la stringa 'Un altro extra'. Il file EXTRA.TXT contiene la porzione del file compresa tra le stringhe 'Extra' e 'Un altro extra', compresa 'extra' ma non 'Un altro extra'.

Si noti che abbiamo eseguito PIP come un programma per preservare i caratteri minuscoli delle stringhe. SE LE STRINGHE FOSSERO EFFETTIVAMENTE TUTTE MIAUSCOLE NEL FILE, allora le stringhe in caratteri maiuscoli e minuscoli non sarebbero state trovate da PIP.

MIGLIORAMENTI NEL CP/M VERSIONE 2.2

Miglioramenti

Se avete il CP/M versione 2.2 ci sono certe restrizioni che vi impediscono di realizzare operazioni di copiatura che sono "normali" nel CP/M versione 1.4. Per esempio, se fate uso delle *aree utente* della versione 2.2, vi sarete già accorti che non potete creare una copia in un'altra area utente, e neppure potete fare una copia di un file che è in un'altra area utente.

I miglioramenti a PIP vi permettono di "aggirare" gli *altri* miglioramenti al resto del CP/M versione 2.2 e dell'MP/M. La sezione sul CP/M versione 2.2 e sull'MP/M spiega le aree utente e gli attributi dei files, ma qui c'é un rapido sommario:

Area utente: ogni disco (dischetto) può mantenere files separati in aree utente distinte (cioé, utente 0, utente 1, utente 2, ecc.). Naturalmente, il vostro disco (dischetto) può avere soltanto un'area utente (utente 0) in modo da essere compatibile con le versioni precedenti del CP/M, ma la versione 2.2 vi permette di separare i files in aree utente e di saltare da un'area utente all'altra usando il comando USER, al fine di *prepararvi* all'MPM (cosicché i vostri dischi e dischetti saranno compatibili con le versioni future dell'MP/M). L'MP/M è un sistema *multi-utente* ed è necessario separare i files degli utenti. Non avete bisogno di usare questa

possibilità e non è raccomandata a meno che il sistema abbia molti utenti contemporaneamente.

Attributi dei files: nel CP/M versione 2.2 e nell'MP/M potete scegliere di mettere un indicatore speciale su un file, chiamato *attributo del file*. Gli attributi dei files regolano l'uso del file e comprendono: sola lettura, lettura - scrittura, sistema e direttrice. Questi attributi sono assegnati con il comando STAT (descritto nella sezione sul CP/M versione 2.2 e sull'MP/M).

Se un file è a sola lettura (abbreviato 'R/O'), non può essere *modificato* (aggiornato) o *cancellato*; cioé, il sistema non può scrivere nel file (o scrivere sopra il file). Dovete prima cambiare l'attributo del file R/O in R/W (lettura-scrittura) usando STAT.

Se un file è di "sistema" (abbreviato '$SYS'), *non* è stampato dal comando DIR e non potete leggere nel file (il che significa anche che non potete copiarlo). Se assegnate anche l'attributo R/O (cioé $SYS e R/O), è ben protetto. Potete modificare l'attributo dei file $SYS per mezzo di STAT nell'attributo '$DIR' (cioé "direttrice"), cosicché si possa accedere al file in modo regolare. Se assegnate anche l'attributo R/O, rimane attivo fino a quando lo cambiate in R/W. Gli attributi dei files sono trattati nella sezione che riguarda il CP/M versione 2.2 e l'MP/M (Capitolo 2).

PIP ha molte nuove caratteristiche per "aggirare" questi attributi dei files e per trasferire i files da un'area a un'altra. Queste caratteristiche sono in forma di parametri:

Gn Prende un file dall'area utente 'n' dove 'n' può variare tra 0 e 15.

W Scrive sopra (cancella) i files a sola lettura (ignora l'attributo R/O)

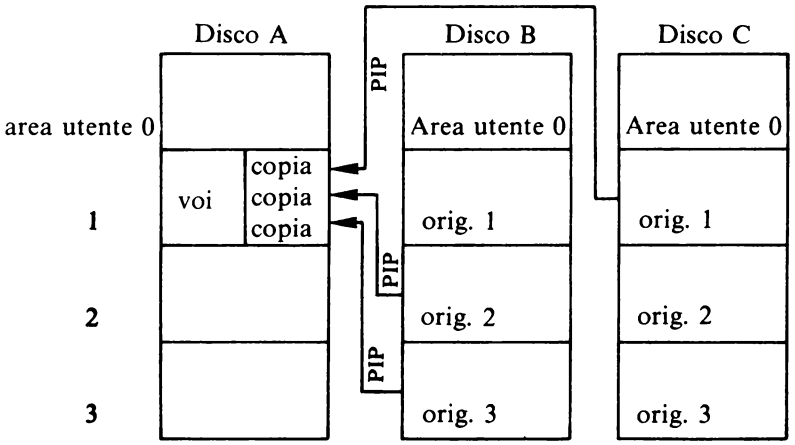
R Legge (copia) i files di sistema (files con l'attributo $SYS). Questo parametro consente anche un W (ignora l'attributo R/O)

Copiare dalle aree utente

Usate il parametro G per copiare un file che si trova in un'altra area utente. (Ricordate che non potete creare una copia in un'altra area utente, soltanto nella vostra). Non occorre usare questo parametro per copiare un file da un disco (dischetto) a un altro se il file da copiare (e la copia da creare) hanno lo stesso numero di area utente. Per esempio, se ORIG esiste nell'area utente 2 sull'unità A, potete copiarlo nel nuovo file COPY nell'area utente 2 sull'unità B. Dall'area utente 2, potete copiare qualsiasi file in qualsiasi altra area utente usando il parametro G, ma la nuova copia può essere creata soltanto nell'*area utente corrente* (cioé

l'area utente in cui siete correntemente).

Potete cambiare l'area utente corrente con il comando USER, descritto in dettaglio nella sezione sul CP/M versione 2.2 e sull'MP/M.



Ecco un esempio di un'espressione PIP che usa il parametro G:

```
*A: = B:JMP.TXT [G3] ↵
* ↵
2A >
```

L'espressione PIP copia il file JIM.TXT sull'unità B nell'area utente 3, portandolo nell'unità A nell'area utente corrente. L'area utente corrente è la 2, come è indicato dal promt di sistema (dopo che il programma PIP è stato terminato da un RETURN).

NOTA: per conservare la compatibilità con le versioni precedenti del CP/M e *anche* con le nuove future versioni del CP/M e dell'MP/M, usate soltanto l'area utente 0. Questa è l'area utente di "default" che non sarà modificata nelle nuove versioni e in quelle future e non causa neppure problemi con le versioni precedenti.

Se usate altre aree utente, dovete prima copiare PIP.COM in ciascuna area utente in cui si devono copiare dei files. Quando avete una copia di PIP.COM in ciascuna area utente di un'unità a dischi, potete facilmente chiamare il programma PIP da un'altra unità specificando la lettera dell'unità. Se PIP.COM non esiste nella vostra area utente corrente di

almeno un'unità a dischi attiva, allora non potete eseguire il comando PIP. Per copiare inizialmente PIP nelle aree utente, usate questa sequenza di comandi che comprende DDT (debugger dinamico) e SAVE (entrambe i comandi sono descritti nel Capitolo 2).

A > USER 0 ↵	(Specifica l'area utente 0-cambia dall'area utente 2).
A > DDT PIP.COM ↵	(Esegue DDT su PIP.COM-debugger).
DDT VERS. xx.xx	(Messagio di presenza del DDT).
NEXT PC 1C80 xxxxxx	(DDT stampa l'indirizzo successivo dopo la fine di PIP.COM. Usate questo indirizzo per calcolare il numero di pagine da usare con SAVE).
-GO ↵XXX	
A > USER 3 ↵	(Cambia l'area utente in 3, dove volete mettere una nuova copia di PIP.COM).
A > SAVE 28 PIP.COM	(SAVE crea un nuovo file, PIP.COM nell'area utente 3 dell'unità a dischi A, con 28 pagine di memoria - uguale al file PIP.COM originale. Ricavate '28' dal valore esadecimale '1C' dove 'C' è uguale a 12 e '1C' è uguale a 28. 1C è il "byte di ordine superiore" del valore esadecimale sotto parola NEXT nella stampa di DDT, 1C80).

Files a sola lettura e files di sistema

PIP non scrive sopra (cancella e ricerca) un file che ha l'attributo di sola lettura 'R/O'. Se tentate, il programma PIP risponde con una domanda:

```
A > PIP B:COPY = ORIG ↵
DESTINATION FILE IS R/O, DELETE (Y/N)?Y
A >
```

In questo esempio, abbiamo tentato di fare una copia di ORIG chiamata COPY ma COPY (il vecchio COPY) esiste già sull'unità B con l'attributo R/O (se non avesse avuto l'attributo R/O sarebbe stato cancellato da PIP e sostituito con il nuovo COPY). PIP stampa il messaggio che COPY (il file destinazione) è a sola lettura e chiede se vogliamo cancellare il vecchio COPY ('Y/N' significa "si" o "no"). Rispondiamo 'Y' per cancellare il vecchio COPY e sostituirlo con il nuovo COPY. (Quando rispondete 'Y' o 'N' non dovete premere RETURN (⏎).)
NOTA: il nuovo COPY *non* ha l'attributo R/O assegnato automaticamente.

Se avessimo risposto 'N' per "no", il vecchio COPY *non* sarebbe stato cancellato e PIP avrebbe stampato il messaggio:

```
**      NOT DELETED      **
```

Se volete *evitare* questa azione di PIP di stampare il messaggio se il file è R/O e di chiedere la verifica, potete usare il parametro W. Il parametro W dice a PIP di ignorare l'attributo di R/O. Potete usare il parametro W alla fine di un'espressione PIP se volete che si applichi a tutti i files in una concatenazione di files:

```
A > PIP WHOLE.TXT = PART1.TXT, PART2.TXT,
PART3.TXT [W] ⏎
A >
```

Se il file originale o il file di destinazione ha l'attributo $SYS (sistema), allora PIP non può trovare i files nella direttrice del disco. Potete usare il parametro R per ignorare gli attributi $SYS (e R/O) in modo che PIP può trovare il file originale o creare il file destinazione. Usate l'attributo R nello stesso modo in cui è stato usato sopra l'attributo W.

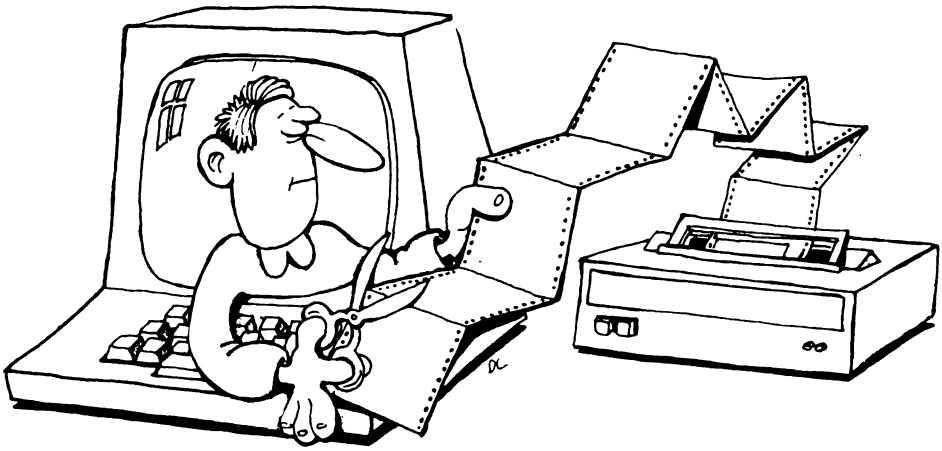
SOMMARIO

PIP è un potente programma di trasferimento di files ad uso generale. Sebbene molti utenti del CP/M lo usino soltanto per trasferimenti da disco a disco, può fare molto di più:

- Trasferire tra dispositivi e dischi.
- Trasferire più files.
- Elaborare, verificare e dare un formato ai files.

PIP può essere usato in modo vantaggioso per ottenere porzioni di un file o per congiungere più files. Può anche essere usato per ottenere

stampe pulite, impaginate o con tabulazioni. Una conoscenza pratica dei parametri più importanti è molto vantaggiosa. Ogni utente del CP/M dovrebbe pertanto leggere questo capitolo completamente almeno una volta, poi rileggere di nuovo in dettaglio le opzioni che usa più frequentemente.



CAPITOLO 4

L'USO DELL'EDITOR

INTRODUZIONE

I Capitoli 1 e 2 vi hanno insegnato tutto ciò che dovete sapere per incominciare ad usare il CP/M. Il Capitolo 3 ha descritto il più importante programma di utilità, PIP. Questo capitolo vi insegnerà come usare un altro importante programma applicativo che viene fornito con il sistema operativo CP/M: l'editor, ED. Vi si mostrerà come usare un effettivo programma editor e seguirete i trasferimenti di dati tra i dischi, la memoria del calcolatore e il terminale.

Non è importante ricordare i comandi specifici forniti da ED. Questi comandi sono riassunti nell'appendice. Ciò che è importante, invece, è capire come funziona un editor e che cosa può fare. Se raggiungete questo obiettivo, troverete semplice da capire la maggior parte degli altri programmi applicativi (se sono stati ben progettati e documentati). Inoltre, probabilmente sarete capaci di usare facilmente un programma di word precessor (trattamento di testi), un programma applicativo molto utile su ogni calcolatore.

Questo capitolo è utile ma indispensabile. Se ritenete di non essere interessati a conoscere l'editor, potete passare al capitolo successivo.

CHE COS'È UN PROGRAMMA EDITOR?

Un buon programma editor vi permette di creare e modificare files di testo - lettere, romanzi, poesie, corrispondenze commerciali o qualsiasi cosa sia formata da caratteri. Vi permette, inoltre, di passare facilmente da una riga all'altra e di modificare i caratteri riscrivendoli, o cancellandoli e inserendone di nuovi. È in grado di trovare in un file qualsiasi gruppo di caratteri specifichiate, e di fare sostituzioni. Un buon programma editor vi permette di fondere due files e di mescolare righe di testo di due files.

Quando digitate un tasto usando un buon programma editor, digitate una riga e la terminate con un RITORNO CARRELLO (RETURN o CR su alcune tastiere), proprio come fareste su una macchina da scrivere

elettrica. In futuro, un programma editor potrebbe anche funzionare in modo più semplice; certamente non abbiamo ancora visto il miglior programma editor né il più semplice da usare.

ED, il programma editor fornito dal CP/M, è soltanto un editor a prestazioni minime e non è così semplice da usare come la maggiore parte dei programmi editor. Se ritenete di dover fare grande quantità di digitazione o di trattamento di testi, dovreste procurarvi un editor più potente. Ci sono molti editor e "word processor" sul mercato oggi che funzionano con il CP/M o l'MP/M.

Un *word processor* è un programma che comprende sia un programma editor (per digitare i testi) che un programma che gestisce la stampante (per stampare i testi), facendo le sottolieneature, i margini allineati, l'espansione delle tabulazioni e la stampa in neretto. Fate attenzione, comunque, a quello che comprate esattamente. Ci sono alcuni cosiddetti "word processor" che sono soltanto programmi di stampa progettati per essere usati con il programma ED del CP/M. Questi sono soltanto programmi di impaginazione e non sono convenienti e sufficientemente potenti per un uso generale come word processor.

Considerate bene i prodotti come se doveste comprare una macchina da scrivere. Potreste volere campanelli e fischi; oppure potreste volere un modello portatile ed economico che richiede più lavoro ma svolge il suo compito. Comunque, la semplicità di uso dovrebbe essere la considerazione primaria (specialmente se dovete usarlo per scrivere). Dopo aver letto questo capitolo saprete quale dovrebbe essere il minimo insieme di possibilità fornito da un editor. ED è sufficiente per la maggior parte delle applicazioni semplici.

ED, L'EDITOR

Il programma editor ED.COM solitamente risiede sul dischetto di sistema ed è eseguito digitando "ED",seguito dal nome del file di testo che create o modificate. Per esempio, se volete creare o modificare il file SAMPLE.TXT, digitate:

```
A > ED SAMPLE.TXT ↵
```

Non digitate soltanto 'ED'. Questo è un errore comune e non funziona. Dovete fornire un nome di file.

NOTA: se ottenete il messaggio 'FILE IS READ/ONLY', o 'SYSTEM FILE NOT ACCESSIBLE', allora dovete usare prima il comando STAT sul file (CP/M versione 2.2 e MP/M). Si veda la sezione sul CP/M versione 2.2 e sull'MP/M nel Capitolo 2.

Se ED non trova il file specificato, assume che vogliate creare un

nuovo file. Questo file, specificato nel comando, diventa il file "sorgente". Successivi comandi di ED porteranno il testo del file sorgente nel *buffer di edit*, (Fig. 4-1).

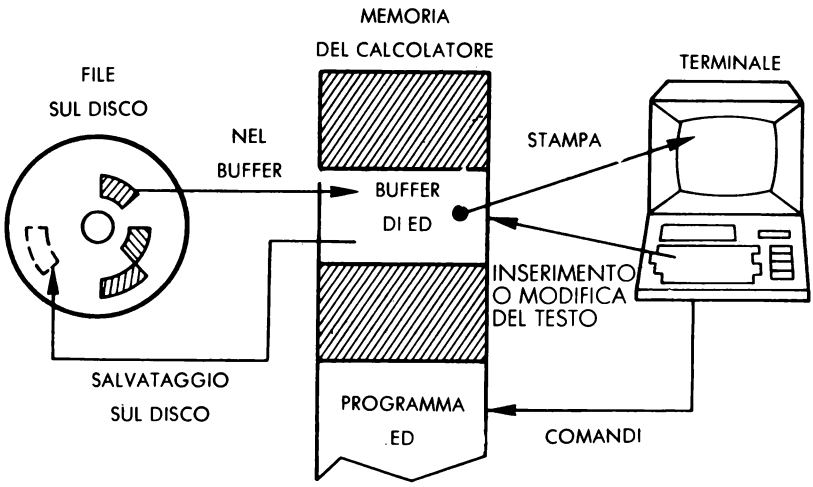


Figura 4.1: Il buffer di ED

Il *buffer* è un blocco di memoria nel calcolatore riservata per la gestione del tasto ED. Se il file di testo è grande, potete caricarne soltanto un blocco per volta nel buffer. (Si noti che questo è un inconveniente inerente a ED che non si presenta in editor più potenti).

Potete soltanto digitare "nuovo testo" nel buffer di edit o modificare il testo che è già presente. Inoltre, il buffer non viene ricopiato sul disco automaticamente. Se terminate ED (o spegnete il sistema) senza *salvare* il testo del buffer in un file su disco, perdete il testo nel buffer. Poiché ED *copia* il file di testo nel buffer di edit, il file originale (cioé prima di usare ED) non è stato modificato ma il nuovo testo e le modifiche vengono perse. Pertanto, dovreste salvare periodicamente il testo nel buffer di edit e ricordare sempre di salvare il testo prima di uscire dal programma ED. Potete salvare il testo usando il comando 'E' di ED (E ↵). Il comando E copia anche il resto del file sorgente nel nuovo file "sorgente" che crea. (Questo è spiegato con maggior dettaglio più avanti).

La maggior parte dei comandi di ED consistono in lettere speciali precedute da un numero o da un simbolo che stabilisce una quantità. Questi comandi sono eseguiti digitandoli come i comandi del CP/M: il

comando, seguito da un RETURN (↵), che trasmette il comando. Altri "comandi" sono speciali combinazioni di tasti (come CTRL e Z (↑ Z), CTRL e C (↑ Z), CTRL e C (↑ C), ecc. che sono trasmessi automaticamente e non richiedono un RETURN.

Il programma ED stampa sempre il prompt di ED:

```
*
```

Il comando E sarebbe digitato in questo modo:

```
*E ↵
```

Molti comandi agiscono sul testo del buffer di edit, mentre altri trasferiscono il testo nel buffer e dal buffer. Il buffer di edit è mostrato in Fig. 4.2:

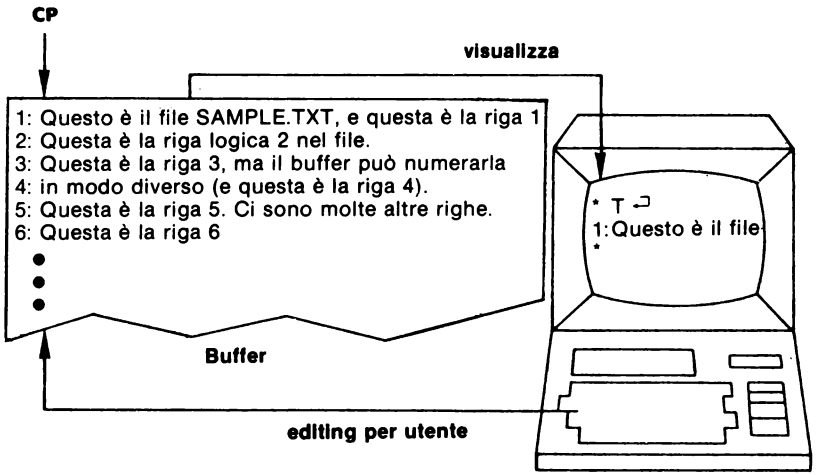


Figura 4.2: Operazioni su un testo

IL 'CP' (CHARACTER POINTER - PUNTATORE DI CARATTERI) E I NUMERI DI RIGA

Il 'CP' nell'illustrazione è il "puntatore di caratteri". Non si vede effettivamente sullo schermo, ma è usato dal programma ED ed è

spostato da comandi di ED. Il puntatore punta sempre a un carattere e i comandi di ED solitamente si riferiscono ai caratteri successivi (e comprendenti) il carattere puntato dal CP, o ai caratteri che spostano il CP. In altre parole, dovete spostare il CP a *destra* per andare avanti nel testo e a *sinistra* per andare indietro. Esempi illustrati sono presentati nella sezione seguente di questo capitolo.

Oltre al CP immaginario, ogni riga ha un *numero di riga* immaginario che non fa parte effettivamente del testo. Se avete l'ultima versione di ED, questa automaticamente stampa il numero di riga con il testo (e potete spostarvi ad ogni riga del testo specificando il numero di riga come comando, come mostrato più avanti). Se avete le versioni precedenti di ED, dovete *attivare* la stampa dei numeri di riga eseguendo il comando V (il comando -V (V negativo) la disattiva). Se avete versioni ancora *più vecchie*, potreste non disporre affatto dei numeri di riga (il che è una sfortuna, poiché rendono più semplice spostarsi nel buffer di edit). I numeri di riga, come il CP, esistono soltanto nel buffer di edit e sono usati soltanto per spostarsi nel buffer. Non compaiono in una stampa del file.

CHE COSA FA ED AL FILE DI TESTO

Per esempio, assumiamo che il file SAMPLE.TXT esista già e contenga il testo riportato in Fig. 4.3

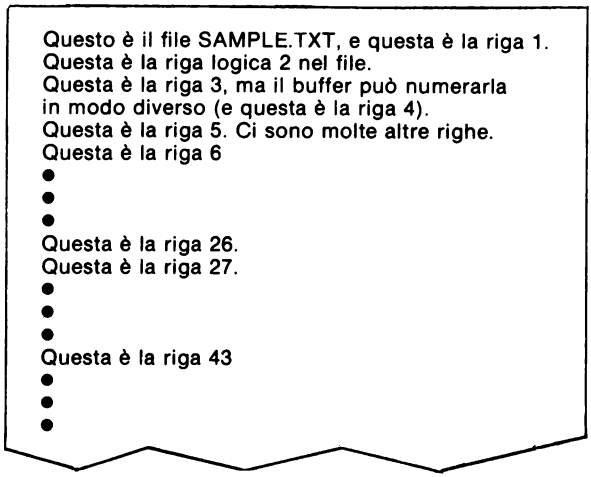


Figura 4.3: Un file "Esempio" nel buffer

Quando eseguite il comando CP/M:

A > ED SAMPLE.TXT ↵

ED mette da parte uno spazio nell'*area dei programmi transitori* (memoria tampone o "scratch-pad" - sarà spiegata nel Capitolo 5) per il buffer di edit, crea un file di uscita temporaneo chiamato SAMPLE.$$$ (per memorizzare la nuova versione del file senza danneggiare quella originale) e prepara SAMPLE.TXT per copiarlo nel buffer di edit. La copiatura è eseguita effettivamente per mezzo del comando A (append) - questo comando "appende" un certo numero di righe nel buffer di edit, come mostrato in Fig. 4.4:

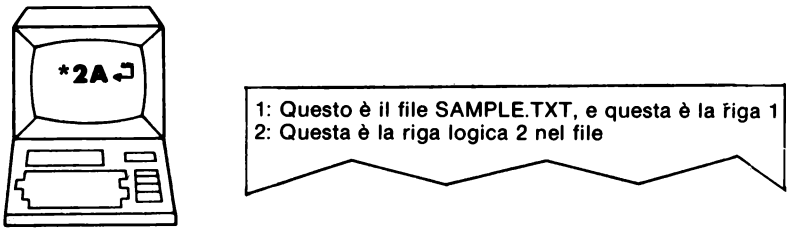


Figura 4.4: Il comando Append è in A

Il comando di ED '2A' appende le prime due righe del file di testo nel buffer. Se volete appendere altre righe, dovete dare un altro comando A (Fig. 4.5).

Ora potete modificare queste righe che si trovano nel buffer di edit. Potete anche aggiungere nuove righe dalla tastiera (usando il comando I, descritto più avanti in questo capitolo). Questo è mostrato in Fig. 4.6.

Avete ora imparato come inserire delle righe nel buffer dal file su disco o dalla tastiera. È mostrato in Fig. 4.7.

Potreste avere appeso l'intero file nel buffer di edit (a meno che il file abbia più di 6000 caratteri e non usiate una versione 16K del CP/M), ma per semplicità, questo esempio manda il testo appena modificato (con le nuove aggiunte) a un file *temporaneo di uscita* in modo che possiate appendere altro testo al buffer di edit. ED chiama SAMPLE.$$$ questo file temporaneo di uscita. Per mandare il testo appena modificato al file temporaneo di uscita, usate il comando W (scrivi) oppure terminate "normalmente" con il comando E (fine edit) o il comando H (fine e riapertura). La Fig. 4.8 mostra l'uso del comando W.

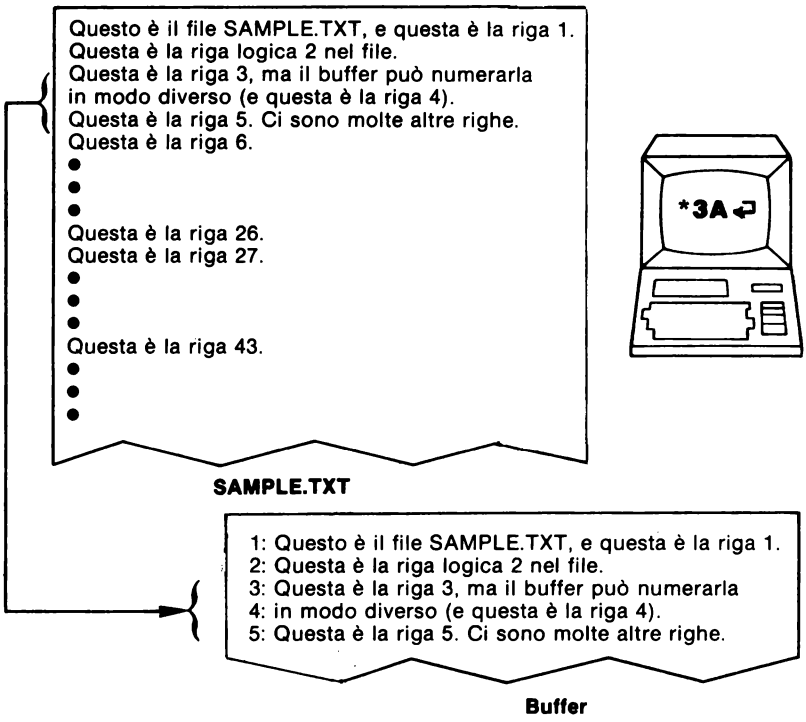


Figura 4.5: Altre tre righe appese

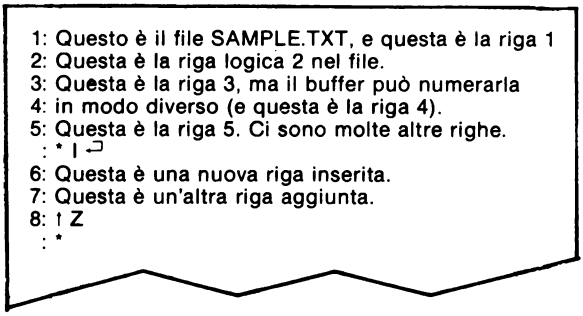


Figura 4.6: Due nuove righe aggiunte dalla tastiera

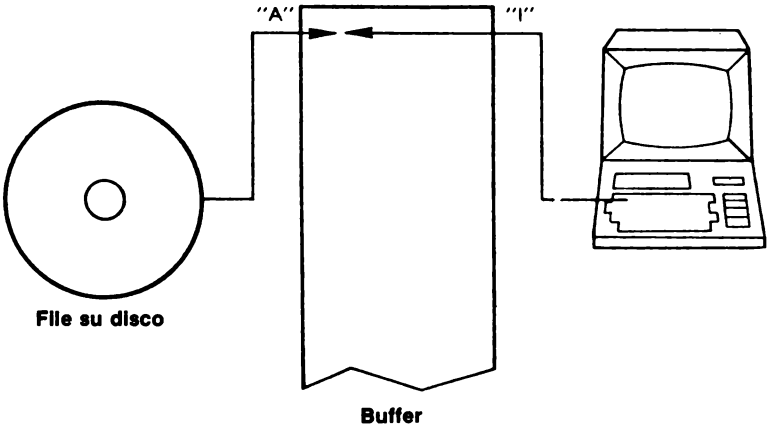


Figura 4.7: Righe inserite nel buffer

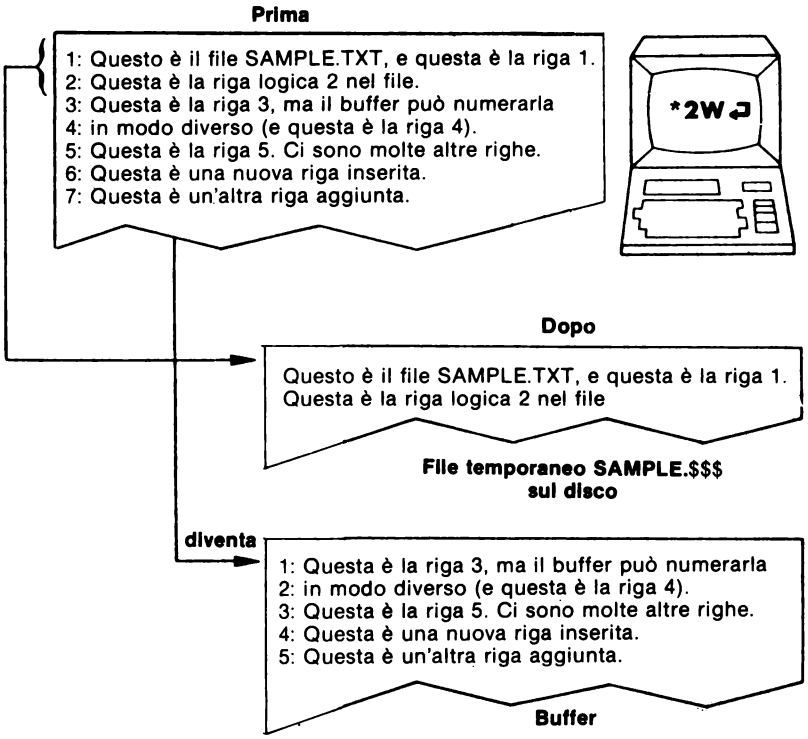


Figura 4.8: Salvataggio del buffer sul disco

L'esempio in Fig. 4.8 mostra un'operazione di scrittura di due righe. Le righe nel buffer finale di edit si spostano verso l'inizio, lasciando altro spazio per aggiungere nuove righe. L'esempio successivo mostra un'operazione in cui si appendono le 20 righe successive. Dopo aver modificato il testo nel buffer, potreste terminare la sessione di lavoro con ED (scrivere il resto del buffer nel file di uscita) usando il comando E, come mostrato nell'esempio in Fig. 4.9. Il comando E copia anche il resto del file sorgente - le righe non appese al buffer nel file temporaneo di uscita. Si noti che il file temporaneo di uscita contiene le righe del testo nel loro ordine corretto.

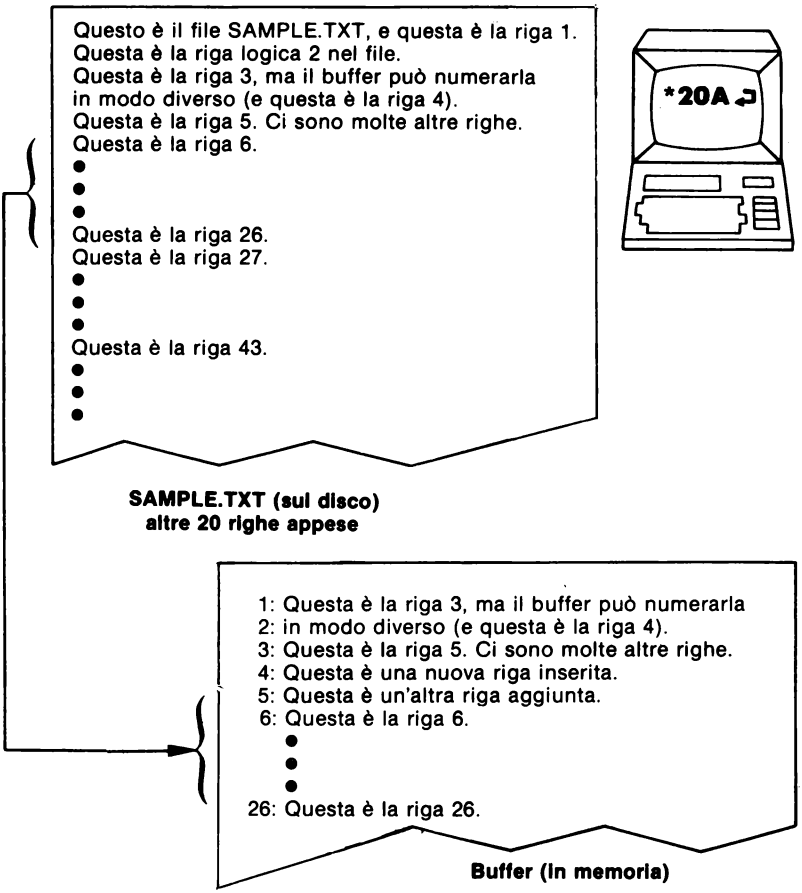


Figura 4.9: 20 righe appese al buffer

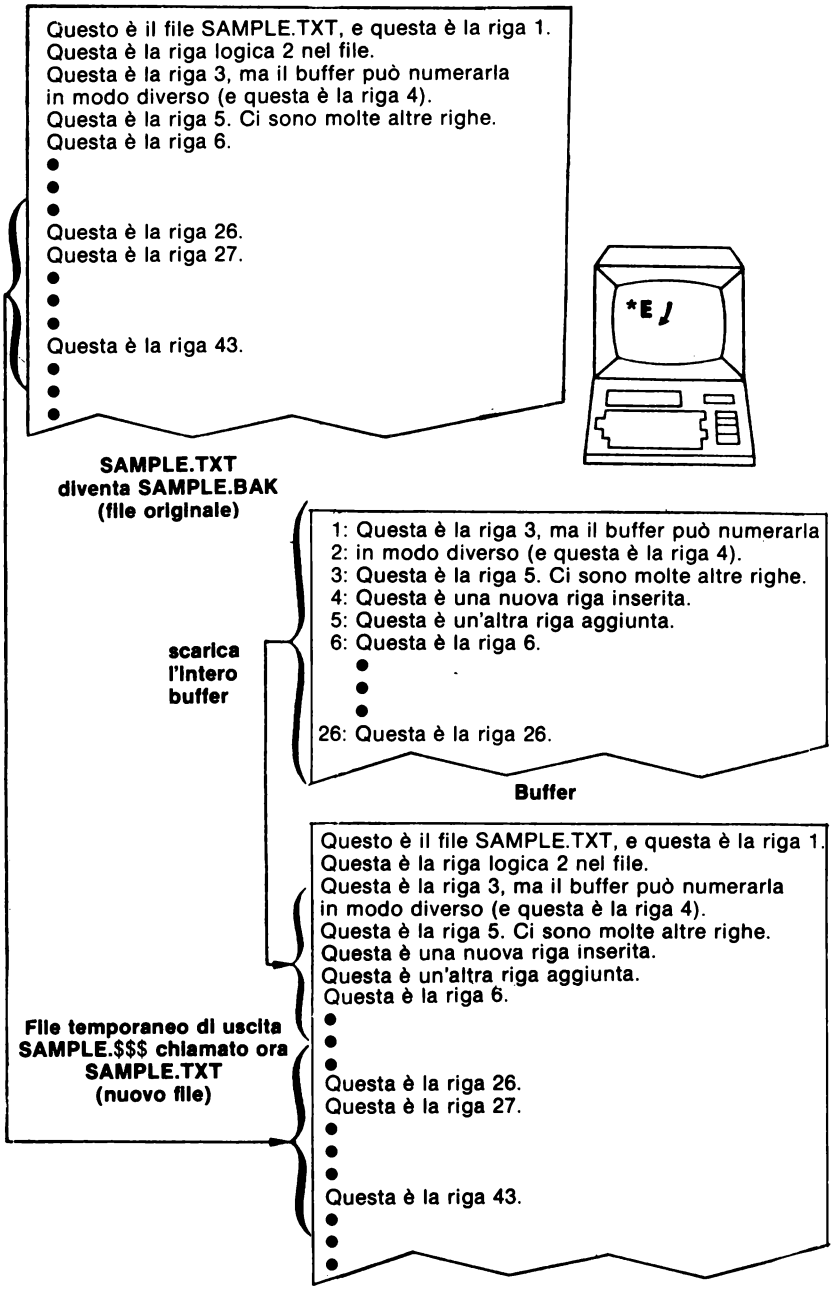


Figura 4.10: Completamento di una sessione di edit

Non appena il buffer di edit è stato copiato correttamente nel file temporaneo di uscita, che si chiama SAMPLE.$$$, ED fa due cose: cambia il nome del file originale SAMPLE.TXT in SAMPLE.BAK e cambia il nome di SAMPLE.$$$ in SAMPLE.TXT. Facendo questo ED crea una copia di sicurezza (backup) del file originale SAMPLE.TXT (chiamata SAMPLE.BAK) e cambia il nome del nuovo file modificato SAMPLE.$$$ in SAMPLE.TXT. (Fig. 4.10). Questo è ciò che appare nelle modifiche fatte da ED sul file SAMPLE.TXT in effeti, ED modifica il testo nel buffer di edit e usa SAMPLE.TXT come backup e SAMPLE.$$$ come versione futura di SAMPLE.TXT.
NOTA: quando eseguite ED su un file, ED automaticamente cancella ogni file 'BAK' associato al file di testo (in vista del nuovo file 'BAK' che ED creerà). I comandi 'O' e 'Q' non evitano questa azione, pertanto fate attenzione.

GESTIONE DEI FILES

Quando passate a ED un nome di file, questo lo cerca; se non lo trova, lo crea. Questo file è chiamato, "sorgente". Nella documentazione del CP/M. Quando appendete testo al buffer di edit dal file sorgente (usando il comando A), ED copia le righe di testo dall'inizio del file nel buffer, contando il numero di righe nel file temporaneo di uscita creato da ED (nomefile.$$$), liberate spazio nel buffer per appendere altre righe di testo dal file sorgente. Quando scrivete nel file di uscita, le righe vengono appese in modo tale da rimanere nello stesso ordine (potete scrivere righe specifiche nel file di uscita per cambiare quest'ordine). Quando finite il lavoro con l'editor o terminate ED usando il comando E (o H), ED automaticamente cambia il nome del file originale (file sorgente) in un file con l'estensione BAK (ad esempio SAMPLE.BAK) per indicare che è un file di backup, e cambia il nome del file temporaneo di uscita (ad es. SAMPLE.$$$) nel nome del file originale (sorgente) (ad es. SAMPLE.TXT), cosicché il file di testo contiene il file appena modificato.

TERMINAZIONE ACCIDENTALE

Se terminate ED accidentalmente, senza usare il comando E (o H) (cioé, c'é un errore di sistema, o inavvertitamente premete ↑ C per rilanciare il sistema, o è caduta la tensione), il file temporaneo di uscita (ad esempio SAMPLE.$$$) rimarrà con quel nome di file e il file sorgente originale (ad esempio SAMPLE.TXT) sarà ancora la vecchia copia. Il file $$$ conterrà soltanto il testo che avete già in esso (così potrebbe non essere utile) il buffer di edit sarà perso e il file originale sarà la versione non modificata (cioé, prima di chiamare ED).

NOTA: USATE IL COMANDO Q (quit) per far questo di proposito.

Potete inserire righe di testo di un altro file di testo con il testo già nel buffer di edit usando il comando R, trattato più avanti (l'altro file non è modificato in alcun modo).

UNA SESSIONE DI LAVORO CON L'EDITOR

Per creare un nuovo file di testo, prima pensate a un nome (un nome che non sia già usato con un'altra estensione). Useremo QUOTE.TXT come esempio. Eseguite il comando ED e create QUOTE.TXT digitando:

```
A > ED QUOTE.TXT ↵
```

In questo esempio, siamo nell'unità A; pertanto, QUOTE.TXT sarà nell'unità A. Siamo nell'unità A perché ED (ED.COM) è nell'unità A. L'esempio nel Capitolo 1 vi mostrava come eseguire ED da un'altra unità premettendo la lettura dell'unità e il carattere "due ponti" prima del nome di file ED. Potete anche eseguire ED dall'unità A e mettere QUOTE.TXT sulla unità B premettendo 'B' a 'QUOTE.TXT'.

ED stamperà il messaggio 'NEWFILE' se sta creando un nuovo file. Quando ED è pronto per un comando, stampa il prompt di ED '*'. Potete ora digitare qualsiasi comando di ED. Per inserire nuovo testo dalla tastiera, dovete usare il comando I. Il comando I incomincia a inserire il testo dopo il CP (puntatore di carattere). Poiché non avete spostato in modo esplicito il CP (con un comando di ED), il CP è all'inizio del buffer. Per incominciare a inserirlo nel nuovo testo, digitate:

```
*I ↵
```

Per inserire effettivamente il testo, digitate il testo come fareste su una macchina da scrivere. Per terminare di inserire il testo, premete i tasti CTRL e Z (↑ Z).

Quando digitate 'I ↵' ED porta il cursore (puntatore lampeggiante sullo schermo o qualsiasi simbolo usi il vostro terminale) alla riga vuota successiva. Potete digitare ora il testo, che sarà inserito automaticamente nel buffer di edit quando trasmettete ciascuna linea. *Ciascuna linea deve*

terminare con un RETURN (come una macchina da scrivere) - RETURN (rappresentato da un ↵ in questo libro) trasmette la riga al buffer di edit. La riga allora diventa una *riga del buffer* (identificata da un numero di riga). Se volete digitare "una riga del buffer" che è in effetti più lunga di una riga che voi potete digitare sul vostro terminale, usate la combinazione di CTRL e E (↑ E) per spostare il cursore alla riga successiva nel vostro schermo senza trasmettere la riga al buffer di edit. Quando infine premete RETURN l'intera riga è trasmessa al buffer. Le righe non possono superare 128 caratteri.

Ricomincia l'esempio dall'inizio. Tenete presente che potete usare i tasti di controllo RUBOUT (DELETE), ↑ U (cancellazione della riga), ↑ E (ritorno del "carrello" senza trasmettere la riga), e ↑ R (ristampa dell'ultima riga) sia nelle vecchie versioni di ED che nelle nuove. Potete usare ↑ H (cursore indietro di una posizione con cancellazioni del carattere) e ↑ X (ritorno del cursore all'inizio della riga) soltanto nella nuova versione di ED.

Ecco ancora il nostro esempio dall'inizio:

```
A > ED QUOTE.TEXT ↵
NEW FILE
*I ↵
"Non abbiamo smesso di esplorare"
scrisse T.S. Eliot.
 ↑ Z
*
```

Usiamo la combinazione CTRL e Z per terminare di inserire il testo (↑ Z). Abbiamo ora due righe di testo - la prima è quela tra virgolette e la seconda è 'scrisse T.S. Eliot'.

Se volete, potete usare il comando U prima del comando I e qualsiasi cosa digitiate viene automaticamente convertita in maiuscolo (anche se quando digitate il testo questo appare in lettere maiuscole e minuscole. Per disattivare questa trasposizione in maiuscolo usate il comando -U (U negativo).

Ora che abbiamo il testo nel buffer, possiamo vederlo sullo schermo. Prima dobbiamo riportare il CP (puntatore di caratteri) all'inizio del

buffer, poiché il comando I inseriva il testo e spostàva il CP. Ecco un'illustrazione che mostra il testo e la posizione del CP nel buffer:

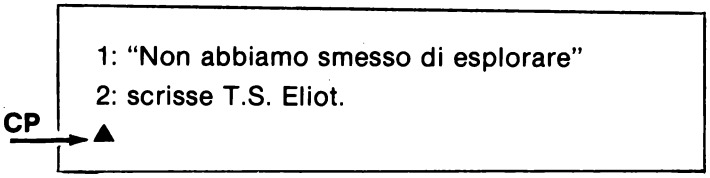


Figura 4.11: Il CP è alla fine del buffer

Il manuale di ED spiega che il CP è *tra* due caratteri. Questo è difficile da immaginare e da usare, così, abbiamo cambiato la descrizione per renderla più semplice. Il CP è un oggetto "immaginario" un puntatore di riferimento da usare con i comandi di ED. Quando vi riferite alla documentazione della Digital Research, tenete a mente che queste descrizioni sono basate sul CP tra due caratteri, mentre le nostre descrizioni sono basate sul CP che punta al carattere più a destra tra i due. L'illustrazione della Fig. 4.12 dovrebbe chiarire questo:

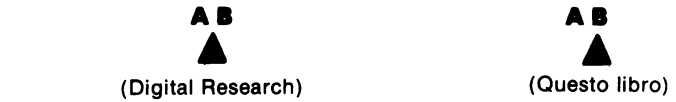


Figura 4.12: Rappresentazione della posizione del CP

La Digital Research dice che il CP è *prima* del primo carattere di un buffer quando lo si porta all'inizio e noi diciamo che il CP è *sul* primo carattere del buffer quando lo si porta all'inizio.

COME VISUALIZZARE IL TESTO NEL BUFFER

Useremo il comando T per visualizzare il testo nel buffer. Il formto del comando T è:

$$\pm \, nT$$

dove n può essere zero o qualsiasi numero, o il simbolo (#) (Fig. 4.13). Se n è zero, T visualizzerà la riga corrente fino al CP escluso (la riga corrente è la riga in cui si trova il CP).

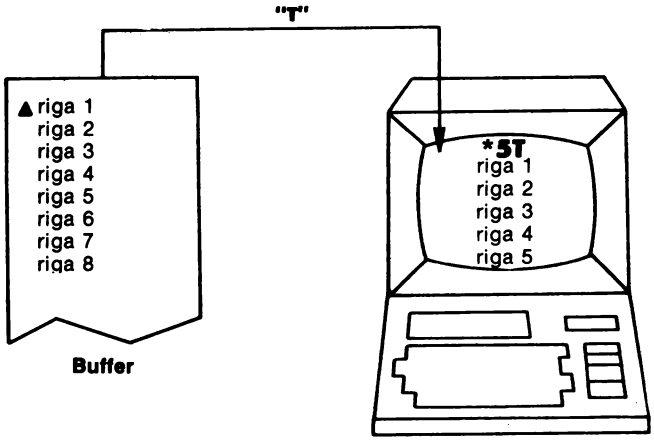


Figura 4.13: Visualizzazione del testo

Se non specificate n, si assume 1. Se n è 1 (o se è assunto essere 1), la riga corrente è visualizzata dal CP alla fine della riga. Se n è un numero positivo, T visualizzerà un numero n di righe dal CP (riga corrente) in avanti; se n è un numero negativo, T visualizzerà le righe prima del CP (non comprendendo il CP). Se usate il simbolo (#), T utilizzerà come numero '65535' (massimo numero di righe permesse nel buffer), cioé l'intero buffer sarà visualizzato. Potete stampare l'intero buffer spostando il CP all'inizio del buffer (usando il comando B, mostrato in Fig. 4.14) e usando il simbolo (#) con T; altrimenti un simbolo (#) stamperà tutte le righe *che seguono* il CP, (comprendendolo) e il simbolo (- #) stamperà tutte le righe che precedono il CP (escludendolo).

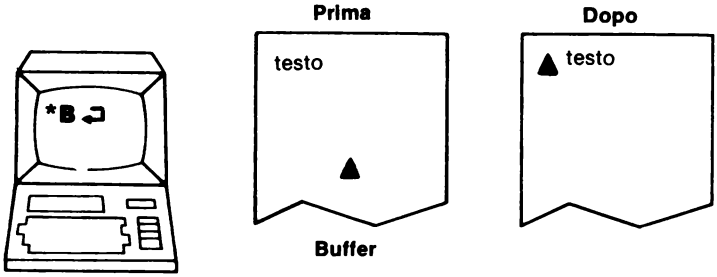
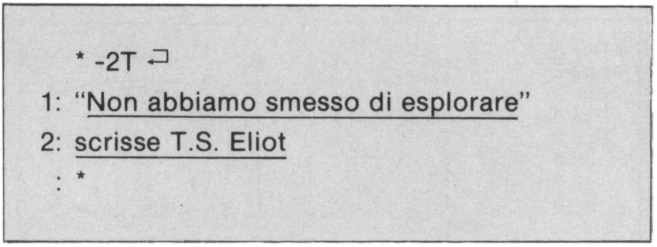
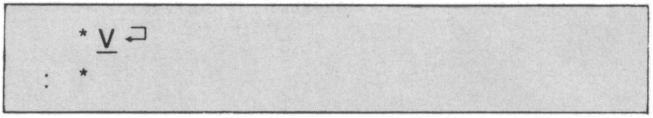


Figura 4.14: Spostamento del cursore

Ecco un esempio:



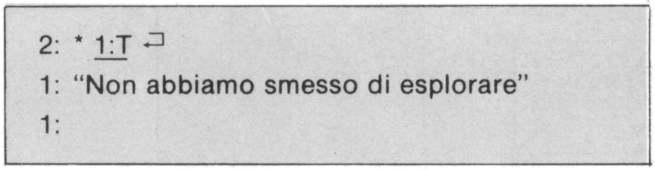
Se il vostro schermo non contiene i numeri di riga (numeri seguiti dal segno:), provate ad eseguire il comando V (versione 1.4 del CP/M o successive):



Il ':' vi dice che il CP è all'inizio della riga 3; comunque, la riga non ha ancora testo, cosicché il CP punta effettivamente alla fine della riga 2 (dopo la sequenza 'Ritorno Carrello').

Il comando '-2T' riportato sopra dice a ED di visualizzare soltanto le due righe che precedono quella contenente il CP (cioé la riga corrente che è la riga 3).

Nella versione 1.4 del CP/M o in quelle successive, potete specificare il numero effettivo di riga in un comando T per visualizzare quella riga. Per esempio, se volete proprio visualizzare la riga 1, potrete digitare '1:T' come comando:



Osservate che il comando '1:' con 'T' ha portato il CP alla riga 1 (la riga corrente ora è la riga 1). Potete specificare un numero di riga come

comando per portare il CP a quella riga. Per esempio:

```
1: * 2: ⏎
2: *
```

Un semplice comando T stamperà la riga corrente:

```
2: * T ⏎
2: scrisse T.S. Eliot.
2: *
```

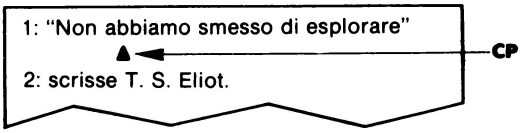
Un modo semplice per visualizzare l'intero buffer è eseguire due comandi: il comando B per portare il CP all'inizio del buffer e il comando T con il simbolo (#) per visualizzare l'intero buffer. Osservate che potete mettere entrambe i comandi sulla stessa riga di comando ed eseguirli:

```
1: * B#T ⏎
1: "Non abbiamo smesso di esplorare"
2: scrisse T.S. Eliot.
1: *
```

Nell'esempio sopra il CP puntava al primo carattere della riga. Potete spostare il CP nella riga corrente usando il comando C:

± nC

C porterà il CP + n caratteri avanti o - n caratteri indietro. Per esempio, se il CP è all'inizio della riga 1 e digitate questo comando:




```
1: * 5C ↵
*1: *
```

Il CP punterà al sesto carattere ('a', poiché le virgolette contano come un carattere) nella riga 1. Il comando 'T' visualizzerà la riga dal CP alla fine (compreso il CP):

```
1: * T ↵
abbiamo smesso di esplorare"
1: *
```

Il comando '0T' stamperà la riga 1 dall'inizio fino al CP escluso:

```
1: * 0T ↵
"Non
1: *
```

IL SALVATAGGIO DEL FILE ALLA FINE DELLA SESSIONE DI ED

A questo punto, dovrete imparare come salvare il contenuto del buffer e uscire da ED. Il modo più semplice per salvare e uscire da ED allo stesso tempo è di usare il comando E:

```
1: * E ↵
```

Ovunque siate nel buffer, questo comando vuota il buffer (il resto del file sorgente se non lo avete appeso per intero) nel file temporaneo di uscita e cambia il nome del file sorgente (originale). Se siete partiti con un file QUOTE.TXT vuoto e avete aggiunto il testo come indicato negli esempi precedenti, dovreste trovarvi un file QUOTE.TXT che contiene una riga di T.S. Eliot, e un file QUOTE.BAK vuoto (un backup del file prima di modificarlo).

COME APPENDERE RIGHE AL BUFFER (MODIFICHE DI UN FILE ESISTENTE)

Quando avete un file di testo esistente e eseguite ED per modificarlo, è necessario eseguire il comando A per portare le righe del testo nel buffer di edit. Altrimenti, non ci sarà testo nel buffer eccettuato quello che inserite usando il comando I di ED. Potreste voler inserire nuovo testo *prima* della prima riga del file di testo esistente - usando il comando I per inserire nuovo testo prima di appendere il testo del file esistente al buffer con il comando A; comunque, potete farlo facilmente *dopo* che il vecchio testo è stato portato nel buffer. Così, vorrete usare il comando A per vedere prima il vecchio testo. Il comando A aggiunge ("appende") le righe alla fine del buffer.

Il formato del comando A è:

nA

Il comando A appende n righe di testo dal file di testo originale (sorgente). Se non specificate n, A appende soltanto una riga. Se usate il simbolo # invece di n, A porterà l'intero file sorgente (fino a 65535 righe) nel buffer di edit. Poiché la maggior parte dei file di testo non sono così grandi, potete usare la formula '# A' per portare dentro al buffer di edit qualsiasi file che possa essere con probabilità contenuto in esso. Se specificate uno zero al posto di n, il comando A appende righe fino a riempire *metà* del buffer (utile per i files estesi). Usate la forma '0A' insieme a '0W' per appendere e scaricare metà del buffer (trattato in questo capitolo in "Lo scaricamento delle righe nel file").

Se appendete soltanto una porzione del file sorgente (originale) al buffer, il comando A *successivo* ricomincerà ad appendere dal file sorgente nel punto in cui aveva terminato la volta precedente. Potreste facilmente appendere 10 righe, cambiare qualcosa del testo, scaricare le 10 righe nel file temporaneo di uscita (con il comando W trattato più avanti) e appendere altre righe dal file sorgente. Oppure se avete abbastanza spazio nel buffer, potete appendere le altre righe del file sorgente

alla fine di quelle già esistenti nel buffer.

Ecco un semplice esempio, che usa QUOTE.TXT come file sorgente:

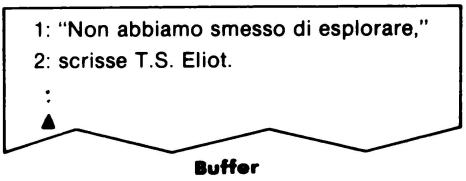
```
A > ED QUOTE.TXT ↵
 *V ↵        (se si usa la versione 1.4, non necessario
              nella versione 2.2)
 :*A ↵       (appende soltanto una riga)
1: *2A ↵     (appende le due righe successive, ma co-
              munque c'é soltanto più una riga nel file)
2: *B#T ↵    (va all'inizio e visualizza tutte le righe)
1: "Non abbiamo smesso di esplorare"
2: scrisse T.S. Eliot.
1: *
```

Poiché ci sono soltanto due righe in QUOTE.TXT, un semplice comando '# A' sarebbe sufficiente a portarle tutte nel buffer.

SPOSTAMENTI ALL'INTERNO DEL BUFFER

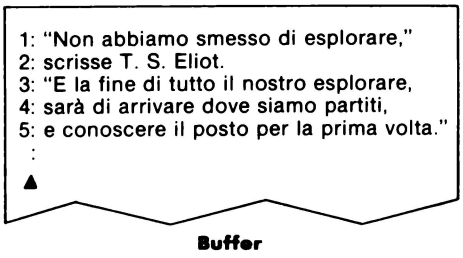
Con il testo nel buffer, potete spostarvi a volontà e cambiare qualsiasi testo così come inserire nuovo testo ovunque. Potete anche localizzare e sostituire pezzi di testo e appendere altre righe di testo da uno speciale file di *sorgenti di libreria* (trattato più avanti).

Se seguite gli esempi lavorando su QUOTE.TXT, dovreste adesso andare alla fine dél buffer e inserire del testo aggiuntivo. Il comando -B (B negativo) è usato per spostarsi alla fine dell'ultima riga come in questo esempio:




```
1: * -B ↵
 : *
```

La "fine dell'ultima riga" è in effetti un "ritorno carrello" che nel codice ASCII è una combinazione di RETURN e LINE FEED: due caratteri che realizzano l'operazione di far tornare il carrello e generare un avanzamento di riga. Pertanto, la "fine" dell'ultima riga è in effetti l'inizio della nuova riga successiva; quindi, il numero della nuova riga successiva non è visualizzato fino a quando non inserite dei caratteri usando il comando I. Potete inserire caratteri che formino la nuova riga. Per esempio:




```
:*I ⏎
3: "E la fine di tutto il nostro esplorare, ⏎
4: sarà di arrivare dove siamo partiti, ⏎
5: e conoscere il posto per la pirma volta. ⏎
6: ↑Z ⏎
:*
```

Ora potete visualizzare l'intero buffer usando i comandi B e T:

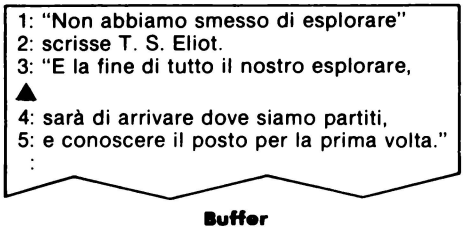
```
:B#T ⏎
1: "Non abbiamo smesso di esplorare"
2: scrisse T.S. Eliot.
3: "E la fine di tutto il nostro esplorare,
4: sarà di arrivare dove siamo partiti,
5: e conoscere il posto per la prima volta".
1: *
```

Il modo più semplice di spostarsi a un'altra riga è di usare il numero di riga. Se avete la versione 2.2 del CP/M, ED visualizzerà automaticamente i numeri di riga. Se avete la versione 1.4, questa caratteristica è disattivata; attivatela eseguendo il comando di ED V (- V la disattiva di nuovo). Se avete una versione precedente alla 1.4 siete sfortunati - non potete visualizzare i numeri di riga e dovete usare il comando L per spostarvi a un'altra riga.

Per selezionare una riga, digitate il numero di riga seguito da un carattere ":" come comando di ED:

```
1: *2: ↵
2: *
```

Per selezionare una serie di righe, digitate il primo numero di riga, seguito da *due* caratteri ":" e il secondo numero di riga, come in questo esempio (si può anche usare il comando T per visualizzare una serie di righe):




```
2: *3::5T ↵
3: "E la fine di tutto il nostro esplorare,
4: sarà di arrivare dove siamo partiti,
5: e conoscere il posto per la prima volta".
3: *
```

Quando specificate una sola riga, il CP viene portato all'inizio di quella riga. Quando specificate una serie di righe, il CP viene portato all'inizio della prima riga della serie e ED conta il numero di righe della

serie (in questo caso 3) e applica quel numero al comando T (cioé '3T') per visualizzare la serie. Il CP rimane sulla riga 3.

Potete anche spostarvi avanti e indietro sulle righe usando il comando L. Il formato del comando L è:

± nL

Il comando L muove il CP + n righe avanti o -n righe indietro nel buffer e porta il CP all'inizio della riga selezionata. Ecco un esempio:

```
3: *1L ↵
4: *-2L ↵
2: *
```

La formula '0L' (in cui n è zero) sposta il CP all'inizio della riga corrente. Potete usare i numeri di riga per selezionare una serie di righe che incomincia alla riga corrente facendo precedere il numero della riga finale con un carattere ":":

```
2: *:5T ↵
2: scrisse T.S Eliot.
3: "E la fine di tutto il nostro esplorare,
4: sarà di arrivare dove siamo partiti,
5: e conoscere il posto per la prima volta".
```

CAMBIAMENTI, INSERZIONI E CANCELLAZIONI DI TESTI

Il testo in una riga viene cambiato portando il CP sul testo, cancellando quello esistente e inserendo il nuovo. Potete anche trovare un gruppo di caratteri e sostituirlo con un altro gruppo, come trattato nella sezione successiva.

Quando cancellate una riga di testo, i numeri di riga riflettono la

cancellazione e cambiano di conseguenza, come in questo esempio:

```
2: *1::5T ↵
1: "Non abbiamo smesso di esplorare"
2: scrisse T.S. Eliot.
3: "E la fine di tutto il nostro esplorare,
4: sarà di arrivare dove siamo partiti,
5: e conoscere il posto per la prima volta."
1: 2: ↵
2: K ↵
2: * 1::4T ↵
1: "Non abbiamo smesso di esplorare"
2: "E la fine di tutto il nostro esplorare.
3: sarà di arrivare dove siamo partiti,
4: e conoscere il posto per la prima volta".
1: *
```

Il comando K riportato sopra cancella la riga 2 e le altre righe "salgono" per usare lo spazio. L'opposto avviene quando inserite una riga - le altre righe "scendono" per lasciar posto alla riga appena inserita.

Il formato del comando K è:

$$\pm \text{nK}$$

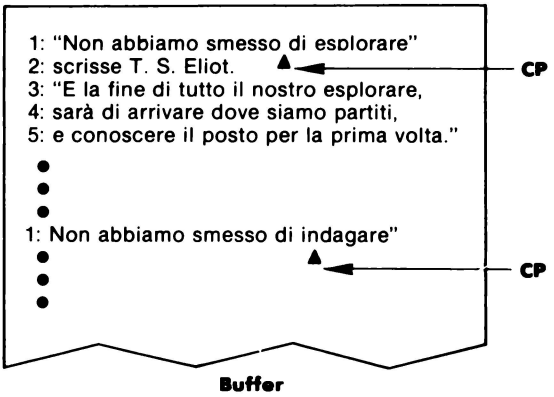
Il comando K cancella ("Kill") la riga corrente se nessun n è specificato. Altrimenti, il comando K cancella o + n o - n righe dalla riga corrente. Se fornite il simbolo (#) al posto di n, K cancella 65535 righe a partire (e comprendendo) la riga corrente (+#K), o 65535 righe seguenti (e non comprendenti) la riga corrente (-#K). Usate questi comandi per eliminare le righe non desiderate dal buffer, ma ricordate - non potete recuperare queste righe (a meno che esistano già nel file sorgente o nel file di backup).

Per cancellare *caratteri* (non righe intere), usate il comando D:

$$+ n\text{D}$$

Il comando D cancella il carattere puntato dal CP (puntatore di caratteri) se nessun n è specificato. Altrimenti, il comando D cancella + n caratteri seguenti (e comprendenti) il CP, o -n caratteri indietro (e *non* comprendenti) il CP.

Ecco un esempio: vogliamo cambiare la parola 'esplorare' nella parola 'indagare' (e poi tornare indietro a 'esplorare'). Lo faremo nel modo più brutale - muovendo il CP per mezzo del comando C, cancellando 'esplorare' per mezzo del comando D e inserendo 'indagare' per mezzo del comando I:




```
1: * 1::4T ⏎
1: "Non abbiamo smesso di esplorare"
2: "E la fine di tutto il nostro esplorare,
3: sarà di arrivare dove siamo partiti,
4: e conoscere il posto per la prima volta."
1: *T ⏎
1: "Non abbiamo smesso di esplorare"
1: *23C ⏎
1: *9DI indagare ↑ ZOLT ⏎
1: "Non abbiamo smesso di indagare"
1: *
```

In questo esempio portiamo il CP sul 24esimo carattere usando il comando '23C'. Contando il carattere puntato da CP, cancelliamo 9 caratteri usando il comando D. Poi usiamo il comando I per inserire 'indagare' (terminato da ↑ Z) - si noti che il comando I inserisce prima del CP e sposta il CP. Il comando D cancella il carattere puntato dal CP e muove anche il CP. Usiamo il comando L per spostare il CP all'inizio della riga e usiamo il comando T per visualizzare la riga.

Questa è una nuova formula del comando I:

Iinserzioni ↑ Z

Realizza la stessa funzione di I ↵, tranne che non inserisce automaticamente i caratteri "ritorno carrello" per generare nuove righe. Se volete cambiare un gruppo di caratteri che comprende una sequenza 'ritorno carrello', usate il comando S (sostituzione) descritto nella sezione successiva, perché è molto più semplice che contare i caratteri per muovere il CP in modo corretto.

RICERCA E SOSTITUZIONE DI TESTI

Un modo semplice di spostare il CP su un gruppo di caratteri nel testo è di "cercare" (find) quel gruppo di caratteri per mezzo del comando F. Il comando S, trattato più avanti, può essere usato per "cercare e sostituire" (substitute). Il comando F è la versione primitiva del comando S e sarà descritto per primo.

Un esempio mostrerà il formato del comando F. Lo eseguiremo per cercare 'indagare' nella riga 1:

```
1: *Findagare ↵
1: *
```

L'istruzione mostra dove si trova il CP in questo momento:

```
"Non abbiamo smesso di indagare▲"
                              CP
```

Il comando F sposta il CP sul carattere immediatamente seguente l'ultimo carattere trovato, in modo da rendere semplice l'uso del comando D

(per cancellare i caratteri trovati). Sposteremo il CP indietro all'inizio della riga ed eseguiremo il comando F insieme con i comandi D e I nell'esempio seguente:

```
1: 0L ⏎
1: *Findagare ↑ Z - 8Dlesplorare ↑ ZOLT ⏎
1: "Non abbiamo smesso di esplorare"
1: *
```

Il comando -8D cancella gli otto caratteri precedenti il CP (cioé 'eragadni'). Il CP è ora in posizione per usare il comando I che inserisce "esplorare". La combinazione di comandi '0LT' sposta il CP all'inizio della riga e la visualizza.

Tenete presente che potete terminare un comando F con un RETURN se non includete un altro comando sulla stessa riga.

Il comando F può anche incominciare con un numero (nF dove n è un numero positivo) che dice di trovare n occorrenze del gruppo di caratteri. Per esempio, se digitate il comando '5Fquesto ⏎', il comando non si ferma e sposta il CP fino a quando ha trovato la 5° occorrenza di 'questo'. Il comando F può cercare nel testo contenuto nel buffer; per cercare nell'*intero file sorgente*, dovete usare il comando N (trattato in "operazioni avanzate di ED").

Se volete trovare un gruppo di caratteri che comprende una sequenza 'ritorno carrello', dovete sostituire una speciale combinazione di tasti, ↑ L, per rappresentare la sequenza di RETURN e LINE FEED. Nel nostro esempio con il comando S, dimostreremo l'uso di ↑ L.

Il comando S (sostituzione) è quello usato più spesso per cercare e sostituire gruppi di caratteri. Il comando S esegue le azioni in successione dei comandi F, D e I mostrati prima. Il formato del comando S è:

nSvecchiotesto ↑ Znuovotesto $\left\{\begin{matrix}\uparrow Z\\ \hookleftarrow\end{matrix}\right\}$

Il comando S cerca nel buffer il gruppo di caratteri *vecchiotesto* e lo sostituisce con il gruppo di caratteri *nuovotesto*. Potete usare ↑ Z o RETURN. Per terminare la stringa *nuovotesto* (usate ↑ Z se volete aggiungere altri comandi al comando S). Potete eseguire questa sostituzione un numero *n* di volte e verrà eseguita fino a quando si raggiungerà l'ennesima volta o fino a quando si raggiunge la fine del buffer.

Per esempio, se digitate il comando "#SPeking ↑ Z Beijing ⏎', questo

sostituirà la parola "Beijing" al posto di "Peking" in tutto il testo nel buffer (Il simbolo (#) rappresenta 65535 volte).

Useremo il comando S per modificare l'esempio e cambiare il testo dalla prosa alla poesia. Dapprima, correggeremo la fine della prima riga e l'inizio della seconda riga con un comando S:

```
1: * 2T ⏎
1: "Non abbiamo smesso di esplorare"
2: "E la fine di tutto il nostro esplorare,
1: *S" ↑ L" ↑ Z/ ↑ L ↑ ZB2T ⏎
1: Non abbiamo smesso di esplorare/
2: E la fine di tutto il nostro esplorare,
1: *
```

Nell'esempio sopra, usiamo il comando S per cercare il gruppo di caratteri che incomincia con le virgolette e termina ancora con le virgolette nella riga successiva (↑ L è al posto di 'Ritorno carrello'), e li sostituiamo con un '/' seguito da un 'Ritorno carrello'. Poi eseguiamo la combinazione 'B2T' per spostare il CP all'inizio del buffer e visualizzare le due righe successive.

Il nostro esempio successivo sostituirà una '/' al posto della virgola (,) in due righe:

```
1: *2::4T ⏎
2: E la fine di tutto il nostro esplorare,
3: sarà di arrivare dove siamo partiti,
4: e conoscere il posto per la prima volta".
4: *2: ⏎
2: *2S, ↑ Z/ ↑ ZB4T ⏎
1: "Non abbiamo smesso di esplorare/
2: E la fine di tutto il nostro esplorare/
3: sarà di arrivare dove siamo partiti/
4: e conoscere il posto per la prima volta".
1: *
```

In questo esempio, prima spostiamo il CP all'inizio della riga 2 ('2:'), e poi realizziamo la sostituzione ('/' al posto di ','). Poi spostiamo CP all'inizio del buffer e visualizziamo quattro righe (B4T).

Dobbiamo soltanto cambiare il primo carattere delle righe 3 e 4 in maiuscolo:

```
1: *3: ⏎
3: *DIS ↑ Z1LDIE ↑ ZB4T ⏎
1: "Non abbiamo smesso di esplorare/
2: E la fine di tutto il nostro esplorare/
3: Sarà di arrivare dove siamo partiti/
4: E conoscere il posto per la prima volta."
1: *
```

In questo esempio, portiamo il CP all'inizio della riga 3 e cancelliamo il primo carattere (il comando D cancella 's'). Poi inseriamo (comando I) una S. Portiamo il CP alla riga successiva (comando 1L), cancelliamo il primo carattere (il comando D cancella 'e') e inseriamo una E. Infine portiamo il CP all'inizio e visualizziamo 4 righe.

SCARICAMENTO DELLE RIGHE NEL FILE DI USCITA

Normalmente, terminerete la sessione di lavori con ED usando i comandi E o H. Ciascuno realizza un'operazione di scrittura sull'intero buffer; cioé, *scrive* le righe del testo nel file temporaneo di uscita. Entrambi i comandi copiano anche il resto del file sorgente (cioé, le righe che non erano state appese al buffer) nel file di uscita, e modificano i nomi dei files in modo che terminate con il testo appena modificato nel file sorgente. Il comando E termina anche ED; mentre il comando H vi mantiene in ED e prepara il nuovo file sorgente per una successiva attività di editing.

Se volete soltanto scrivere righe nel file di uscita senza copiare l'intero buffer o il resto del file sorgente, usate il comando W (write). Il comando W prende la forma seguente:

```
nW
```

Il comando W scrive n righe a partire dalla riga corrente (comprendendo

la riga corrente) nel file di uscita. Il file di uscita è il file temporaneo con l'estensione '$$$'. Se non specificate n, la riga corrente viene scritta nel file.

Per illustrare il comando W, aggiungiamo altre righe al nostro esempio e scriviamo alcune righe nel file temporaneo di uscita:

```
1: * -BI ↵
5: ↵
6:                                  -scrisse T.S. Eliot ↵
7: ↑Z
6: B#T ↵
1: "Non abbiamo smesso di esplorare/
2: E la fine di tutto il nostro esplorare/
3: Sarà di arrivare dove siamo partiti/
4: E conoscere il posto per la prima volta."
5:
6:                                  -scrisse T.S. Eliot
1: *3W ↵
1: #T ↵
1: E conoscere il posto per la prima volta."
2:
3:                                  -scrisse T.S. Eliot
1: * #W ↵
 : *
```

In questo esempio, prima aggiungiamo nuovo testo alla fine (comandi '-BI') del buffer (usiamo un RETURN per fare una riga bianca e un ↑I per inserire una tabulazione). Poi visualizziamo l'intero buffer con il CP all'inizio del buffer (comandi 'B#T'). Quindi scriviamo le tre righe successive, comprendendo la riga corrente (righe 1, 2 e 3). Queste righe sono ora in QUOTE.$$$, il file temporaneo di uscita. Le righe rimanenti del buffer sono ora rinumerate. L'ultimo comando ('#W') scrive le

successive 65535 righe comprendendo la riga corrente, il che scarica il resto del buffer. Il buffer è vuoto e ED non visualizza alcun numero di riga (':*').

Dobbiamo ancora salvare il *resto* del file sorgente (se ci fosse qualche riga non appesa) e *modificare* il nome del file temporaneo di uscita QUOTE.$$$ in QUOTE.TXT (e modificare il nome del vecchio QUOTE.TXT in QUOTE.BAK). I comandi E ed I fanno questa operazione automaticamente.

Una forma speciale del comando W funziona con la forma '0A' del comando A - '0W' (dove *n* è 0) scarica metà del buffer e '0A' appende righe fino a riempire metà del buffer. Il numero di righe che corrisponde a metà del buffer dipende dalla lunghezza del testo e dall'estensione del sistema. Queste forme sono usate soprattutto per "caricare mezzo buffer, scaricarne mezzo, caricarne ancora mezzo, ecc.".

Se volete modificare la sequenza delle righe nel buffer o inserire righe di un altro file, allora dovete usare comandi speciali (descritti in "Operazioni avanzate") per portare righe dai files al buffer e *poi* scrivere la versione definitiva in un file usando il comando W (o salvare l'intero buffer e il rimanente del file sorgente con i comandi E o H).

OPERAZIONI AVANZATE DI ED

Ricerca attraverso il file sorgente

Potete usare il comando N per fare una ricerca (comando F) nel buffer *e* nel resto del file sorgente. Il comando N opera allo stesso modo del comando F ma non si ferma alla fine del buffer - appende automaticamente e continua ad appendere righe dal file sorgente fino a quando trova il gruppo di caratteri specificato nel comando.

Il formato del comando N è:

NOTA: le parentesi { } indicano un'opzione tra due comandi. Il comando N cerca l'n-sima occorrenza di *testo* nelle righe che seguono il CP nel buffer; se non trova l'n-esima occorrenza, appende righe dal file sorgente nel buffer fino a quando la trova. Potete terminare la stringa *testo* con il simbolo Z se volete aggiungere un altro comando; altrimenti potete usare RETURN.

Il comando N porta il CP dopo l'ultimo carattere dell'n-esima occorrenza di *testo*, proprio come il comando F.

Inserzione di testo da un file di sorgenti di libreria

Un "file di sorgenti di libreria" è un termine usato nella documentazione della Digital Research per qualsiasi file con un'estensione 'LIB' (ad esempio, SAMPLE.LIB). Potete usare un "file di sorgenti di libreria" come un file sorgente alternativo - un file con testo che volete inserire nel buffer mescolandolo con le righe del file sorgente originale. Per far questo dovete prima avere un file con un'estensione 'LIB' che contenga già il testo.

Il formato del comando R è:

```
Rnomefile
```

Il comando R inserisce le righe dal file LIB che specificate in *nomefile*. Inserisce le righe nella posizione corrente del CP in modo simile al comando I. R inserisce l'intero file 'LIB' fino a quando trova l'indicatore di end-of-file (↑ Z).

Trasferimento di righe in un file temporaneo e inserzione di righe da un file temporaneo

Se avete la versione 1.4 o una versione successiva del CP/M, potete usare il nuovo comando X per trasferire righe dal buffer in un file temporaneo di "deposito" e potete usare il comando R per inserire righe nel buffer da questo file di "deposito". Il file di "deposito" è chiamato X$$$$$$$.LIB ed esiste soltanto mentre funziona il programma ED. Qualsiasi terminazione normale di ED cancella il file, ma se terminate ED con un ↑ C (partenza a caldo), il file rimarrà (comunque, quando eseguite di nuovo ED, il file viene cancellato).

Il formato del comando X è:

```
nX
```

Il comando X trasferisce (copia) n righe successive a partire dalla riga corrente del file temporaneo X$$$$$$$.LIB. Le righe nel file rimangono lì: sono soltanto copiate nel file temporaneo. Se volete potete usare il comando K per cancellare le righe dopo averle copiate. Le righe trasferite si accumulano nel file temporaneo nell'ordine in cui sono state copiate dai successivi comandi X. Usando questo comando, si può costruire un nuovo testo in blocchi successivi.

Le righe possono essere recuperate usando il comando R nella forma 'R' senza il *nomefile*. Tutte le righe trasferite sono allora inserite a partire

dal CP (in modo simile al comando I). Comunque, il comando R non vuota il file temporaneo, semplicemente copia le righe. Potete recuperarle di nuovo quante volte volete (utile per righe ripetitive). Potete vuotare il file temporaneo (cioé cancellarlo), eseguendo la forma '0X' (dove n è zero) del comando X.

Se volete salvare il file temporaneo X$$$$$$$.LIB, usate ↑ C per interrompere ED e cambiate immediatamente il nome del file (per liberarvi dall'estensione LIB) cosicché non sarà distrutto quando eseguirete di nuovo ED.

Giustapposizione

Il comando J è usato per giustapporre tre gruppi di caratteri in un testo. Trova un gruppo di testo, giustappone un secondo gruppo e cancella i caratteri seguenti questa copia fino a quando trova un terzo gruppo di testo, giustapponendo effettivamente tutti e tre i gruppi di testo. Il formato del comando J è:

$$n\text{Jstringa1} \uparrow \text{Zstringa2} \uparrow \text{Zstringa3} \left\{ \begin{matrix} \uparrow \text{Z} \\ \hookleftarrow \end{matrix} \right\}$$

Il comando J parte cercando dal CP nel buffer la prima occorrenza di 'stringa1'. Se trova 'stringa1', il comando J inserisce 'stringa2' immediatamente dopo 'stringa 1' e sposta il CP alla fine di 'stringa2'. Il comando J poi cerca 'stringa 3'; se trova 'stringa 3' il comando J cancella tutti i caratteri tra 'stringa2' e 'stringa3' e lascia il CP che punta sul primo carattere di stringa3. Se il comando J non trova 'stringa3' non cancella nulla. Il comando J fa questo un numero n di volte o fino a quando esce dalle righe del buffer. Un uso del comando J è di accorciare le righe del testo. Scegliete un gruppo di caratteri che sia alla fine della nuova riga da accorciare. Questi saranno la 'stringa1'. Il concetto è di giustapporli alla sequenza 'Ritorno Carrello', rappresentata da ↑ L ('stringa3'). Il risultato è ottenuto inserendo una 'stringa2' vuota.

Ripetizione di un insieme di comandi

Potete collegare insieme molti comandi di ED in un solo "comando" che possa poi essere eseguito ripetutamente. Il comando M ("macro") prende la forma seguente:

$$n\text{Mstringadicomandi} \left\{ \begin{matrix} \uparrow \text{Z} \\ \hookleftarrow \end{matrix} \right\}$$

Raggruppate i comandi in 'stringadicomandi', preceduta da 'M', e M eseguirà i comandi n, se n è maggiore di 1. Se n è 0 o 1, la 'stringadicomandi' è eseguita ripetutamente fino a quando si verifica un errore (come il raggiungimento della fine del buffer).

Ecco un esempio che cambia tutte le occorrenze di 'Peking' in 'Beijing' nel buffer corrente e stampa tutte le righe modificate:

```
MSPeking↑ZBeijing↑ZOTT
```

CONDIZIONI DI ERRORE DI ED

Quando si verifica un errore all'interno del programma ED, questo stampa l'ultimo carattere che ha incontrato prima dell'errore e uno dei seguenti indicatori di errore:

?	Non riconosce il comando, che cos'é?
>	Il buffer è pieno. Usate uno di questi comandi D, K, S, W, E, o H per eliminare caratteri. Oppure, la stringa con il comando F, N o S è troppo lunga.
#	Non può eseguire il comando tutte le volte richieste (come in un comando F che raggiunge la fine del buffer).
0	Non può aprire il file LIB in un comando R (verificate se il file esiste o se avete usato il nome corretto).

Figura 4.15: Messaggi di errore di ED

Le nuove versioni del CP/M stampano 'BREAK x AT c' dove x è uno dei simboli di errore e c è il comando ED in esecuzione.

Occasionalmente, il sistema sotto ED (cioé il CP/M o MP/M) trova una condizione di errore di sistema (come un errore sul disco). A seconda dell'errore, potete di solito terminare ED con un ↑ C (partenza a caldo), ma dovete prima recuperare la copia originale del file sorgente. Per esempio, se il CP/M trova un errore CRC (cyclic redundancy check, cioé controllo di parità longitudinale), stamperà:

```
PERM ERR DISK u
```

dove u è l'unità a dischi corrente. Potete scegliere di ignorare l'errore digitando qualsiasi carattere al terminale (comunque dovreste verificare il buffer se vi sono errori) o potete realizzare una partenza a caldo (↑C) o a freddo (rilanciare il sistema) e recuperare il file BAK (il file con estensione BAK, ad esempio QUOTE.BAK).

Riassunto dei miglioramenti del CP/M versione 2.2

Nel CP/M versione 2.2, ED assume che l'opzione di numerazione delle righe sia sempre attiva. Per eliminare i numeri di riga, digitate: -V, (naturalmente questa modalità può essere riattivata con: V).

Quando si usa l'inserzione (I), i caratteri di controllo soliti del CP/M possono essere usati tutti: DEL, ↑C, ↑E, ↑H, ↑J, ↑M, ↑R, ↑U, ↑X. Sono descritti nell'Appendice D.

Infine, ED rispetta gli attributi dei files della versione 2.2. Per esempio, un file a sola lettura può essere esaminato ma non modificato. Se il file deve essere modificato allora bisogna prima modificare lo stato di sola lettura in R/W, usando un comando STAT.

SOMMARIO

In questo capitolo, avete imparato come usare un editor, ED. ED è un editor generalizzato che permette di modificare testi con pochi comandi chiave. ED è usato per lo più per creare files semplici, come programmi (se non è disponibile nessun altro editor specializzato). Sebbene i comandi di ED siano meno convenienti di quelli di un "word processor" specializzato, può essere utilizzato per digitare lettere o documenti.

ED è uno strumento comodo per correggere o modificare files esistenti. (In particolare, ED può essere usato per "pulire" un dischetto di sistema che abbia caratteri errati o spurî dovuti ad errori). ED può anche essere usato per sostituire nuove parole o righe in un file. Un riassunto dei caratteri di controlo e dei comandi di ED è presentato nelle Appendici D e E.

In questo capitolo, non avete soltanto imparato come usare ED, l'editor, ma avete imparato ad usare molti comandi, caratteri di controllo e altre convenzioni. Avete anche imparato come operare sui files e avete seguito i trasferimenti di testo tra dischi, memoria e schermo. Questa conoscenza vi aiuterà a capire le operazioni di molti altri programmi sul calcolatore.

CAPITOLO 5

DENTRO AL CP/M (E ALL'MP/M)

INTRODUZIONE

Questo capitolo descriverà le operazioni interne del sistema operativo del CP/M. Leggete questo capitolo se siete interessati a conoscere il modo in cui funziona un sistema operativo o pensate di modificare o usare alcune delle routines del CP/M. Se invece volete soltanto realizzare un compito specifico per mezzo del CP/M, le informazioni contenute in questo capitolo non sono necessarie.

Questo capitolo sarà particolarmente valido per un programmatore di sistema o per qualsiasi persona che, avendo familiarità con la programmazione vuole capire come funziona il CP/M. Poiché un libro sul CP/M non sarebbe completo senza queste informazioni, il capitolo è stato realizzato per questi particolari lettori. Se volete sapere "che cosa accade dietro le quinte", andate avanti.

Presenteremo dapprima uno sguardo d'insieme semplificato delle operazioni del CP/M, introducendo i suoi componenti logici, il loro ruolo e il modo in cui interagiscono tra loro. Quindi sarà descritta la *allocazione di memoria* cioé, il modo in cui questi moduli di software sono sparsi nella memoria, insieme all'organizzazione del files sistem (sistema di gestione del file). Poi ciascuno dei tre moduli del CP/M sarà presentato in dettaglio insieme ai comandi che sono disponibili per usare le loro capacità. Un'altra sezione tratterà i problemi che si incontrano nell'adattare il CP/M a una nuova configurazione di hardware e sarà mostrato un esempio delle modifiche al CP/M per farlo funzionare come un sistema a "menu". Infine, si dedicherà una sezione speciale all'MP/M.

UNO SGUARDO ALLE OPERAZIONI DEL CP/M

Schema a blocchi

Il CP/M ha tre moduli funzionali. Si chiamano *CCP* (Console Command Processor o Gestore dei Comandi di Console), *BIOS* (Basic Inpu-

t/Output Sistem, o Sistema di Ingresso/Uscita di Base) e *DBOS* (Basic Disk Operating System o Sistema Operativo Di Base su Disco) (Fig. 5.1).

La funzione del CCP è di comunicare con l'utente e di interpretare i comandi digitati su tastiera. Il CCP è prima di tutto un *interprete di comandi* e usa le risorse fornite dagli altri due moduli, BIOS e BDOS. Questa è soltanto una descrizione semplificata poiché vedremo più avanti che il CCP, in effetti, esegue anche un'elaborazione interna. Concettualmente, il CCP può essere visto come la "parte intelligente" del sistema operativo, mentre gli altri due moduli sono moduli di servizio.

Il BIOS contiene una serie di *gestori di periferiche*, cioé, routines che hanno l'incarico di comunicare con i vari dispositivi connessi al sistema.

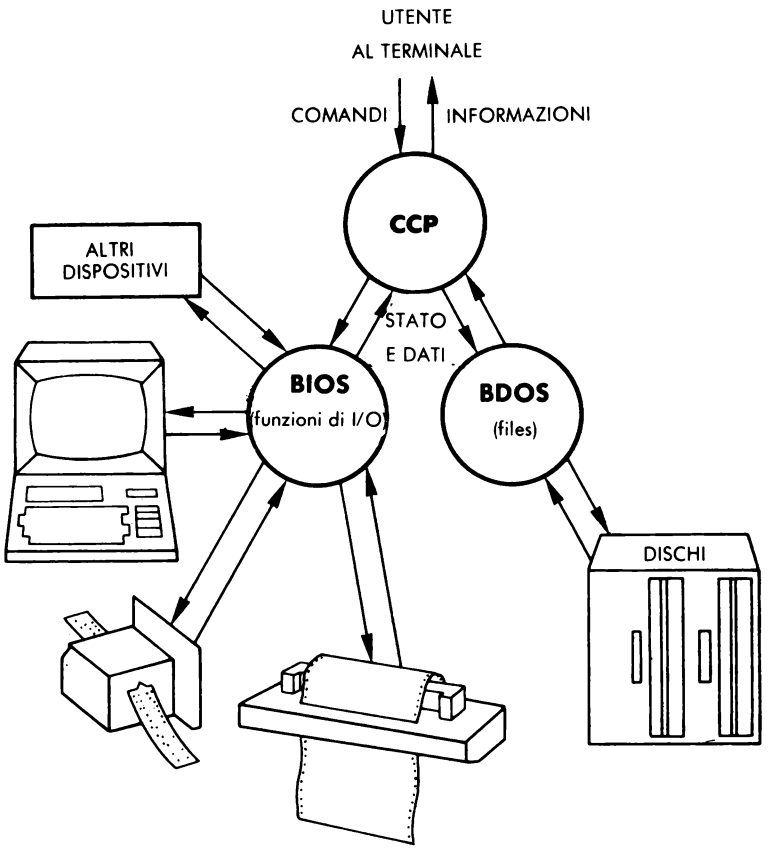


Figura 5.1: Schema a Blocchi

La funzione del BIOS è di mandare o ricevere lo stato e le informazioni tra un dispositivo e il debugger del CCP. Il BIOS è chiamato dal CCP con parametri specifici che individuano il servizio richiesto. Questo è illustrato nella Fig. 5.1.

Il BDOS dirige la gestione dei files su disco. Comprende alcune routines di utilità che realizzano le funzioni richieste sul disco. Come ogni buon sistema operativo a dischi, l'obiettivo del BDOS è di rendere invisibile (o "trasparente") il gestore di files all'utente. Il BDOS si assume tutti i compiti richiesti per localizzare i vari blocchi di informazioni sparsi sulla superficie del dischetto, verificare la validità dell'accesso e l'integrità dei dati, e allocare e rilasciare in modo efficiente lo spazio di memoria.

L'allocazione della memoria

Il CP/M suddivide la memoria disponibile in quattro zone, come mostrato nella Fig. 5.2. La parte superiore della memoria è riservata alle

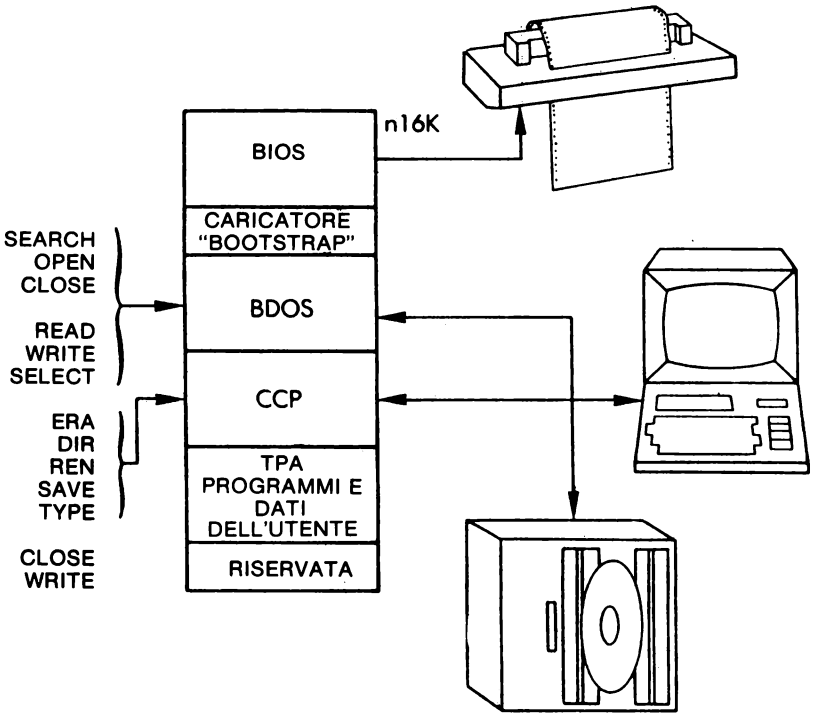


Figura 5.2: Mappa di memoria del CP/M

routines del CP/M propriamente dette, cioé CCP, BDOS, BIOS, (Fig. 5.2). Alcune allocazioni in fondo alla memoria sono riservate al sistema. Sono le prime 256 locazioni di memoria, cioé *la pagina 0* (descritta con maggior dettaglio più avanti). Infine, la parte più grande della memoria, tra la locazione esadecimale 0100 e CBASE (chiamata TPA) è disponibile per l'esecuzione dei programmi. Questa allocazione standard della memoria carica i programmi nella posizione 100H nei calcolatori a bus S-100 come quelli della Cromemco, della Imsai, della Altair e della North Star. Invece, nel caso degli altri calcolatori (ad esempio, il TRS80 e Heat H8) con programmi memorizzati in ROM nella parte bassa della memoria, i programmi incominciano all'allocazione 4300H.

La TPA o *Transient Program Area* (Area dei Programmi Transistori) è il nome usato nella documentazione del CP/M per ogni programma che deve essere eseguito. Il CP/M assume che il sistema abbia 16, 32, 48 o 64kbyte di memoria. In un sistema a 16K la base del CCP, chiamata CBASE, è all'indirizzo di memoria 2900 (esadecimale). Per ogni 16K aggiunti al sistema, questo indirizzo è incrementato di 4000 (esadecimali). Ciò significa che ogni 16K di memoria aggiunti al sistema estende la TPA di questa quantità. Si vedrà più avanti che un programma utente può usare quasi tutta la memoria ricoprendo il CCP o altre aree di memoria che appartengono alle routines del CP/M. Questo sarà trattato con maggior dettaglio più avanti. Comunque, non significa che quando termina l'esecuzione del programma, bisogna ricaricare il CP/M in memoria prima di uscire.

DESCRIZIONE DETTAGLIATA

Il file system

Una delle funzioni principali di ogni *sistema operativo a dischi* (DOS) è di provvedere un'effettiva e conveniente gestione dei files su disco. Una comprensione dell'organizzazione generale del file system è quindi richiesta per capire le operazioni del sistema operativo a dischi vero e proprio (BDOS per il CP/M), così come le operazioni del CCP, che costruisce e manipola i descrittori dei files. Tutta la gestione dei files è fatta dal CCP e dal BDOS e la porzione BIOS del CP/M non deve preoccuparsi della natura esplicita di un file. Il BIOS essenzialmente trasmette e riceve dei semplici flussi di dati. Esaminiamo la struttura dei files del CP/M.

Un *file* è un'unità logica che contiene testo, dati o programmi. Il compito del sistema operativo a dischi è di realizzare questa possibilità logica con le risorse fisiche fornite dal supporto di memoria, cioé il dischetto nel nostro caso. Vediamo come questo viene fatto.

Abbiamo già spiegato che i dischetti sono organizzati in tanti settori.

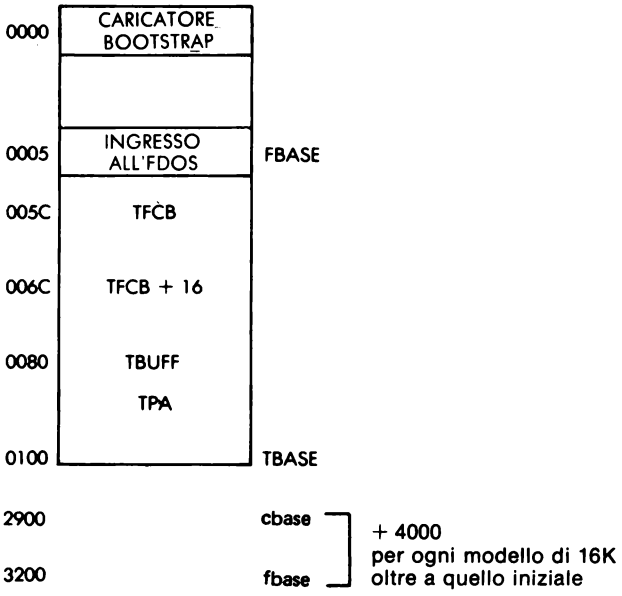


Figura 5.3: Mappa standard del CP/M

Ogni settore su un dischetto standard a 8 pollici contiene 128 bytes di informazioni. Nella nomenclatura del CP/M, questa porzione è chiamata *record.* Ogni file sul dischetto è una collezione di records. Poiché non è possibile conservare un file come una sequenza completa di records sul disco, è necessario usare settori sparsi sull'intera superficie del disco. Si deve stabilire una certa forma di struttura a lista per tener traccia di tutti i records associati a un file. Si possono usare molte tecniche a questo scopo. Nel caso del CP/M, la lista dei settori che appartengono a un file su disco è contenuta in un'entità speciale, chiamata *descrittore* nella terminologia della scienza dei calcolatori, o *blocco di controllo del file* nella nomenclatura del CP/M.

Con il CP/M, ogni file può avere un massimo di 16 unità. Ogni unità contiene da 0 a 128 records, cioé, da 0 a 16K byte. Il più grande file del CP/M può pertanto avere fino a 16 unità di 16K byte, cioé 16 x 16K = fino a 256K byte. Questa quantità è leggermente più grande della capacità massima di un dischetto standard da 8 pollici. Ogni *blocco di controlo del file* (file control block - FCB) descrive fino a 16K byte di un file particolare. Sono forniti meccanismi addizionali per collegare insieme fino a 15 estensioni del file.

L'utilizzazione effettiva dello spazio su disco è mostrata in Fig. 5.4. Le due tracce più esterne del dischetto sono usate per memorizzare il sistema CP/M. Il resto del disco è usato per memorizzare i files e la direttrice dei files.

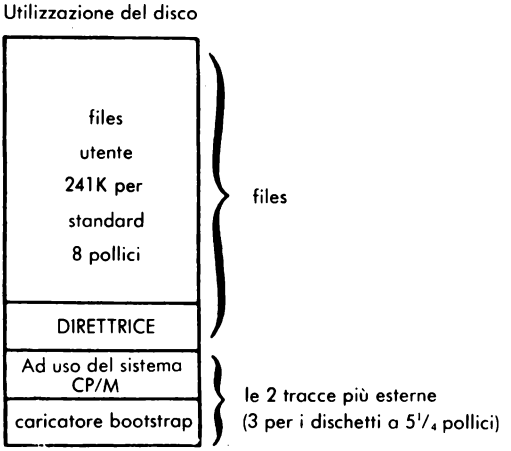


Figura 5.4: Utilizzazione dello spazio su disco

Il blocco di controllo del file stesso utilizza 33 bytes ed è memorizzato in un'area della direttrice del dischetto riservata a quello scopo (Fig. 5.4). Quando il file è attivato, cioé vi si accede tramite CP/M, il suo FCB viene portato nell'area dei programmi transitori in modo che il sistema operativo può accedervi in modo rapido e conveniente.

Abbiamo realizzato il nostro obiettivo: quello di assegnare a un file logico i settori disponibili su un dischetto standard. Questo è stato fatto mantenendo una direttrice dei blocchi assegnati al file in una locazione speciale chiamata FCB.

Un requisito di base di ogni buon sistema operativo è che l'utente sia in grado di designare un file usando un *nome simbolico* (l'utente trova fastidioso riferirsi a un file per mezzo di numeri). Il CP/M fornisce questa possibilità, insieme a una salvaguardia addizionale che richiede all'utente di specificare anche il *tipo* del file. Pertanto, si deve fornire un meccanismo per localizzare un file *effettivo* quando viene fornito il suo nome *logico*, guardando nella direttrice del file fino a quando si trova un nome che corrisponde a quello fornito dall'utente. Questa associazione tra un file effettivo e il suo nome è realizzata nell'FCB. L'altra parte

essenziale del blocco di controllo del file (FCB) è la mappa del file. Si veda la Fig. 5.5.

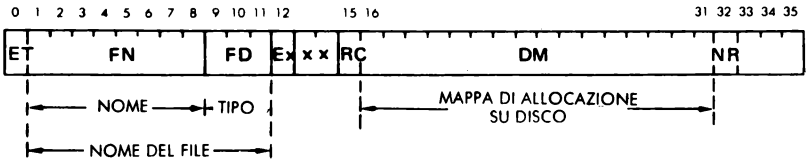


Figura 5.5: Blocco di controllo del file

Il compito successivo associato con il progetto di un efficiente file system è di provvedere delle caratteristiche di sicurezza e di salvataggio per il periodo in cui si accede ai files. Per esempio, si può caratterizzare i files come *Sola-Lettura* o *Lettura-Scrittura*. Ancora, si può specificare un accesso con *parola d'ordine* per un file o un file può essere *eseguibile* ma non *leggibile* e dotato di molti dispositivi di protezione che prevengono gli accessi o le esecuzioni non autorizzate. Il CP/M versione 1.4 non fornisce virtualmente tali salvaguardie, ma ha riservato molti bytes nell'FCB (blocco di controllo dei files) per incorporarle. Il CP/M versione 2.2 fornisce ora 4 attributi per i files: R/O, R/W, DIR e SYS.

Infine, sono richiesti alcuni bytes per le funzioni di registrazione del sistema (Fig. 5.5). In particolare, la locazione NR nella posizione 32 contiene il numero del record successivo da leggere o scrivere.

Riguardiamo i campi del Blocco di Controllo dei Files mostrati in Fig. 5.5. Contiene 8 campi che sono, da sinistra a destra:

ET (posizione 0): Questo è il tipo di ingresso. Normalmente è 0 e non è usato nel CP/M 1.4. È il codice di unità nel CP/M versione 2.2 e nell'MP/M.

FN (posizioni 1-8): È il nome del file che contiene fino a 8 caratteri. Ogni carattere non fornito dall'utente è sostituito da uno spazio in codice ASCII.

FD (posizioni 9-11): È il tipo del file. Contiene da uno a 3 caratteri alfanumerici ed è completato, in modo simile al precedente, con spazi, se necessario. Ogni volta che viene fatta una lista

	della direttrice di un dischetto, vengono stampati il nome del file e il tipo. Nel CP/M versione 2.2, due bits indicano l'attributo e un bit indica che il file è stato modificato.
EX (posizione 12):	È l'estensione, normalmente 0.
XX (posizione 13 e 14):	Questa parte non è usata nel CP/M versione 1.4, normalmente 0.
RC (posizione 15):	È il contatore di record. Un modulo di file può avere da 0 a 128 record (fino a 16K bytes). È chiamato misura di estensione corrente.
DM (posizioni 16-31):	La mappa di allocazione su disco, tiene traccia dei settori effettivi su disco usati da questo modulo del file.
NR (posizione 32):	Contiene il numero del record successivo da leggere o da scrivere, normalmente 0, e chiamato CR nel CP/M versione 2.2. R0-1-2 (posizione 33-35) è usato soltanto dal CP/M versione 2.2 per gli accessi casuali e rappresenta il numero di record opzionale (da 0 a 64K).

Operazioni di sistema

Sotto il sistema operativo CP/M, ogni programma in linguaggio macchina è installato o caricato nell'area dei programmi transitori (mostrata in Fig. 5.2). Nel "CP/M standard", l'esecuzione parte all'indirizzo 100H. Nella terminalogia del CP/M, il programma è chiamato "transitorio" e può usare le risorse del sistema operativo eseguendo un JMP all'indirizzo 05H. Questo indirizzo è la porta per il sistema operativo. È un punto di ingresso singolo e fisso, indipendente dall'estensione effettiva della memoria e produce il passaggio del controllo al CP/M. Le routines effettive del CP/M sono residenti nella parte finale della memoria (Fig. 5.2). La chiamata al sistema operativo deve essere accompagnata da un parametro che specifica il servizio richiesto. Questo parametro è passato come numero di funzione ed è contenuto nel registro C dell'8080 e dello Z80. Ventisette codici sono forniti dal CP/M 1.4 (36 dal CP/M 2.2) per accedere ai vari dispositivi di I/O, compresi i files su disco.

Dal punto di vista dell'utente del CP/M, il sistema lavora nella maniera seguente: il CCP (gestore dei comandi di console) stampa il prompt di sistema e attende un comando. Le trasmissioni effettive sono gestite dal BIOS (sistema di ingresso/uscita di base) e la linea di comando finisce nel buffer del CCP.

Quando il CCP riceve il comando (e i nomi dei files associati), esegue il comando immediatamente se il comando è interno (cioé, risiede permanentemente nella porzione di sistema della memoria del calcolatore). Se il comando non è interno, il CCP assume che sia un comando transitorio con un nome di file del tipo "COM" (ad esempio PIP.COM) e chiede a BDOS di trovare il file e di trasferirne una copia nella TPA (Area dei programmi transitori), la parte non utilizzata della memoria del calcolatore (cioé, non già usata dai componenti di sistema del CP/M). Il CCP poi costruisce un blocco di controllo del file per il file o i files che devono essere elaborati dal comando (o programma).

Se vi è più di un nome di file, il CCP costruisce un blocco di controllo di file per ciascun file, ma presuppone che il nuovo programma nella TPA gestirà la ricerca e l'accesso ai files. Il CCP cerca soltanto il primo file chiamando il BDOS e chiedendogli una copia del file.

Il CCP a questo punto termina liberando così spazio nella memoria del calcolatore: il programma transitorio può ora ricoprire lo spazio usato dal CCP.

Dal punto di vista del sistema, tutti i files appaiono allo stesso modo. Il BDOS trova il file guardando nel blocco di controllo del file creato dal CCP. Questo FCB è pieno a metà e contiene soltanto il nome del file. Ciascun file sul dischetto ha il suo proprio FCB (una copia dell'ultimo creato dal CCP), cosicché il BDOS semplicemente verifica la corrispondenza tra il nome di file contenuto nel FCB del CCP e i nomi di file contenuti negli FCB dei files.

Quando il BDOS trova il file corretto, fornisce ulteriori informazioni per il blocco di controllo del file (FCB) creato dal CCP. Ogni volta che il programma accede a quel file particolare, il BDOS cambia alcune delle informazioni nel blocco di controllo o del file. Quando il programma chiude il file, il BDOS fa un cambiamento finale e poi copia l'FCB contenuto nella memoria del calcolatore (insieme alle informazioni modificate del file) sul dischetto, aggiornando sia il file che la copia dell'FCB del file.

Il BDOS assume che tutti i files siano equivalenti e siano identificati da un nome di file. Il CCP guarda la prima parola digitata (dopo il prompt di sistema) e assume che sia o un comando interno o un comando (programma) che risiede su un dischetto con un nome di file che è il nome del comando, con l'estensione "COM" (ad esempio, PIP.COM, LOAD.

.COM, ecc.). Se la prima parola non è un comando interno e il CCP non trova un file del tipo COM con un nome che corrisponde alla prima parola, il CCP ristampa la parola che era stata digitata e la fa seguire da un punto interrogativo.

Se un'estensione 'COM' è assegnata a un file che non è un vero programma, e il nome di quel file è poi digitato come comando, il CCP cercherà di eseguire il file come se fosse un programma.

Il CCP guarda come ogni nome di file associato che possa essere digitato con il comando, ma non li considera né decide come usarli. Appena inizia l'esecuzione del comando (o programma) il comando (programma) stesso decide cosa fare con i files associati. Per esempio, il comando ASM è un programma transitorio che lavora soltanto su files con estensione 'ASM'. Il comando LOAD è un altro programma transitorio che si aspetta di dover caricare un file del tipo 'HEX'. Il comando LOAD produce un file 'COM' dal file di programma in linguaggio macchina 'HEX'.

Per comunicare con i dispositivi (periferiche), il sistema usa un "portatore di messaggi", il BIOS. Il BIOS realizza un'operazione semplice come "leggi un carattere dalla tastiera del terminale" e "scrivi un carattere sulla stampante". Per il programmatore del sistema e l'installatore, il BIOS è la parte del sistema che dev'essere modificata per adattarsi all'ambiente hardware. La Digital Research fornisce un BIOS che funziona con un Intel MDS-800 con i dispositivi standard che si collegano all'MDS-800 (corrispondenti alle definizioni delle periferiche dell'ambiente hardware dell'Intellec MDS). Per far funzionare questa versione del CP/M in un altro ambiente hardware, il programmatore di sistema deve modificare soltato il modulo BIOS.

FDOS E OPERAZIONI DEL CCP

Organizzazione generale

La mappa della memoria del CP/M (dopo che è stato caricato nella memoria principale del calcolatore) è mostrata in Fig. 5.6. Il punto di ingresso principale al FDOS è alla locazione *boot* + 0005H nella "pagina 0", l'area della memoria principale che il sistema riserva per i valori di sistema. La locazione *boot* contiene le istruzioni iniziali del codice macchina che realizza un rilancio del sistema (partenza a caldo); pertanto, i programmi transitori devono soltanto saltare a *boot* per richiamare il CCP e rientrare nelle operazioni del CP/M. Il valore esatto per *boot*, *tbase*, *cbase* e *fbase* dipende dalla versione del CP/M, ma sono sempre utilizzati questi indirizzi - un programma utente (un programma transitorio) o un comando transitorio riempie la memoria da *tbase* e *cbase* e possibilmente a *fbase* se ricopre il CCP.

Quando il CCP riceve una riga di comando, cerca un file COM il cui nome corrisponde al comando, a meno che il comando sia interno, nel qual caso lo esegue immediatamente. Il CCP costruisce anche i blocchi di controllo del file per ogni nome di file che appaia nella riga di comando. Dopo che il CCP ha terminato, l'immagine del file COM è portata nella TPA (eventualmente anche ricoprendo il CCP), dove può accedere alle operazioni del FDOS.

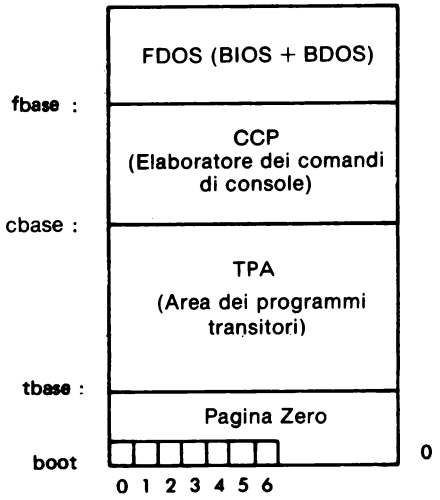


Figura 5.6: Ingresso all'indirizzo fbase effettivo (estensione della TPA disponibile + spazio del CCP). FDOS è un salto a fbase.

L'FDOS è diviso in due parti che sono già state introdotte: il BIOS, che controlla tutte le periferiche e le trasmissioni, e il BDOS, che cerca i files su disco e gestisce i blocchi di controllo del file per ciascun file al fine di fornire gli accessi casuali. Esaminiamo ora entrambe le parti più dettagliatamente.

Operazioni del BIOS

Si accede al BIOS e al BDOS attraverso il punto di ingresso principale dell'FDOS. Un'operazione di BIOS è specificata passando un numero di funzione e un indirizzo dell'informazione. Per esempio, se si deve mandare al terminale il carattere ASCII 'B', il numero di funzione per l'operazione "scrivi su console" (funzione 2) deve essere posto nel

registro C e il valore di 'B' deve essere posto nella copia di registri D, E.

Le operazioni del BIOS sono riassunte qui di seguito , ciascuna con il numero di funzione. (È prudente verificare sulla documentazione della Digital Research per aggiunte e variazioni).

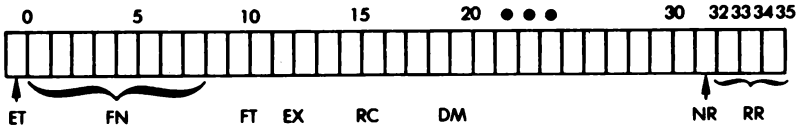
1. Legge dalla console (terminale). Fornisce un carattere ASCII.
2. Scrive sulla console (terminale). Manda un carattere ASCII.
3. Legge dal lettore. Fornisce un carrattere ASCII ricevuto dal dispositivo "lettore" (RDR:).
4. Scrive sul perforatore. Manda un carattere ASCII al dispositivo "perforatore" (PUN:).
5. Scrive sul dispositivo di stampa. Manda un carattere ASCII al dispositivo di "stampa" ("list") (LST:).
6. Dirige le operazioni di ingresso/uscita su console (soltanto CP/M versione 2.2 e MP/M). Manda 'FF' per ricevere un carattere o lo stato, o manda un carattere alla console.
7. Riceve lo stato I/O. Fornisce un byte con lo stato del dispositivo.
8. Assegna lo stato di I/O. Manda un byte con lo stato al dispositivo.
9. Stampa un buffer. Manda un'intera stringa (a partire da un indirizzo fino a incontrare un '$'.
10. Legge un buffer. Manda un indirizzo del buffer di lettura e rientra con il buffer pieno.
11. Interroga la console se è pronto un carattere. Se il bit meno significativo del byte è 1, è pronto un carattere di console.

Blocchi di controllo dei files

Quando il BDOS crea un file, costruisce anche il blocco di controllo del file con le informazioni che comprendono il nome del file e la locazione di memoria. Ciascun blocco di controllo del file può descrivere 16K bytes, corrispondenti a 128 records di 128 bytes ciascuno. Questa unità è chiamata "estensione". Fino a 256K bytes possono essere indirizzati con un solo blocco di controllo del file usando i meccanismi automatici del CP/M.

I blocchi di controllo dei files sono memorizzati insieme ai files sul dischetto e il BDOS li porta nella memoria principale prima di accedere effettivamente ai files (operazioni di "apertura file" e "creazione file" del BDOS). Il BDOS e il programma transitorio aggiornano continuamente i blocchi di controllo dei files durante l'esecuzione. Quando il programma transitorio esegue l'operazione di chiusura del file del BDOS, questo registra la versione finale del blocco di controllo del file sul dischetto. Così, tutte le modifiche su ciascun file avvengono nella memoria principale del computer, senza alterare i files originali che rimangono sul disco.

Il CCP costruisce i blocchi di controllo transitori dei files per i nomi dei files di una riga di comando, e lascia in bianco la maggior parte delle informazioni che saranno fornite dal comando (programma) transitorio. Il CCP confronta il nome di file trovato nella riga di comando (memorizzata in un buffer di lettura per il CCP) con i nomi di files trovati dal BDOS nei blocchi di controllo dei files, allo scopo di trovare le informazioni corrette sull'allocazione di memoria (cioé, trovare le estensioni di un file). Il CCP traduce sempre i caratteri minuscoli nella riga di comando in maiuscoli per conformarsi alle convenzioni dei nomi di files del CP/M.



Campo	Posizioni	Informazioni
ET	0	Tipo di ingresso (sempre zero nel CP/M 1.4, contenente il "codice di unità" nel CP/M 2.2 e nell'MP/M).
FN	1-8	Nome del file completato con spazi (caratteri ASCII).
FT	9-11	Tipo del file, chiamato anche estensione del nome del file (anch'esso completato con spazi).
EX	12	Estensione del file, normalmente assegnata a zero.
	13-14	Riservate ad uso del sistema.
RC	15	Contatore di record, o dimensione dell'estensione corrente (da 0 a 128 records).
DM	16-31	Mappa di allocazione sul disco, riempita e usata dal CP/M.
NR	32	Numero del record successivo da leggere o scrivere (Numero del record corrente nel CP/M versione 2.2).
RR	33-35	Numero opzionale dei records casuali (soltanto CP/M versione 2.2 e MP/M) nel campo 0-65535.

Figura 5.7: Informazioni delle posizioni dei campi

Operazioni del BDOS

Si può accedere al BDOS anche usando l'ingresso del FDOS e passando un numero di funzione e un indirizzo delle informazioni (il punto di ingresso del FDOS è *boot* + 0005H). Per esempio, se si deve fare una

lettura in un file su disco, il programma deve mandare il numero di funzione per la lettura su disco (20 o 33), insieme con l'indirizzo del blocco di controllo del file che si vuol leggere. Il BDOS realizza la funzione e poi rientra o con un indicatore successo o con un indicatore di errore (che indica che la lettura non ha avuto successo).

Le operazioi del BDOS e i loro numeri di funzione sono qui riassunti. (Ancora, si verifichino sulla documentazione della Digital Research le aggiunte e i cambiamenti).

12. Solleva la testina del disco (versione 1.4). La funzione solleva la testina dal disco corrente. Restituisce il numero di versione (versione 2.2 e MP/M). Restituisce il numero di versione del sistema CP/M per permettere una programmazione indipendente dalla versione.
13. Inizializza il disco di sistema. Questa funzione inizializza BDOS, azzera lo stato di lettura/scrittura per tutti i dischi, seleziona l'unità A e assegna l'indirizzo di default DMA a *boot* + 0080H (usato dai programmi per cambiare i dischi senza richiedere un ↑C dal terminale o un rilancio del sistema).
14. Seleziona il disco. È possibile designare (1 per A, 2 per B, 3 per C, ecc.) un'unità a disco come unità corrente per le successive operazioni sui files.
15. Apre un file. Mandando l'indirizzo del blocco di controllo del file, il BDOS cerca un blocco di controllo di un file corrispondente nell'area di direttrice del disco, e restituisce il codice associato di direttrice che indica che l'informazione corrispondente è stata copiata nel blocco di controllo del file. Questo permette i successivi accessi al file.
16. Chiude un file. Mandando l'indirizzo del blocco di controllo di un file, il BDOS registra le nuove informazioni di direttrice nel blocco di controllo del file sul disco (l'opposto di un'operazione di apertura di file, con la restituzione degli stessi codici).
17. Cerca un file. Mandando l'indirizzo del blocco di controllo del file che contiene un nome di file, il BDOS cerca la prima corrispondenza per quel nome di file e restituisce l'indirizzo del blocco di controllo del file su disco che corrisponde a quello fornito dal CCP (o dal programma).
18. Cerca l'occorrenza successiva. Usando questa funzione *dopo* la funzione 17 per cercare l'occorrenza successiva di un nome di file che corrisponda a quello assegnato, restituisce l'indirizzo del blocco di controllo del file successivo sul disco.
19. Cancella un file. Mandando l'indirizzo del blocco di controllo del file che contiene il nome del file, il BDOS cancellerà quel file dal disco.

20. Legge sequenzialmente. Se il file è stato aperto o attivato da una funzione di generazione, questa funzione legge i successivi 128 bytes (record) nella memoria all'indirizzo corrente del DMA e restituisce un parametro che indica: lettura con successo, fine del file, o dati non scritti durante l'accesso casuale.
21. Scrive in sequenza. Se il file è stato aperto o attivato da una funzione di generazione, questa funzione scrive 128 bytes a partire dall'indirizzo corrente del DMA nel file indicato nel blocco di controllo. Questa funzione ricopre i dati esistenti nel file (se ve ne sono).
22. Genera un file. Simile alla funzione di apertura di un file, questa funzione crea un nuovo file nell'aprirlo. Mandando l'indirizzo di un blocco di controllo di un file con un nuovo nome di file, questa funzione creerà il file e inizializzerà il suo blocco di controllo (nella memoria principale come sul disco) come file vuoto. Dovete assicurarvi di non creare un duplicato di un file sullo stesso disco, rendendoli entrambi inaccessibili. Provate usando la funzione di cancellazione prima sul nuovo nome di file.
23. Cambia il nome a un file. Mandando l'indirizzo del blocco di controllo del file, il BDOS cambierà il nome nell'area del nome del file del blocco e la registrerà sul disco.
24. Restituisce il vettore delle connessioni. Questa funzione determina quali unità a disco sono "attivate" (on-line).
25. Restituisce il disco corrente (soltanto versione 2.2 e MP/M). Questa funzione restituisce un numero corrispondente alla lettera (A, B, C, ecc.) dell'unità a dischi che è selezionata correntemente.
26. Assegna l'indirizzo di DMA. Questa funzione assegna all'indirizzo di accesso diretto alla memoria (indirizzo dove il puntatore del file si ferma dopo un'operazione di lettura o scrittura) un altro valore, per trovare records di dati ovunque in memoria. Una partenza a freddo, a caldo o un rilancio del disco assegnerà al DMA il valore *boot* + 0080H.
27. Restituisce l'indirizzo del vettore di allocazione. Il sistema conserva un vettore di allocazione nella memoria principale per ciascuna unità attiva. Questa funzione restituisce l'indirizzo del vettore per l'unità corrente. Molti programmi (come STAT) usano questo vettore per determinare la quantità di spazio che rimane in memoria.
28. Protegge il disco contro la scrittura. Questa funzione fornisce una protezione temporanea contro la scrittura. Ogni tentativo di scrivere sul disco (senza che intervenga una partenza a freddo o a caldo) genererà un messaggio di errore.
29. Restituisce il vettore di sola lettura. Questa funzione restituisce un

vettore che indica quali unità sono protette contro la scrittura; cioé, hanno attivato il bit di sola lettura.

30. Assegna gli attributi dei files. Questa funzione vi permette di assegnare o togliere gli indicatori collegati ai files che costituiscono gli attributi di sola lettura.
31. Restituisce l'indirizzo del blocco di parametri del disco. Questa funzione restituisce l'indirizzo del blocco di parametri del disco di BIOS ed è utile per calcolare lo spazio e cambiare i valori dei parametri del disco quando cambia l'ambiente del disco.
32. Fornisce o assegna il codice utente. Potete usare questa funzione per trovare o cambiare il codice utente correntemente attivato (le aree utente sono soltanto nella versione 2.2 e nell'MP/M).
33. Legge ad accesso casuale. Questa funzione usa il campo RR del blocco di controllo del file per selezionare un numero di record e leggere il record. Rientra con il DMA che punta al record desiderato e il numero di record *non* è aggiornato come di un'operazione di lettura sequenziale.
34. Scrive ad accesso casuale. Questa funzione viene usata allo stesso modo di un'operazione di lettura ad accesso casuale, tranne che scrive i dati nel disco dal DMA corrente. Se lo spazio sul file non è stato ancora assegnato, l'operazione lo assegna prima di scrivere. Il numero di record *non* è aggiornato.
35. Calcola lo spazio sul file. Mandando a questa funzione l'indirizzo del blocco di controllo di un file, restituisce l'indirizzo del "record logico" che segue la fine del file (spazio virtuale del file). Potete aggiungere dati a un file esistente usando quest'informazione per assegnare la posizione del record ad accesso casuale prima di realizzare una serie di operazioni di scrittura ad accesso casuale. Lo spazio virtuale corrisponde allo spazio fisico se il file è stato scritto sequenzialmente; altrimenti il file può avere dei "buchi" non assegnati come risultato di operazioni di scrittura ad accesso casuale.
36. Assegna la posizione del record ad accesso casuale. Questa funzione restituisce la posizione del record ad accesso casuale dopo una serie di operazioni di lettura o scrittura sequenziali. È utile per passare dalle operazioni sequenziali sul file a quelle ad accesso casuale o per la scansione sequenziale del file prima delle operazioni di lettura o scrittura ad accesso casuale.

L'INSTALLAZIONE E LA MODIFICAZIONE DEL CP/M

Il CP/M è sempre ritagliato per una configurazione specificata di ingresso/uscita (e memoria). La Digital Research distribuisce una forma

del CP/M che funziona immediatamente sul sistema di sviluppo a microcalcolatore MDS-800 dell'Intel. Altri venditori di hardware e di software forniscono versioni del CP/M che funzionano su altri sistemi harddware. Probabilmente, comprerete una versione del CP/M che funziona automaticamente senza modifiche. Comunque, se avete una versione del CP/M e volete ritagliarla per un nuovo ambiente hardware a causa di nuovi dispositivi di ingresso/uscita, dovrete correggere la porzione BIOS del CP/M. Se avete l'MP/M, dovrete correggere le porzioni BIOS e XIOS. Modificare il BIOS significa inserire nuovi programmi di ingresso/uscita richiesti dai vostri dispositivi specifici. Questo non è un compito difficile, ma dipende dal dispositivo e dall'installazione. Per questa ragione, non si possono presentare qui istruzioni specifiche. Fate riferimento alla versione applicabile della *CP/M Alteration Guide* della Digital Research.

Se dovete costruire una nuova versione del CP/M da quella di partenza, il problema è più complesso. Se il vostro sistema possiede già gli elementi rudimentali per lo sviluppo e l'esecuzione dei programmi, potete scrivere i vostri programmi (chiamati GETSYS e PUTSYS) per leggere il "sistema" su un nuovo dischetto da usarsi come dischetto di sistema. Altrimenti, dovete usare un altro sistema per generare il nuovo dischetto di sistema da usare nel vostro.

Se avete una versione del CP/M attiva e funzionante, potete scrivere facilmente dei programmi in linguaggio assembler per realizzare dei compiti specifici; potete anche fare uso di SYSGEN.COM e MOVCPM .COM per aiutarvi a modificare il CP/M in modo da non dover scrivere voi i programmi GETSYS e PUTSYS. Comunque, la vostra versione di SYSGEN.COM potrebbe non funzionare con il vostro tipo di dischi o dischetti (minidischetti, dischi rigidi e altri). Potreste aver bisogno di modificare il CP/M prima ancora di poter usare SYSGEN.COM.

La Digital Research fornisce versioni minime di programmi GETSYS e PUTSYS nella documentazione fornita (*CP/M Alteration Guide* o *MP/M User's Guide*). Per incominciare, dovete scrivere un programma GETSYS per leggere le prime due tracce del dischetto fornito (i files delle prime due tracce non vengono stampati da DIR ma costituiscono il sistema stesso). Potete trovare la porzione BIOS del sistema e cambiarla. Potete salvare il sistema modificato sul dischetto scrivendo un programma PUTSYS. Infine, potete scrivere una versione di GETSYS, un programma di "bootstrap", e collocarlo sulla traccia 0, settore 1 usando il vostro programma PUTSYS. Dopo averlo verificato, dovreste avere un sistema che funziona correttamente e parte in modo automatico quando fate partire "a freddo" il vostro calcolatore.

Se usate un sistema CP/M per modificare un sistema CP/M per un altro ambiente hardware e il supporto a dischi è compatibile, allora potete usare delle scorciatoie per creare il nuovo dischetto di sistema:

MOVCPM (MOVCPM.COM) e SYSGEN (SYSGEN.COM). Questo è chiamato una "rigenerazione di sistema di secondo livello" nella documentazione della Digital Research.

Potete realizzare una configurazione di memoria (MOVCMP) e un'inizializzazione del dischetto (PUTSYS) usando MOVCPM invece di GETSYS per leggere il sistema esistente e SYSGEN invece di PUTSYS per collocare la versione modificata sul nuovo dischetto di sistema.

MOVCPM è il comando transitorio MOVCPM.COM, e dovete fornire gli argomenti:

```
MOVCPM bb *
```

dove bb è il numero di kilobytes (cioé, 32K, 64K, 20K) per la nuova immagine di memoria del sistema. Fornite un asterisco (*) per dire a MOVCPM di lasciare quest'immagine di memoria in memoria (la TPA). Potete fornire anche un asterisco al posto di bb e MOVCPM calcolerà la più grande quantità di memoria che può dedicare al nuovo sistema. Il comando MOVCPM è descritto con maggiore dettaglio nella parte successiva di questa sezione.

Ecco un esempio di un'operazione di MOVCPM:

```
A > MOVCPM 32* ↵
CONSTRUCTING 32K CP/M VERS x.x
READY FOR "SYSGEN" OR
"SAVE 34 CMP32.COM"
A >
```

Come suggerisce la stampa, il nuovo sistema a 32K è nella TPA, pronto per l'operazione successiva, che dovrebbe essere o un SYSGEN o una SAVE. Poiché volete modificare la porzione BIOS del sistema, vorrete anche salvare la versione (chiamandola 'CPM32.COM' per un sistema a 32K se volete). Una volta che è stata salvata, potete caricarla ancora in memoria usando DDT (programma debugger del CP/M) e modificare la porzione BIOS.

Se pensate di fare modifiche più estese (o modifiche in un altro momento) sarebbe più semplice creare il vostro BIOS, dal momento che dovete anche creare il vostro programma bootstrap (potete chiamarli 'CBIOS' e 'BOOT' se volete). Potete usare il programma ED (programma editor del CP/M) per creare CBIOS.ASM e BOOT.ASM e usare il programma ASM (l'assembler del CP/M) o un altro assembler per creare CBIOS.HEX e BOOT.HEX, che può essere caricato usando

LOAD per creare CBIOS.COM e BOOT.COM programmi effettivi che possono essere verificati prima di inserirli nel nuovo sistema.

Quando avete il nuovo sistema CPMbb.COM (dove bb è l'estensione di memoria usata con MOVCPM) in memoria per mezzo di DDT, potete inserire CBIOS.COM e BOOT.COM in esso e modificare queste porzioni verificandole e infine usare SYSGEN per mettere il sistema modificato sulle prime due tracce del nuovo dischetto di sistema. Ecco un esempio:

```
A > SYSGEN ↵
SYSGEN VERSION xx.xx
SOURCE DRIVE NAME (OR RETURN TO SKIP) ↵
```

(Rispondere con un RETURN per saltare l'operazione di lettura di SYSGEN, poichè avete già il sistema modificato in memoria).

```
DESTINATION DRIVE NAME
(OR RETURN TO REBOOT) B
```

(Si assume che il nuovo dischetto di sistema sia nell'unità B).

```
DESTINATION ON DRIVE B, THEN TYPE RETURN ↵
FUNCTION COMPLETE
DESTINATION DRIVE NAME
(OR RETURN TO REBOOT) ↵
A > PIP B: = A: *.* [V] ↵
• • •
```

(Messaggi di copiatura)

```
• • •
A >
```

Se avete una copia del CP/M come file .COM sul disco, potete usare una scorciatoia:

```
A >SYSGEN CPM.COM ↵
   SYSGEN VERSION 2.2
   DESTINATION DRIVE NAME
   (OR RETURN TO REBOOT):
   (ecc..)
```

Dovreste seguire le istruzioni fornite dalla *CP/M Version 2.2 Alteration Guide* della Digital Research, per modificare BIOS e creare i programmi GETSYS, PUTSYS, e BOOT.

LA RICONFIGURAZIONE (SISTEMAZIONE DELLO SPAZIO DI MEMORIA) CON MOVCPM

Frequentemente lo spazio di memoria di un sistema viene aumentato o ridotto (togliendo porzioni di memoria). Il CP/M deve essere modificato di conseguenza. Il programma MOVCPM (MOVCPM.COM) permette all'utente di riconfigurare in modo semplice il sistema CP/M per ogni dimensione di memoria.

Il programma MOVCPM può essere eseguito come comando con argomenti opzionali:

MOVCPM { * / bb } *

L'argomento opzionale bb dice a MOVCPM di quanta memoria dispone il nuovo sistema CP/M; se non specificate bb o se specificate un asterisco (*) invece di bb, MOVCPM configurerà il nuovo sistema in modo che gestisca *tutta* la memoria RAM disponibile del vostro calcolatore ospite (RAM è "random access memory", cioé memoria ad accesso casuale). Nella maggior parte dei casi, vorrete che il CP/M gestisca (e disponga) di tutta la memoria RAM disponibile.

Il secondo asterisco opzionale (*), se lo fornite, dirà a MOVCPM di mantenere il sistema appena configurato in memoria (la TPA), in previsione di una SYSGEN o una SAVE. Nella maggior parte dei casi vorrete salvare questo nuovo sistema o scriverlo su un dischetto usando SYSGEN. Se non fornite il secondo asterisco (*), MOVCPM farà il bootstrap di questo nuovo sistema senza registrarlo in modo permanente.

Ecco alcuni esempi:

A > MOVCPM ↵ — Questo comando rialloca il sistema CP/M in modo da utilizzare tutta le memoria RAM (a partire da 100H, all'inizio della TPA) nel calcolatore ospite e poi eseguire il sistema senza registrarlo sul dischetto.

A > MOVCPM 32 ↵	Questo comando rialloca CP/M in modo da gestire 32K di memoria ed eseguire il sistema senza registrarlo.
A > MOVCPM 32 * ↵	Questo comando rialloca il CP/M in modo da gestire 32K di memoria e lascia l'immagine di memoria del sistema in memoria (la TPA) in preparazione per un SYSGEN o un SAVE.
A > MOVCPM ** ↵	Questo comando alloca il CP/M in modo da gestire tutta le memoria RAM del calcolatore ospite (a partire da 100H) e lascia l'immagine di memoria del sistema in memoria in previsione di un SYSGEN o SAVE.

Le ultime due forme usano il secondo asterisco (*) per lasciare la nuova versione del sistema in memoria in modo da poter o salvare il contenuto della memoria in un file su disco o conservare il sistema in memoria per scriverlo sulle prime due tracce del dischetto (per creare un dischetto di sistema). Quando si esegue o 'MOVCPM ** ↵ o 'MOVCPM bb * ↵', viene stampato il messaggio READY FOR "SYSGEN" o "SAVE bb CPMbb.COM" al termine del programma MOVCPM (bb è il numero di kilobytes).

La riga di comando che fornite potrebbe contenere il programma "menu" che provvede all'accesso ad altri programmi concatenandoli ai programmi precedenti in dipendenza dal programma scelto dall'utente del "menu". Ogni volta che termina un programa, dopo ogni partenza a freddo o a caldo, sarebbe ristampato il menu e l'utente potrebbe fare un'altra selezione. Qualora si usasse questa versione del sistema, l'utente non sarebbe in grado di eseguire comandi CP/M, poiché dopo ogni esecuzione o interruzione di programma, il sistema automaticamente eseguirebbe di nuovo il programma del menu.

Per inserire la vostra riga di comando per l'esecuzione automatica, portate il sistema usando MOVCPM o SYSGEN e salvatelo (comando SAVE). Quando avete salvato l'immagine di memoria del sistema, potete usare DDT per inserire le modifiche. La locazione per la modifica dovrebbe incominciare all'indirizzo 0980H; usando il comando D di DDT, potete stampare il contenuto della memoria in quella locazione:

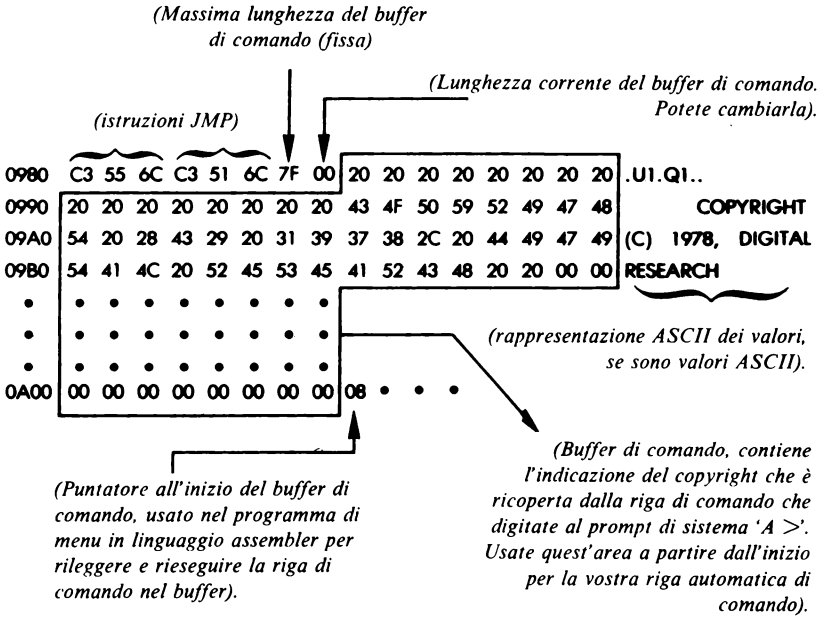


Figura 5.8: Stampa del contenuto della memoria

UN ESEMPIO DI MODIFICHE DEL CP/M: UN SISTEMA A MENU

Si riporta quest'esempio per mostrare come usare le risorse del CP/M. Un'applicazione pratica interessante è un sistema a "chiavi in mano" che parte immediatamente a eseguire i programmi quando date una partenza a freddo (ad esempio premendo il bottone RESET, il sistema esegue automaticamente un altro programma che in sequenza si concatena ad altri programmi per realizzare operazioni di gestione o di controllo). Un altro strumento utile per ambienti commerciali o gestionali è un *programma a menu* (menzionato precedentemente) che dà all'utente una scelta di programmi da eseguire; questo programma a menu sarà eseguito automaticamente in un sistema "chiavi in mano" per mantenere la protezione sui programmi di sistema.

Potete provvedere a entrambe le cose modificando locazioni speciali nel CP/M - le locazioni usualmente riempite dal comando digitato quando ricevete il prompt di sistema. Queste locazioni sono la base del CCP (elaboratore dei comandi di console) e potete modificarlo usando DDT (o qualsiasi altro debugger , come il SID). Essenzialmente, inserite

una riga di comando in queste locazioni e un "flag" (segnale) per dire al CP/M di eseguire la riga di comando invece di stampare il prompt di sistema. Allora, quando l'utente esegue una partenza a freddo o a caldo (rilancio del sitema o ↑ C), la riga di comando inserita viene eseguita automaticamente. I valori nella stampa di memoria sono esadecimali. Ciascun carattere ASCII ha un valore esadecimale - la 'C' di 'COPYRIGHT' ha il valore 43H ('H' significa esadecimale), la 'O' di 'COPYRIGHT' ha il valore 4FH, e uno spazio ha il valore 20H (il buffer di comando incomincia con molti spazi). Questi valori possono essere trovati in una tabella ASCII.

Osservate il valore 00 come "lunghezza corrente del buffer di comando", dopo il valore 7F come "lunghezza massima del buffer di comando", sulla riga di indirizzo 0980. Questo valore 00 dice al CP/M che al momento non c'é alcun comando nel buffer (in conseguenza di una partenza a freddo o a caldo). Il sistema pertanto stampa il prompt 'A>' e attende che l'utente digiti un comando; la riga di comando che l'utente digita viene memorizzata (a partire dal valore "lunghezza corrente") nel buffer di comando, ricoprendo qualsiasi cosa vi fosse prima (in conseguenza di una partenza a freddo, l'indicazione di copyright occupa il buffer). Questa è una normale operazione CP/M.

Per trasformare il CP/M in un sistema "chiavi in mano", dovete introdurre una condizione *anormale*: se cambiate il valore della "lunghezza corrente del buffer di comando" in un numero diverso da 0, il sistema, dopo una partenza a freddo o a caldo, penserà che c'é *già* un comando nel buffer e eseguirà il comando. A seguito della terminazione o della interruzione di un programma (o di una partenza a caldo o a freddo) il sistema tornerà di nuovo a questo valore e penserà di nuovo che c'é già un comando nel buffer. In breve, il sistema non arriverà mai a stampare il prompt di sistema 'A>'!

Dovete naturalmente fornire una riga di comando per il buffer, ricoprendo l'indicazione di copyright (potete anche spostare copyright alla fine del buffer di comando). Potete usare fino alla locazione 0A07H compresa, ma non potete assolutamente modificare le locazioni che la seguono. La locazione 0A08H contiene il puntatore all'inizio del buffer di comando di cui avrete bisogno se scrivete un programma menu in linguaggio assembler.

Per inserire una riga di comando, digitate il comando 'S' di DDT (come descritto nella documentazione del DDT fornita dalla Digital Research), usando i valori esadecimali per i caratteri ASCII della riga di comando. Questa riga di comando deve terminare con il valore 00 e dovete contare tutti i caratteri (compresi gli spazi) della riga di comando fino allo 00 finale non compreso e mettere questo numero di caratteri nella locazione che contiene la "lunghezza corrente del buffer di comando", per dire al CP/M quant'é lunga la vostra riga di comando.

Un modo semplice per realizzare un sistema menu è usare il BASIC. La maggior parte dei BASIC ha l'istruzione di concatenamento di programmi (CHAIN), cosicché un programma BASIC può concatenarsi a un altro e l'altro può concatenarsi indietro al programma menu. Questo programma menu può essere scritto in BASIC e l'interprete del linguaggio BASIC si curerà del buffer di comando per eseguire programmi BASIC (cioé, non dovrete usare il puntatore all'inizio del buffer di comando nella locazione 0A08H). Si può usare il CBASIC2 (Software Sistems) o il BASIC Microsoft (MBASIC.COM del Microsoft Consumer Products), poichè entrambi permettono di eseguire un programma BASIC come parte di una riga di comando CP/M per lavorare con il BASIC.

Per esempio il BASIC Microsoft è fornito come MBASIC.COM, un programma che potete eseguire da solo per "caricare" il BASIC Microsoft (eseguire l'interprete BASIC) o eseguire *con un argomento* che è il nome di un programma BASIC, come in questo esempio:

```
A > MBASIC PROG ↵
```

In questo esempio, MBASIC è il programma interprete BASIC che a sua volta esegue il programa BASIC PROG.BAS (i programmi in BASIC Microsoft devono avere nel nome l'estensione '.BAS').

Se scrivete in BASIC un programma MANU.BAS, potete usarlo nella riga di comando seguente:

```
MBASIC MENU
```

Potete inserire questa riga nel buffer di comando. La lunghezza corrente del buffer di comando sarà 11, così dovrete mettere il valore 0B (esadecimale) nella locazione che contiene la "lunghezza corrente del buffer di comando" e dovrete poi inserire la riga di comando "MBASIC MENU" nel buffer di comando come mostrato nella Fig. 5.9.

Quando terminate di inserire la modifica riportata sopra, dovete eseguire il comando GO del DDT per terminare il DDT. Senza fare nulla che distrugga il contenuto della TPA, usate il comando SAVE per salvare la nuova versione del sistema:

```
A > SAVE 34 AUTOCPM.COM ↵
```

(chiamate la nuova versione con un nome del tipo 'AUTOCPM.COM' per distinguerla dalle altre versioni del sistema).

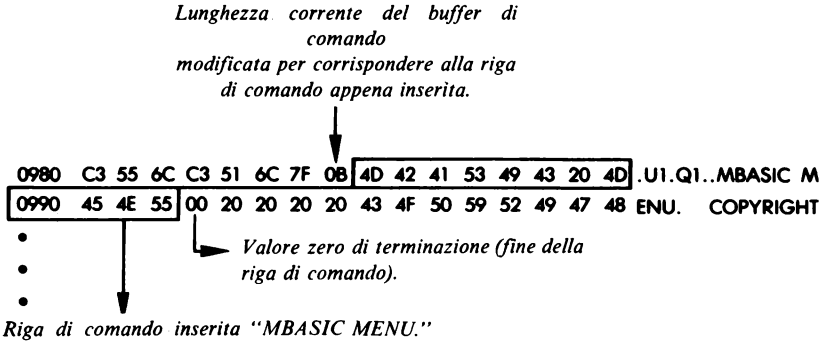


Figura 5.9: Stampa di memoria della riga inserita

Se usate il CP/M versione 1.4, dovete *prima* eseguire la seguente operazione SYSGEN per mettere il sistema appena modificato sulle prime due tracce del nuovo dischetto di sistema, *poi* fate l'operazione SAVE riportata sopra (perché il comando SAVE della versione 1.4 distrugge il contenuto della TPA). Se usate la versione 2.2 del CP/M, potete tranquillamente usare il comando SAVE prima (per salvare un'immagine di memoria del nuovo sistema) e poi usare SYSGEN per mettere il nuovo sistema sulle prime due tracce del nuovo dischetto di sistema:

A > SYSGEN ↵

SOURCE DRIVE NAME (OR RETURN TO SKIP) ↵

(Premete RETURN per saltare, poichè il sistema appena modificato è già in memoria).

DESTINATION DRIVE NAME
(OR RETURN TO REBOOT) B

(Si assume che il nuovo dischetto di sistema sia nell'unità B).

DESTINATION IN DRIVE B, THEN TYPE RETURN ↵

FUNCTION COMPLETE

DESTINATION DRIVE NAME
(OR RETURN TO REBOOT) ↵

(Premete RETURN per terminare il programma SYSGEN).

A >

A questo punto tutto quello che dovete fare è copiare i files appropriati (files di comando CP/M, MBASIC.COM e MENU.BAS) sul nuovo dischetto di sistema. Dopo la partenza a freddo, il sistema eseguirà MBASIC e MENU.BAS, invece di dare il solito prompt di sistema. Il programma MENU.BAS deve essere in grado di consentire l'accesso (per mezzo delle istruzioni CHAIN o SWAP) ad altri programmi BASIC.

Se volete scrivere il programma MENU in linguaggio assembler, questo deve essere abbastanza intelligente da costruire un'altra riga di comando nel buffer di comando per saltare ad altri programmi. Questo programma deve usare il "puntatore all'inizio del buffer di comando" nella locazione 0A08H per dire al sistema di tornare a leggere (ed eseguire) la riga di comando ricostruita.

MP/M

Installazione e modificazione dell'MP/M

Per installare un sistema multiutente MP/M, avete bisogno di un sistema CP/M, perché il caricatore dell'M/MP (MPMLDR.COM) deve contenere una versione del BIOS del CP/M (LDRBIOS.COM). L'MP/M può essere eseguito ("caricato") dal CP/M eseguendo MPMLDR.COM, che carica il sistema MP/M (MPM.SYS) nella memoria dal dischetto (in modo simile a SYSGEN.COM, che carica un sistema CP/M in memoria). Dal CP/M, dovete prima generare il sistema MP/M usando il programma GENSYS.COM, che è fornito con l'MP/M e funziona in CP/M.

Il programma GENSYS pone alcune domande e usa le informazioni fornite per costruire MPM.SYS. Il programma MPMLDR carica poi MPM.SYS in memoria e lo riloca automaticamente.

Per rispondere a tutte le domande di GENSYS, dovete conoscere quale tipo di sistema volete, se volete una "memoria a banchi" o no, e quali "processi di sistema residenti" desiderate (i "processi di sistema residenti" sono descritti nella sezione successiva). Ecco un esempio di esecuzione di GENSYS:

```
A > GENSYS ↵
MP/M 1.0 SYSTEM GENERATION
TOP PAGE OF MEMORY = 0 ↵
NUMBER OF CONSOLES = 2 ↵
BREAKPOINT RST # = 5 ↵
ALLOCATE USER STACKS FOR SYSTEM
CALLS (Y/N) ?Y
```

```
MEMORY SEGMENT BASES, (FF TERMINATES LIST)
: 00,0 ↵
: 00,1 ↵
: 00,2 ↵
: FF ↵
Select Resident System Processes: (Y/N) Y
TIME          ?Y ↵
SCHED         ?Y ↵
ATTACH        ?Y ↵
SPOOL         ?Y ↵
MPMSTAT       ?Y ↵
A >
```

Le risposte riportate sopra sono qui descritte:

Top page of memory (pagina superiore di memoria): Introducete l'indirizzo esadecimale della pagina superiore della memoria RAM del vostro sistema. Se introducete uno zero, il caricatore dell'MP/M determina l'estensione della memoria al momento del caricamento cercando la pagina superiore della memoria RAM.

Number of consoles (numero di console): Introducete il numero di consoles che devono essere agganciate al sistema; ciascuna console occupa 256 bytes di memoria. L'MP/M versione 1.0 accetta fino a 16 consoles.

Breakpoint RST #: Introduce il numero della "restart di breakpoint" (il punto di ingresso del codice che gestisce le interruzioni generate per bloccare i programmi nella fase di messa a punto) che deve essere usata dal DDT o dal SID (programmi debugger). La restart zero non è permessa; non sono permesse neanche le restarts usate dal sistema MP/M. Consultate la documentazione dell'MP/M fornita dalla Digital Research.

Allocate user stacks for system calls (assegna lo "stack" utente per i sottoprogrammi di sistema): Rispondete 'Y' se intendete usare i files '.COM' del CP/M nel sistema MP/M. L'MP/M richiede più spazio di "stack" (zona di memoria gestita a "pila") che il CP/M.

Memory segment bases (basi dei segmenti di memoria). Potete specificare da uno a otto segmenti di memoria utente con lo stesso spazio di indirizzi ma differente numero di banco, come descritto nella documentazione dell'MP/M. La prima locazione di memoria che specificate dovrebbe

essere la prima locazione effettiva della RAM (se avete la ROM che inizia a 0000H). Il numero di banco segue la locazione, separato da una virgola. Questa lista è terminata da un 'FF'.

Select resident system processes (selezione dei processi di sistema residenti): Rispondete 'Y' per ogni programma che volete sia "residente" invece che "rilocabile" o "transitorio". I processi residenti sono programmi che risiedono nel sistema operativo e non sono stampati in una stampa di direttrice (in modo molto simile ai comandi interni del CP/M).

Se questa routine sembra complicata, è perché lo è effettivamente! L'MP/M è un concetto nuovo per i microcalcolatori - i sistemi che condividono più utenti non hanno proliferato nel mercato dei calcolatori di basso livello. Come è tipico per le "idee avanzate", questa è troppo complicata per le normali operazioni dei microcalcolatori. Ovviamente, questo processo di generazione diventerà più da realizzarsi nelle versioni successive dell'MP/M. (Molte di queste informazioni possono essere trovate in una forma facilmente modificabile nell'*User's Guide to MP/M* della Digital Research).

Una volta che avete generato l'MPM.SYS potete usare MPMLDR.COM per caricarlo in memoria ed eseguirlo. MPMLDR.COM non richiede risposte - semplicemente viene eseguito come un comando.

L'MP/M è progettato per funzionare su un sistema di sviluppo a microcalcolatore Intel MDS-800, ma ha una porzione chiamata XIOS (BIOS esteso) che potete modificare per altri ambienti hardware. Oltre a riscrivere la porzione XIOS dovete anche adattare il programma MPMLDR.COM per caricare MPM.SYS ed eseguirlo. Tenete presente che MPMLDR.COM usa il BIOS standard del CP/M (chiamato LDRBIOS) per rilocare ed eseguire il sistema MP/M; pertanto, avete almeno bisogno di una versione del BIOS del CP/M (potete scriverne una o modificarne una esistente). Usando la BIOS modificata (LDRBIOS), modificate la porzione BIOS di MPMLDR.COM e rimettetela sul dischetto usando SYSGEN.COM (se non potete usare SYSGEN.COM dovete scrivere i programmi GETSYS e PUTSYS, come descritto nella documentazione dell'MP/M e nella prima sezione di questo capitolo). Per realizzare questa modifica, caricate in memoria MPMLDR.COM usando DDT o SID (programmi debugger) ed eseguite la modifica manualmente oppure inserite il vostro LDRBIOS.HEX in esso. Quando avete finito, salvate il contenuto in MPMLDR.COM con SAVE ed eseguite MPMLDR.COM per iniziare MP/M da un CP/M in esecuzione.

Per costruire la vostra porzione XIOS dell'MP/M, seguite le istruzioni dettagliate nella documentazione dell'MP/M fornita dalla Digital Rese-

arch. Usate il programma GENMOD per produrre il file XIOS.SPR (rilocabile a pagina di sistema) da due files HEX concatenati.

Funzionamento dell'MP/M

L'MP/M comprende molti componenti: lo XIOS (BIOS e IOS esteso) per interfacciare l'ambiente hardware (che può cambiare), il BDOS e lo XDOS (operazioni di base ed estese sul disco) per realizzare le operazioni su disco e CLI/TMP ("Command Line Interpreter and Terminal Message Process", cioé, interprete della riga di comando e processo di messaggio del terminale) per gestire l'ingresso/uscita della console.

CP/M è un sistema "sequenziale" con un solo programma funzionante alla volta. L'MP/M, invece, deve gestire molti programmi che funzionano "nello stesso momento" e suddividere le stesse risorse: la CPU (calcolatore), i supporti a disco, le consoles e le stampanti. L'MP/M è un sistema "a priorità", il che significa che il processo (programma funzionante) con la priorità più elevata ottiene la CPU. Un processo occupa una risorsa fino a quando non ha finito di usarla, o esegue una chiamata al sistema, o è interrotto, o l'orologio di "tempo reale" segnala la scadenza di un tempo elementare (opzionale). La configurazione delle risorse che si viene a determinare è chiamata *"dispatching"*.

Il "dispatcher" guarda i *"descrittori"* dei processi per determinare la loro priorità e decidere se devono funzionare prima di altri processi. Il dispatcher usa anche il descrittore dei processi per memorizzare informazioni temporanee sui processi e lo stato in cui si trovavano quando sono stati interrotti.

Quando tutti i processi hanno uguale priorità, il dispatcher li esegue in ordine "primo arrivato-primo servito" (FiFo). Le *code* sono usate per sincronizzare i processi in quanto si ha un processo che manda un messaggio in un certo momento durante il suo funzionamento, mentre un altro attende sulla coda per quel messaggio. Il processo in attesa è sospeso fino a quando arriva il messaggio.

Una *coda* è una lista di attesa organizzata come un file speciale che può essere aperto e chiuso e a cui si può fornire informazioni in modo sequenziale. Una coda può essere usata per ricevere informazioni temporanee da trasferire a un altro programma che scrive le informazioni sul disco. Le code sono usate anche per trasmettere in sequenza files alla stampante e garantire l'uso esclusivo di una risorsa da parte di un processo. Per esempio, se un processo genera una coda che contiene informazioni soltanto quando la stampante non è occupata e un altro processo deve ricevere informazioni dalla coda prima di accedere alla stampante, allora il meccanismo garantisce l'uso esclusivo della risorsa condivisa. Il secondo processo attenderà fino a quando ha ricevuto il messaggio che la stampante è libera.

L'MP/M fornisce un altro modo di sincronizzare i processi utilizzando *flags* (segnali, letteralmente "bandiere") attivati da un processo ed esaminati da un altro processo. I flags forniscono un metodo di sincronizzazione e di interruzione dei processi che è indipendente dagli altri dispositivi di interruzione hardware e meccanismi di interruzione software.

Come opzione, si può usare un orologio di "tempo reale" (che può essere azzerato) per fornire accurate misure di tempo e temporizzazione del sistema per la gestione dei processi o per l'esecuzione dei programmi su disco. Fornisce anche la possibilità di ritardare l'esecuzione di un processo.

Non ci sono comandi "interni" per l'MP/M; tutti i "comandi" sono o programmi transitori '.COM', o programmi a pagina rilocabile '.PRL', o processi di sistema residenti creati dai programmi '.RSP' al momento della generazione del sistema (esecuzione di GENSYS). I programmi transitori richiedono l'uso dell'area di memoria "TPA assoluta" e richiedono uno spazio di stack addizionale; la maggior parte dei programmi dovrebbero essere "rilocabili" in modo da poter occupare virtualmente *qualsiasi* spazio disponibile in memoria e poter essere ancora eseguiti in modo corretto. La maggior parte dei programmi rilocabili utilizzano *macro* che sono costituiti da molte istruzioni racchiuse in una, per cui avete bisogno di MAC (Programma Macro Assembler venduto separatamente dalla Digital Research) per assemblare i programmi. Il programma GENMOD fornito con l'MP/M, converte due files '.HEX' concatenati in un file rilocabile a pagine con un'estensione '.PRL'. Questo programma '.PRL' viene eseguito in modo corretto se usate bene le istruzioni ORG, come descritto nella *MP/M User's Guide* della Digital Research.

GENMOD accetta un file che contiene due files '.HEX' concatenati distanziati l'uno dall'altro di 100 esadecimale bytes. Il formato del comando GENMOD è:

```
GENMOD file.hex file.prl $bbbb
```

L'argomento file.hex deve essere il nome di un file con l'estensione '.HEX' che contiene due file '.HEX' concatenati distanziati di 100H bytes. L'argomento file.prl deve essere un nome di file con l'estensione '.PRL' per il nuovo programma rilocabile a pagine. L'argomento opzionale $bbbb fornisce la memoria addizionale dopo lo spazio di codice esplicito assegnato, al fine di fornire ulteriore spazio per i buffers. Se il vostro programma richiede questa memoria addizionale, fornite un simbolo di dollaro ($) seguito da quattro esadecimali. Ecco un esempio:

```
1A > GENMOD B:FINAL.HEX PERFORM.PRL $ 1000 ↵
```

Questo comando converte FINAL.HEX sull'unità B in PERFORM.PRL sull'unità corrente e assegna una memoria ulteriore di 1000 bytes al programma.

Potete anche creare il vostro processo di sistema residente usando GENMOD e sostituendo come argomento un file con estensione. RSP al file con estensione. .PRL. Un processo di sistema residente comincia come programma ".RSP"; quando generate l'MP/M usando GENSYS, avete l'opzione di incorporare tutti i files '.RSP' nel sistema come processi di sistema residenti. I files '.RSP' forniti comprendono MPMSTAT, SPOOL, il tempo di sistema (TOD) e lo schedulatore (SCHED). Potete creare un vostro file '.RSP', purché lo facciate rilocabile a pagine, i primi due bytes del file siano riservati per gli indirizzi di BDOS/XDOS e il descrittore di processo sia costruito in accordo alle istruzioni fornite dall'MP/M.

L'MP/M gestisce ciascuna console utilizzando un processo chiamato TMP (Terminal Message Process, cioé processo di messaggio del terminale). Quando viene digitata una riga di comando, il TMP la manda al CLI (Command Line Interpreter, cioé, interprete della riga di comando) dove è analizzata minuziosamente. Il CLI è in effetti un più avanzato CCP (Console Command Processor, cioé elaboratore del comando di console) usato dal CP/M. Il CLI prende la prima parola della riga di comando e cerca di aprire una coda con quel nome, assumendo, dapprima, che sia semplicemente una richiesta di mettere un messaggio in una coda (poiché il CLI gestisce anche quelle operazioni). Se non c'é una coda con quel nome, CLI cerca prima un file. '.PRL' con quel nome; se c'é una coda con quel nome, il resto della riga di comando è copiata nella coda come messaggio. Se il CLI trova un file '.PRL' con quel nome, fa una richiesta in memoria rilocabile in cui caricare e far funzionare il programma. Poi, carica e lancia il programma. Se il CLI non trova un file '.PRL', cerca un file '.COM'; se trova un file '.COM', fa una richiesta di memoria TPA assoluta e carica ed esegue il programma. Se il programma contiene specifiche di files o è seguito da nomi di files nella riga di comando, il CLI crea anche i blocchi di controllo di questi files (come il file CCP nei sistemi CP/M).

Dalla console, potete "distaccare" un programma funzionante (cioé, i risultati del programma non appaiono sul terminale e il terminale è libero) usando ↑ D, al fine di eseguire altri programmi. Quando premete di nuovo ↑ D, il primo processo in attesa sulla console (processo con più alta priorità) rientra. Potete anche usare il programma ATTACH per attaccare la console a un programma specifico. Ecco un esempio:

```
1A > ATTACH PROG1 ↵
                    (PROG1 prende la console)
```

Un programma può essere attaccato soltanto alla console dalla quale era stato distaccato. Potete usare il processo residente MPMSTAT (se generato con il sistema) o il programma MPMSTAT.RSP (se modificato in MPMSTAT.PRL) per disporre di una visualizzazione dei processi nel sistema:

```
1A > MPMSTAT ↵
          * * * * * MPM 1.0 STATUS DISPLAY * * * * *
Ready Process(es):
MPMSTAT     cli     Idle
```

(I processi pronti sono quelli che sono pronti a entrare in esecuzione e sono in attesa di tempo di CPU. Il primo è quello che ha la priorità più alta ed è in esecuzione al momento).

```
Process(es) DQing:
[Sched]          Sched
[ATTACH]         ATTACH
[SPOOL]          Spool
```

(Questi processi sono in attesa di messaggi dalle code che sono tra parentesi quadrate. Sono ordinati dalla priorità più alta a quella più bassa).

```
Process(es) NQing:
```

(Solitamente simile alla visualizzazione precedente, questa mostra che non ci sono processi che in questo momento scrivono sulle code).

```
Delayed Process(es):
```

(Non ci sono processi ritardati, in questo momento. Un processo ritardato è uno che è in attesa di una specifica quantità di colpi di orologio secondo l'unità di tempo del sistema).

```
Polling Process(es):
    PIP
```

(Il processo PIP sta interrogando, il dispositivo di console).

```
Swapped Process(es):
```

(Lo "Swapping" cioé, lo scambio dell'unità centrale fra i processi, non è implementata nella versione 1.0 dell'MP/M).

```
Process(es) Flag Waiting:
01 - Tick
02 - Clock
```

(Questi sono processi che attivano e modificano "flags", cioé segnali, per sincronizzare altri processi).

```
Flag(s) Set:
03
```

(Il flag "intervallo di un minuto" è attivo).

```
Queue(s):
tod SCHED ATTACH STOPSPLER SPOOL MPMSTAT
Cliq Parseq  ListMQ  DiskMQ
```

(Queste sono tutte le code del sistema. Alle code con tutti i caratteri maiuscoli possono essere mandati messaggi dal CLI o dalla console per mezzo del CLI. Per esempio, la coda SPOOL riceve nomi di files digitando 'SPOOL', seguito dal nome del file, sulla console).

```
Process(es) Attached to Consoles:
[0] - MPMSTAT
[1] - PIP
```

(I processi attaccati alla console sono elencati con il numero della console e il nome del processo).

```
Process(es) Waiting for Consoles:
[0] - TMP0      DIR
[1] - TMP1
```

(Questi processi sono in attesa della console dalla quale erano stati distaccati; per esempio TMP0 è in attesa della console 0 e DIR è in attesa

che TMP0 termini di usare la console 0. Poiché TMP0 è il processo di messaggio della console, DIR attenderà fino a che sarà eseguito un ↑D (o ATTACH)).

```
Memory Allocation:
Base = 0000H   Size = 4000H   Allocated to PIP 1
Base = 4000H   Size = 2000H   * Free *
Base = 6000H   Size = 1100H   Allocated to DIR 0
```

(Questa stampa mostra la base, l'estensione e i proprietari dei segmenti di memoria, insieme con il numero di console che ha originato il proprietario in parentesi quadre. I segmenti non allocati sono liberi per l'uso).

SOMMARIO

In questo capitolo sono state spiegate le operazioni interne del CP/M e dell'MP/M. I principi di queste operazioni non sono complessi e dovrebbero essere compresi se si vuole modificare il CP/M.

A questo punto abbiamo imparato tutte le risorse disponibili con il CP/M e l'MP/M. Comunque, questo non significa automaticamente che l'utente ha un'esperienza per usare con successo e in modo effettivo il calcolatore: è richiesta la pratica.

Facendo pratica, apprezzerete il valore delle raccomandazioni presentate nel capitolo seguente.

CAPITOLO 6

GUIDA DI RIFERIMENTO AI COMANDI E AI PROGRAMMI DEL CP/M E DELL'MP/M

INTRODUZIONE

Questo capitolo è una guida rapida di riferimento ai comandi e ai programmi di utilità del CP/M e dell'MP/M (introdotti nei Capitoli dall'1 al 4). Questa guida è organizzata in modo tale che l'utente può cercare la parola chiave di un comando o programma. Queste parole chiave sono presentate in ordine alfabetico.

Descriveremo ora il formato usato in questo capitolo. Il seguente è un esempio di descrittore che appare associato ad ogni comando:

- ● CP/M versione 1.4
- ○ CP/M versione 2.2
- ○ MP/M versione 1.0

Guardando l'esempio sopra, il punto nero vicino a CP/M versione 1.4 dice che il comando o programma si applica a quella versione particolare del CP/M. Il punto vuoto che appare vicino a CP/M versione 2.2 e a MP/M versione 1.0 dice che l'esempio non si applica a queste versioni. Molti comandi e programmi si applicano a tutti e tre i sistemi.

Dopo la descrizione dello scopo di ciascun comando o programma, viene indicata tra parentesi la sua natura: cioé, comando interno, file '.COM', file '.PRL', o processo residente (MP/M).

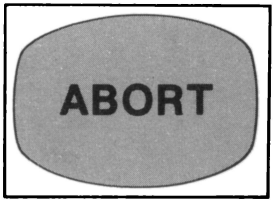
Di seguito, vengono mostrati in un formato conciso i possibili *argomenti* usati nell'esecuzione del comando o programma. Questi *caratteri tipografici* indicano che l'argomento è richiesto (ad esempio *nomefile*), mentre questi caratteri tipografici indicano che l'argomento è opzionale (ad esempio, nomefile). Le parentesi { } indicano una scelta, dove almeno uno degli argomenti è richiesto (a meno che uno degli argomenti sia opzionale all'interno delle parentesi).

In alcuni casi, una parte dell'argomento è opzionale, mentre un'altra parte è richiesta (ad esempio, *u: nomefile* (dove *u:* è opzionale e *nomefile* è richiesto)). In ogni caso, leggete nelle descrizioni degli argomenti sotto i formati.

In questa guida vengono fatte alcune assunzioni generali. Per esempio, si assume che un nome di file che appare come argomento, sia opzionale che richiesto, possa avere all'interno opzionalmente la lettera dell'unità (ad esempio B:FILE) che specifica una unità a dischi alternativa rispetto a quella corrente. Questo è sempre valido, a meno che l'argomento sia definito tale da escludere gli specificatori dell'unità. Nel caso in cui gli specificatori di unità siano indicati come parte dell'argomento, dovete leggere le istruzioni per quell'argomento particolare.

Un'altra assunzione generale è che tutti i files '.COM' possano essere eseguiti se sono su un disco (sull'unità a dischi corrente o su un'unità alternativa), così come tutti i files '.PRL'. Per esempio, per eseguire SAMPLE.COM, dovete digitare 'SAMPLE ↵'. Se SAMPLE.COM esiste sull'unità B e se siete nell'unità A, allora dovete digitare 'B:SAMPLE ↵' per eseguirlo.

Infine, tutto quello che dovete digitare nel sistema è sottolineato. Il simbolo ↵, insieme a una lettera come C (cioé, ↵ C), indica il tasto Control (CTRL) usato contemporaneamente al tasto della lettera.



o CP/M versione 1.4
o CP/M versione 2.2
● MP/M versione 1.0

Abortisce un programma funzionanate
(ABORT.COM o ABORT.PRL)

FORMATI

1. ABORT nomeprogramma
2. ABORT nomeprogramma e numeroconsole

ARGOMENTI

nomeprogramma — Il nome del programma funzionante da abortire.

numeroconsole — Numero della console dal quale il programma era stato iniziato. Deve essere specificato se la console è diversa da quella alla quale si specifica ABORT.

DESCRIZIONE

Questo comando termina l'esecuzione di un programma specifico. Deve essere usato con cura poiché ogni utente può abortire qualsiasi programma iniziato a qualsiasi console.

COME USARLO

Se il programma era stato iniziato dalla vostra console, semplicemente specificate ABORT seguito dal nome del programma. Se il programma era stato iniziato da un'altra console, usate il secondo formato e specificate il numero di console.

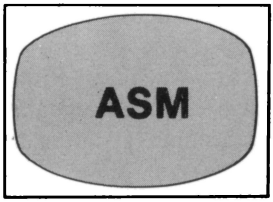
ESEMPI

0A > ABORT COMPUTER ↵

(COMPUTE era stato iniziato da questa console).

2A > ABORT TEST 1 ↵

(TEST era stato iniziato dalla console 1).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Assembla un file

(ASM.COM) fornito con CP/M (o MP/M)

FORMATI

1. ASM nomefile
2. ASM nomefile.*shp*

ARGOMENTI

nomefile — Il nome di un file sorgente '.ASM' (file di testo) che contiene le istruzioni in linguaggio assembler come testo ASCII. ASM cerca il 'nomefile.ASM'; l'estensione '.ASM' non deve essere specificata nel nome di file.

.shp — I parametri opzionali di ASM, che consistono di tre lettere precedute da un punto. La s deve essere la lettera dell'unità (A, B, ..., P,) che contiene il file sorgente '.ASM' se non è sull'unità corrente. La h deve essere o una lettera dell'unità (A, B,..., P,) che deve ricevere il file '.HEX' creato da ASM, o 'Z' per dire a ASM di non creare il file '.HEX' (descritto sotto). La p deve essere o la lettera dell'unità (A, B, ..., P) che deve ricevere il file '.PRN' creato da ASM o 'X' per mandare il file '.PRN' sullo schermo del terminale o 'Z' per dire ad ASM di non creare il file '.PRN' (descritto sotto).

DESCRIZIONE

Il programma assembler (ASM.COM) converte un file sorgente in linguaggio assembler (scritto in codice 8080 o Z-80) in un file in codice

macchina del tipo '.HEX' che può essere successivamente caricato (usando il comando LOAD) nel sistema come comando transitorio (programma eseguibile). ASM crea anche un file con estensione '.PRN' che contiene le righe del sorgente in linguaggio assembler con le segnalazioni di errore e la notazione esadecimale (codice macchina) generata da ASM.

COME USARLO

Usate il formato 1 se il file sorgente 'ASM' è sul disco corrente e se volete creare anche files '.HEX' e '.PRN' sull'unità a dischi corrente. Altrimenti, dovete usare il formato 2 e specificare esplicitamente s come unità per il file sorgente, h come unità per ricevere '.HEX' e p come unità per ricevere il file '.PRN' (cioé, non generi il file '.HEX'), allora specificate una 'Z' per h. Se volete che ASM crei soltanto il file '.HEX' quando traduce il file sorgente, allora specificate una 'Z' per p (per non generare il file '.PRN'). Se volete che ASM mandi il file '.PRN' soltanto sullo schermo del terminale (e non ne salvi una copia sul disco), allora specificate una 'X' per p.

In entrambi i formati, ASM traduce ("assembla") le righe del sorgente in linguaggio assembler nella notazione esadecimale Intel per rappresentare il codice macchina (codice binario). Se ASM trova errori nel file sorgente, stampa la riga errata e un codice di errore.

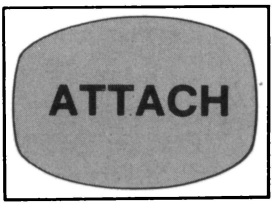
ESEMPI

A > ASM PROG ↵

(Esegue ASM sul file PROG.ASM nell'unità corrente).

A > ASM FAQUESTO.ABZ ↵

(Esegue ASM nel file FAQUESTO.ASM nell'unità A; mette il nuovo file FAQUESTO.HEX sull'unità B e non crea FAQUESTO.PRN).



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Attacca una console a un programma distaccato

(Processo residente o ATTACH.PRL)

FORMATO

ATTACH nomeprogramma

ARGOMENTO

nomeprogramma Il nome di un programma che era stato staccato dalla console che esegue ATTACH.

DESCRIZIONE

Il programma ATTACH attacca a una console un programma distaccato. Il programma distaccato, comunque, deve essere stato distaccato dalla stessa console (terminale). Potete distaccare un programma dalla console premendo ↑ D durante l'esecuzione del programma. Un processo in attesa sulla console si attacca automaticamente quando è distaccato un processo in esecuzione.

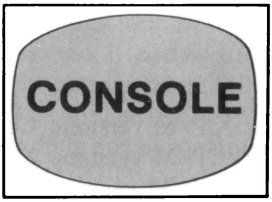
COME USARLO

ATTACH può essere eseguito come un comando con argomento il nome del programma, se avete ATTACH come processo di sistema residente generato con il sistema MP/M, o se avete ATTACH.PRL accessibile su un disco.

ESEMPIO

1A > ATTACH PROG.PRL ↵

(PROG1.PRL prende la console)



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Stampa il numero della console (terminale) che esegue il comando CONSOLE

(CONSOLE.COM o CONSOLE.PRL)

FORMATO

CONSOLE

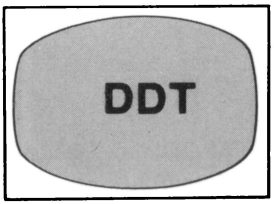
DESCRIZIONE

Quando il comando CONSOLE è digitato a un terminale, restituisce il numero di console del terminale stesso. Potrebbe essere utile per determinare quale console (terminale) ha distaccato un programma in attesa su di essa come mostrato dalla stampa MPMSTAT.

ESEMPIO

```
0A > CONSOLE ↵
Console = 1
0A >
```

(La console usata correntemente è la console 1, la seconda console dopo la console 0).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Esegue il programma di messa a punto per caricare, modificare e provare i programmi

(DDT.COM o DDT.PRL)

FORMATO

DDT *nomefile*

ARGOMENTO

nomefile Argomento opzionale che indica il nome di un file '.COM' o 'HEX'; dovete anche specificare l'estensione.

DESCRIZIONE

DDT sostituisce CCP in memoria a carica un file nella TPA; se non è specificato un file, DDT occupa la TPA e attende che venga caricato un file in memoria. DDT stampa anche l'indirizzo NEXT (indirizzo successivo all'ultimo del programma sotto prova) e il PC (contatore di programma). DDT ha i suoi comandi per inserire valori, stampare locazioni di memoria, salvare commenti, mettere punti di interruzione (break points) e altre funzioni di messa a punto.

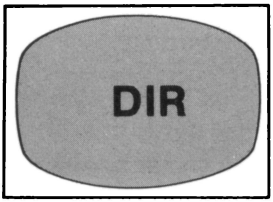
COME USARLO

Per poter eseguire DDT deve esistere come programa su un disco accessibile (cioé, come DDT.COM o DDT.PRL nei sistemi MP/M). Per terminare DDT in modo corretto usate il comando GO.

ESEMPIO

```
A > DDT PIP.COM ↵
```

NOTA: per informazioni aggiuntive, riferirsi alla CPM *Dynamic Debugging Tool (DDT) User's Guide* fornita dalla Digital Research.



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Direttrice - Stampa una lista di nomi dei files nella direttrice dell'unità a dischi corrente

(Comando interno nel CP/M, DIR.COM o DIR.PRL nell'MP/M)

FORMATO

DIR { *nomefile* | *maschera* }

ARGOMENTI

nomefile — Argomento opzionale per dire a DIR di cercare soltanto il file chiamato nomefile.

maschera — Sostituzione opzionale di nomefile per dire a DIR di cercare più files e fare una lista dei loro nomi. Sia nomefile che maschera possono avere gli specificatori di unità.

DESCRIZIONE

Se non si specifica il nomefile o una maschera, DIR assume che si vuole una lista di tutti i nomi dei files nella direttrice dell'unità corrente (soltanto i files con l'attributo $DIR, non i files con l'attributo $SYS). Si noti, comunque, che DIR stampa soltanto i files nell'area corrente, nel CP/M versione 2.2 e nell'MP/M.

Se si fornisce un nome di file, DIR stampa soltanto quel file, se è nell'unità a dischi corrente e nell'area utente corrente. Se si fornisce uno specificatore di unità (cioé, A:, B:, C: ..., P:), DIR cerca in quell'unità specifica, nell'area utente corrente.

Se si fornisce una maschera dei nomi di files, invece di un nome di file,

DIR certa tutti i files che soddisfano alla maschera nell'unità a dischi corrente (o nell'unità specificata) e nell'area utente corrente.

COME USARLO

Nel CP/M versione 1.4 e versione 2.2, DIR è un comando interno, (cioé, parte del sistema operativo) e può essere eseguito da qualsiàsi unità a dischi e da qualsiasi area utente. Nell'MP/M, DIR può essere fornito come DIR.COM o DIR.PRL. DIR.COM e DIR.PRL devono esistere nell'unità corrente a meno che specifichiate un'altra unità come prefisso a DIR. Devono anche essere nell'area utente corrente per poter essere eseguiti in un sistema MP/M.
NOTA: la stampa di DIR nella versione 1.4 è soltanto verticale, mentre la stampa della versione 2.2 (e dell'MP/M) ha righe orizzontali e colonne verticali.

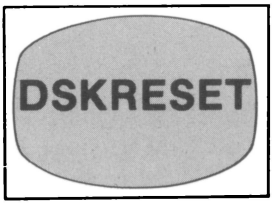
ESEMPI

```
A > DIR ↵
ASM.COM    DUMP.COM    ED.COM     PIP.COM
LOAD.COM   PROG.HEX    BASIC.COM
STAT.COM   SAMPLE.TXT  FILE.TXT
TONE.TXT   SYSGEN.COM  32CPM. COM
GAME1.INT  GAME1.BAS   GAME2.INT
GAME2.BAS  SOURCE.BAS  TEST.SYS
•
•
•
```

(Stampa del tipo CP/M versione 2.2, di tutti i files con l'attributo $ DIR nell'unità A, utente 0).

```
B > DIR*.TXT ↵
SAMPLE.TXT
PROG.TXT
POEM.TXT
NAME.TXT
BOOK.TXT
ONE.TXT
LETTER.TXT
FILE.TXT
```

(Stampa tipo CP/M versione 1.4, di tutti i files nell'unità B con l'estensione '.TXT').



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Permette la sostituzione del disco in un sistema multiutente
(DSKRESET.COM o DSKRESET.PRL)

FORMATO

DSKRESET

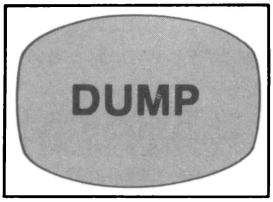
DESCRIZIONE

Quando viene eseguito, DSKRESET manda un messaggio agli altri terminali agganciati al sistema: "Confirm reset disk system (Y/N)?" (Confermare la sostituzione del disco di sistema). Se qualche terminale risponde con una "N" per 'no', allora la richiesta di sostituzione del disco è rifiutata. Se tutti i terminali rispondono con una "Y" per 'si', allora l'utente può cambiare il disco (dischetto). È importante che l'utente indichi agli altri utenti che intende cambiare il dico o dischetto, poiché gli altri utenti potrebbero avere in corso aggiornamenti o accessi ai files sul disco.

ESEMPI

```
0A > DSKRESET ↵
Confirm reset disk system (Y/N)?Y
```

(Questo messaggio appare su ogni terminale collegato al sistema).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Stampa il contenuto di un file su disco in formato esadecimale sul terminale

FORMATO

DUMP nomefile

ARGOMENTI

nomefile — Il nome (compresa l'estensione) di un qualsiasi file su disco.

DESCRIZIONE

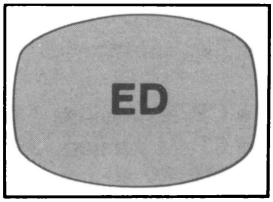
DUMP stampa in notazione esadecimale sullo schermo del terminale il contenuto di qualsiasi file su disco, in righe di 16 bytes con l'indirizzo assoluto del primo byte sulla sinistra.

COME USARLO

Eseguite il programma DUMP (DUMP.COM) come comando, fornendo un nome di file con l'estensione. Se DUMP.COM non è sull'unità corrente, specificate la lettera dell'unità prima del comando.

ESEMPI

A > DUMP SCRATCH.HEX ↵
A > DUMP B:NONAME.COM ↵



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Edita su un file

(ED.COM o ED.PRL)

FORMATO

ED nomefile

ARGOMENTO

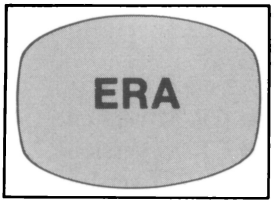
nomefile — Nome del file su cui si fa l'edit; deve essere un file di testo (o un altro tipo di file ASCII). Dovete anche fornire l'estensione.

DESCRIZIONE

Il programma ED crea un buffer di edit e permette all'utente di modificare un testo in questo buffer. Dapprima, ED cancella qualsiasi file '.BAK' che corrisponda al primo nome del file (ad esempio, SAMPLE.BAK per SAMPLE.TXT). Poi, permette all'utente di aggiungere testo al buffer per modificarlo. Il testo può essere scaricato su un file temporaneo mentre viene modificato l'altro testo nel buffer. Può essere inserito un testo dai files "libreria". Quando ED viene terminato da un comando E aggiorna il file sorgente e crea un file di sicurezza (backup) del file sorgente originale.

COME USARLO

Dovete avere ED.COM o ED.PRL su un disco (dischetto) accessibile. Il Capitolo 4 descrive come usare ED. C'é un riassunto addizionale dei comandi di ED nelle Appendici D e E.



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Cancella uno o più files dal disco o dischetto

(Comando interno nel CP/M, ERA.COM o ERA.PRL nell'MP/M)

FORMATO

ERA { nomefile | maschera }

ARGOMENTI

nomefile — Si deve dare come argomento ad ERA un nome di file per dire di cancellare un certo file. Il nome deve includere l'estensione e può comprendere anche uno specificatore di unità.

maschera — Si può fornire una maschera dei nomi di file al posto del nome (come descritto nel Capitolo 2) per dire ad ERA di cancellare più files in una volta. La maschera '*.*' cancella tutti i files nell'unità corrente nella versione 1,4, o tutti i files nell'unità corrente e nell'area utente corrente, ma non nelle altre aree utente, nella versione 2.2 dell'MP/M.

DESCRIZIONE

Il comando ERA cancella qualsiasi file il cui nome sia fornito come argomento, a meno che il files sia a sola lettura (abbia l'attributo $R/O), o il disco corrente (o il disco specificato) sia a sola lettura. Se ERA non trova il file, stampa il messaggio "No File" (Nessun file). ERA, nel CP/M versione 2.2 e nell'MP/M, cancella soltanto i files nell'area utente corrente. Potete cancellare files in un'unità alternativa specificando l'unità come parte del nome di file o della maschera (ad es., 'ERA

B:FILE1.*' cancella tutte le ricorrenze di FILE1 con qualsiasi estensione, che esistono nell'area utente corrente nell'unità B).

COME USARLO

ERA è un comando interno nel CP/M versione 1.4 e versione 2.2 che potete eseguire da qualsiasi unità. Nell'MP/M, ERA è fornito come ERA.COM o ERA.PRL (file di comando per la memoria assoluta o file rilocabile per la memoria rilocabile). ERA.COM o ERA.PRL devono esistere nell'unità corrente o essere richiamati da un'altra unità usando uno specificatore di unità (ad es., B:ERA).

Per cancellare un intero disco nel CP/M versione 1.4, dovete usare soltanto la maschera '*.*' che corrisponde a tutti i files sul disco. Per cancellare un intero disco nel CP/M versione 2.2 e nell'MP/M, dovete scrivere un programma che riempia il disco con dati inutili o usare la forma 'ERA *.*' in ciascuna area utente, assicurandovi che non ci siano files a sola lettura (con l'attributo $R/O) rimasti non cancellati.

ESEMPI

A > ERA SAMPLE.TXT ↵

(Questo comando cancella il file SAMPLE.TXT nell'unità A).

A > ERA B:JUMP.TXT ↵

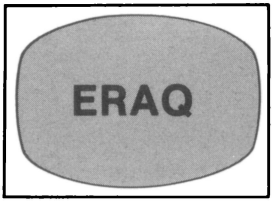
(Questo comando cancella il file JUPM.TXT nell'unità B).

0A > ERA *.* ↵

(Questo è un esempio MP/M, dove ERA cancella tutti i files nell'area utente 0 dell'unità A).

A > ERA*.HEX ↵

(Questo comando cancella tutti i files che hanno l'estensione '.HEX' nell'unità A).



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Cancella uno o più files dal disco o dischetto
(ERAQ.COM o ERAQ.PRL nell'MP/M)

FORMATO

ERAQ maschera

ARGOMENTO

maschera Si fornisce una maschera di nomi di files per dire ad ERAQ di cancellare più files uno dopo l'altro.

DESCRIZIONE

Il comando ERAQ cancella in successione tutti i files che corrispondono alla maschera fornita con l'argomento, a meno che il files sia a sola lettura (abbia l'attributo $R/O), o il disco specificato sia a sola lettura. Diversamente da ERA, ERAQ chiede all'utente di confermare prima di cancellare ciascun file.

COME USARLO

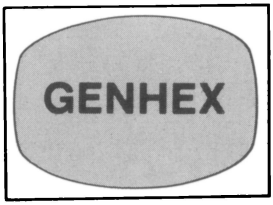
ERAQ è fornito sotto l'MP/M come ERAQ.COM o ERAQ.PRL (file di comando per la memoria assoluta o file rilocabile per la memoria rilocabile). ERAQ.COM o ERAQ.PRL devono esistere nell'unità corrente o essere chiamati da un'altra unità usando uno specificatore di unità (ad es. B:ERAQ).

Per cancellare l'area utente, usate la maschera '*.*' che corrisponde a tutti i files nell'area utente corrente. Per cancellare un intero disco, dovete scrivere un programma che riempia il disco con qualche dato, o

usare la forma 'ERAQ *.*' in ciascuna area utente, assicurandovi che non vi siano files a sola lettura (con l'attributo $R/O) rimasti non cancellati.

ESEMPI

```
0A > ERAQ PROG.* ↵
A:PROG TXT? Y
A:PROG INT? Y
```



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Trasforma un file COM in un file HEX
(GENHEX.COM o GENHEX.PRL)

FORMATO

GENHEX nomeprogramma.COM offset

ARGOMENTI

nomeprogramma — Il nome del programma (deve essere del tipo .COM). Può essere preceduto da un indicatore di disco. Offset per il file .HEX da generare (esadecimale).

DESCRIZIONE

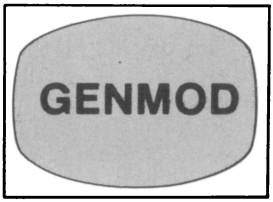
Questo comando genera un file del tipo HEX da un file del tipo COM e sposta il file risultante di una quantità specifica.

COME USARLO

Questo comando è usato generalmente per generare un file rilocabile a pagine (tipo PRL) usando il comando GENMOD. In questo caso lo spostamento è 0 o 0100 bytes (esadecimale).

ESEMPIO

1A > GENHEX ACTION.COM 100 ↵



- ○ CP/M versione 1.4
- ○ CP/M versione 2.2
- ● MP/M versione 1.0

Genera un programma modificato
Crea un programma rilocabile da due files 'HEX' concatenati

(GENMOD.COM o GENMOD.PRL)

FORMATO

GENMOD file.hex file.prl $*bbbb*

ARGOMENTI

file.hex — Questo file '.HEX' deve contenere due files '.HEX' concatenati, spostati uno rispetto all'altro di 100H bytes.

file.prl — Questo è il nome del file '.PRL' da creare. Potete sostituire un'estensione '.RSP' per creare un processo di sistema residente.

$*bbbb* — Questo è un numero esadecimale opzionale di bytes di memoria addizionale per il programma.

DESCRIZIONE

Il programma GENMOD produce un programma rilocabile con un'estensione '.PRL' (o '.RSP') da due files '.HEX' concatenati, spostati di 100H. Se si fornisce $*bbbb*, GENMOD assegna anche questa quantità di memoria addizionale per il programma.

COME USARLO

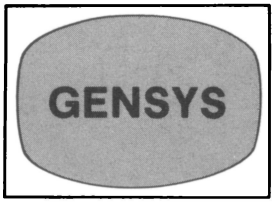
GENMOD deve esistere sul dischetto come GENMOD.COM o GEN-

MOD.PRL. GENMOD.COM può essere eseguito anche dal CP/M versione 2.2, ma il file '.PRL' può essere eseguito soltanto sotto l'MP/M.

ESEMPIO

1A > GENMOD FINAL.HEX PERFORM.PRL $*1000*↵

(Questo comando produce il programma rilocabile PERFORM.PRL dal file FINAL.HEX (che è una concatenazione di due files '.HEX' spostati di 100H), con una memoria addizionale di 1000H).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Genera un sistema MP/M dal CP/M
(GENSYS.COM)

FORMATO

GENSYS

DESCRIZIONE

Il programma GENSYS pone molte domande sul nuovo sistema, poi costruisce il file MPM.SYS per contenere il sistema. Il programma MPMLDR.COM carica il sistema (MPM.SYS) in memoria. GENSYS è usato per generare nuove versioni del sistema. GENSYS permette all'utente di incorporare processi residenti nel sistema. GENSYS cerca i files con estensione '.RSP', e chiede all'utente di selezionare i processi residenti da una lista.

COME USARLO

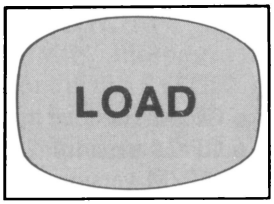
GENSYS.COM può essere eseguito da un sistema CP/M o MP/M. Dovete rispondere alle domande con un valore e un RETURN. Per una descrizione più completa di ciascuna domanda, si veda il Capitolo 5, "Installazione e modificazione dell'MP/M".

ESEMPIO

```
A1 > GENSYS ↵

MP/M 1.0 System Generation

Top page of memory  = C0 ↵
Number of consoles  = 2 ↵
Breakpoint RST #    = 5 ↵
Allocate user stacks for system calls (Y/N) Y ↵
Memory segment bases, (ff terminates list)
  : 00 ↵
  : 40 ↵
  : 60 ↵
  : ff ↵
Select Resident System Process: (Y/N)
TIME      ?Y ↵
SCHED     ?N ↵
ATTACH    ?Y ↵
SPOOL     ?Y ↵
```



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Carica un file nella memoria eseguibile
Converte un file '.HEX' in un file di un comando eseguibile '.COM'.

(LOAD.COM)

FORMATO

LOAD nomefile

ARGOMENTO

nomefile — Il nome di un file con estensione ".HEX"; l'estensione non è necessaria.

DESCRIZIONE

Il programma LOAD prende un programma che sia in "formato esadecimale" Intel corretto e lo converte in un file di comando che può essere eseguito (file con l'estensione '.COM'). Il file di comando prende il nome di nomefile.COM (il file esadecimale è nomefile.HEX).

COME USARLO

Per eseguire LOAD.COM, dovete averlo su un disco accessibile. Potete eseguire LOAD.COM da un altro disco specificando la lettera dell'unità prima del comando. Potete usare il file ".COM" creato da LOAD come comando transitorio, da caricare ed eseguire nella TPA, semplicemente digitando il nome del file nomefile.COM.

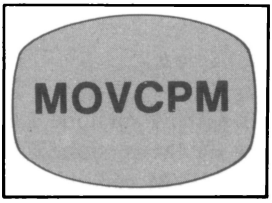
ESEMPIO

A > LOAD SAMPLE ↵

(Sample.HEX è convertito in SAMPLE.COM)

A > SAMPLE ↵

(Ora potete eseguire SAMPLE.COM).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Riconfigura una versione del CP/M per adattarsi ad altri requisiti di memoria

(MOVCPM.COM)

FORMATO

MOVCPM { * / bb } *

ARGOMENTI

bb Estensione opzionale di memoria, in cifre decimali che rappresentano il numero di Kilobytes (esempio: "32" per un sistema a 32K). Se sostituito con un asterisco (*), MOVCPM calcola l'ammontare totale di RAM (memoria ad accesso casuale), del calcolatore ospite e costruisce il CP/M per questa estensione.

* Un secondo asterisco (*) dopo bb o il primo asterisco (*) dice a MOVCPM di lasciare il nuovo sistema in memoria in preparazione di una operazione di SYSGEN o SAVE. Anche questo argomento è opzionale. Se non è fornito, MOVCPM esegue il nuovo sistema senza registrarlo su un dischetto (disco).

DESCRIZIONE

Il programma MOVCPM crea un'immagine di memoria del sistema e lo riconfigura per adattarlo ad un'altra estensione (bb) o all'estensione massima del sistema ospite. Se viene fornito il secondo asterisco (*) come

in 'MOVCPM * *', o bb è seguito da un asterisco come in 'MOVCPM 32 *', MOVCPM lascia il sistema appena costruito nella TPA in preparazione di una SYSGEN o SAVE per registrare la versione sul dischetto. Se non fornite l'asterisco, il nuovo sistema viene eseguito ma non registrato in modo permanente.

COME USARLO

Potete eseguire MOVCPM se esiste su un dischetto accessibile. Solitamente è usato per preparare un nuovo sistema modificato secondo un altro ambiente hardware.

ESEMPI

A > MOVCPM 48 ↵

(Questo comando costruisce una versione 48K del CP/M e la esegue senza memorizzarla sul dischetto).

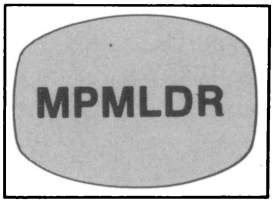
A > MOVCPM 32 * ↵

(Questo comando crea un CP/M a 32K e lo lascia in memoria in preparazione di un'operazione di SYSGEN o SAVE).

READY FOR "SYSGEN"OR
"SAVE 32 CPM32.COM"
A > MOVCPM * * ↵

(Questo comando costruisce una versione del CP/M a memoria massima e la lascia in memoria, pronta per un SYSGEN o SAVE).

NOTA: per informazioni addizionali, vedi Capitolo 5, "Installazione e modificazione del CP/M".



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Caricatore dell'MP/M
Carica, rialloca ed esegue il sistema MP/M

(MPMLDR.COM)

FORMATO

MPMLDR

DESCRIZIONE

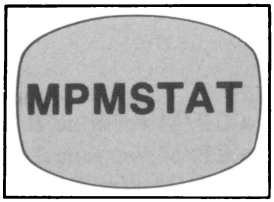
Il programma MPMLDR carica il file MPM.SYS che contiene il sistema generato, lo rialloca in memoria, poi lo esegue per caricare l'MP/M. MPMLDR fornisce anche una stampa dei parametri di sistema - il numero di console, il breackpoint, l'indirizzo massimo di memoria e una tabella di segmenti di memoria.

COME USARLO

Potete eseguire MPMLDR.COM dall'MP/M o dal CP/M. Potete anche portarlo nel sistema dalle prime due tracce di un dischetto di sistema usando un programma di caricamento su partenza a freddo. Per maggiori informazioni, vedi Capitolo 5, "Installazione e modificazione dell'MP/M".

ESEMPIO

```
A > MPMLDR ↵
MP/M 1.0 Loader
Number of consoles    = 2
Breakpoint RST #      = 5
Top of memory         = C0FFH
Memory Segment Table:
SYSTEM DAT C000H 0100
●
●
●
```



- ○ CP/M versione 1.4
- ○ CP/M versione 2.2
- ● MP/M versione 1.0

Stampa lo stato del sistema MP/M

(Processo residente di MPMSTAT.PRL)

FORMATO

MPMSTAT

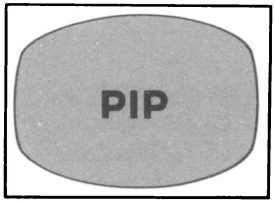
DESCRIZIONE

MPMSTAT stampa i nomi dei processi in attesa di tempo di CPU, dei processi in attesa dei messaggi dalle code e dei processi in attesa di mandare messaggi. Stampa anche i processi ritardati e quelli che interrogano una console, i processi in attesa di segnali (flags), i segnali attivi, le code attive, i processi in attesa delle consoles, i processi attaccati alle consoles e la allocazione di memoria dell'intero sistema.

ESEMPIO

1A > MPMSTAT ↵

NOTA: La stampa indica lo stato dei vari processi. Vedere a pagina 104 per un esempio dettagliato.



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Programma di scambio tra periferiche
Realizza una o più operazioni di copiatura

(PIP.COM)

FORMATO

1. PIP $\left\{\begin{matrix} u\text{:nuovacopia} \\ \text{u:} \end{matrix}\right\}$ = *u*:vecchiacopia[p]

2. PIP
   * *u*:nuovacopia = *u*:vecchiacopia[p]
   * ...
   * ...
   * ↵

3. PIP$\left\{\begin{matrix} \text{dev:} \\ u\text{: nomefile} \end{matrix}\right\}$ = $\left\{\begin{matrix} \text{dev:} \\ u\text{: nomefile} \end{matrix}\right\}$ [p], $\left\{\begin{matrix} \text{dev:} \\ u\text{: nomefile} \end{matrix}\right\}$, ... [p]

4. PIP u: = u:maschera [p]

ARGOMENTI

$\left\{\begin{matrix} u\text{:nuovacopia} \\ \text{u:} \end{matrix}\right\}$ L'utente deve scegliere, nei formati 1 e 2, se vuole che la nuova copia abbia il nuovo nome nuovacopia o se la nuova copia deve avere lo stesso nome ma essere su un'unità differente, l'unità u:

u:vecchiacopia — In entrambe le forme 1 e 2 è richiesto il nome del file da copiare, ma non lo specificatore di unità.

[p] — In tutti i formati, l'utente può usare un parametro di PIP, facendolo seguire al file a cui si riferisce (opzionale).

disp:
u:nomefile — Nel formato 3, l'utente deve scegliere tra un disp: (nome di dispositivo, ad esempio CON) o un nome di file con lo specificatore di unità u:. Questa forma può essere usata per mandare un file a un dispositivo, ricevere dati da un dispositivo a un file, o mandare codici speciali a un dispositivo.

u: = *u*:maschera — Sono richiesti sia u: come destinazione che la maschera come sorgente; la u: sulla maschera è opzionale. Il formato 4 è usato per copiare più files su un altro dischetto usando gli stessi nomi per i files.

DESCRIZIONE

Formato 1: Se si fornisce soltanto u: e non nuovacopia, la nuovacopia avrà lo stesso nome della vecchia; comunque, la u: a sinistra deve essere diversa dal prefisso opzionale u: a destra. Se si omette u:, si assume che la vecchia copia sia sull'unità a dischetti correnti. Se l'utente fornisce nuovacopia, la nuova copia avrà un nuovo nome e l'utente può copiare vecchiacopia su nuovacopia senza specificazione di unità (senza u: o *u:).*

Formato 2: Le espressioni PIP seguono le stesse regole; se l'utente vuole realizzare più operazioni PIP, può eseguire PIP e lasciarlo in memoria mentre l'utente fornisce espressioni PIP al comparire dell'asterisco di prompt (*). PIP può essere terminato digitando soltanto RETURN. Con alcuni parametri, si verificano azioni differenti quando PIP è eseguito come comando (formato 1) invece che come programma (formato 2).

Formato 3: Nei comandi e espressioni PIP, si può usare anche disp: (nome di dispositivo) al posto di u:nomefile (nome di file). L'utente non può copiare dati da un dispositivo a sola ricezione e mandarne a un dispositivo a sola trasmissione. La parte sinistra dell'espressione è sempre la destinazione (cioé, un dispositivo ricevente o un file), la parte destra è sempre sorgente (cioé, il dispositivo trasmittente o un file da copiare). L'utente può concatenare (congiungere) più files sorgente in un file destinazione o su un dispositivo. L'utente può anche usare nomi di dispositivi speciali indicati nell'Appendice F.

Formato 4: L'utente può copiare più files su un altro dischetto usando una maschera (maschera dei nomi di file) con lospecificatoredell'unità *u*: opzionale. Si tenga presente che lo specificatore di unità nella parte sinistra è obbligatorio (u:) e che le nuove copie hanno lo stesso nome delle vecchie ma sono su un altro dischetto. L'utente non può avere due files con lo stesso nome sullo stesso dischetto (nella stessa area utente).

I nomi di dispositivi permessi nelle espressioni PIP sono elencati nell'Appendice F. Le parole chiave usate per realizzare funzioni speciali sono elencate nell'Appendice G.

COME USARLO

Potete eseguire PIP.COM da qualsiasi unità alternativa specificando la lettera di unità come prefisso al comando (ad esempio "B:PIP ↵"). Potete anche eseguire PIP, lasciandolo in memoria mentre togliete il dischetto di sistema per inserirne uno da copiare e tornare al sistema dopo aver inserito il dischetto di sistema e aver terminato PIP con un semplice RETURN. Il Capitolo 3 contiene la descrizione completa di quasi tutte le applicazioni PIP.

ESEMPI

```
A > PIP B: = *.* ↵
```

(Questo comando copia tutti i files dall'unità corrente all'unità B).

```
A > PIP ↵
*FILE2=TEST2 ↵
```

(Questa espressione copia TEST2 in un file chiamato FILE2).

```
*LST:=FILE2 ↵
```

(Questa espressione manda una copia di FILE2 al dispositivo LST:).

```
*PUN:=NUL:,PROG.ASM,EOF: ↵
```

(Questa espressione manda 40 Nulls al dispositivo PUN:, insieme al file PROG.ASM e al carattere di fine file).

```
*B:=PROG.ASM ↵
```

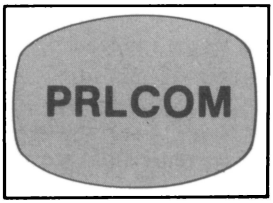
(Crea una copia PROG.ASM sull'unità B con lo stesso nome).

```
* ↵
```

```
A > PIP FILE2=FILE1 [G2] ↵
```

(Questo esempio funziona soltanto nel CP/M versione 2.2, e copia il FILE1 dall'area utente 2 nel FILE1 nell'area utente corrente sullo stesso disco).

NOTA: per informazioni addizionali vedere il Capitolo 3, e le Appendici F (nomi dei dispositivi di PIP), G (parole chiave di PIP) e H (parametri di PIP).



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Trasforma un file PRL in un file COM

(PRLCOM.COM o PRLCOM.PRL)

FORMATO

PRLCOM nomeprogramma1.
PRL nomeprogramma2. COM

ARGOMENTI

nomeprogramma1 Il nome del programma sorgente (può essere preceduto da un indicatore di unità a disco).

nomeprogramma2 Il nome del programma destinazione (può essere preceduto da un indicatore di unità a disco).

DESCRIZIONE

Questo comando trasforma un programma del tipo PRL in un programma del tipo COM. Se il nome del programma COM risultante è già utilizzato, l'utente viene avvertito e gli si dà l'opzione di cancellare il comando.

COME USARLO

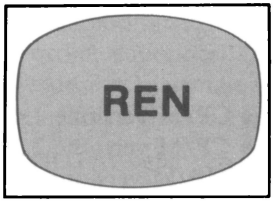
Quando l'utente vuole che un programma risieda in una TPA assoluta invece che in un segmento di memoria rilocabile, può ottenere questo risultato con PRLCOM, trasformando un file PRL in un file COM.

ESEMPI

1A > PRLCOM DOIT.PRL DOIT.COM ↵

oppure:

2A > PRLCOM DOIT.PRL B:NUNAME.COM ↵



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Cambia nome a un file

(Comando interno nel CP/M, REN.COM o REN.PRL nell'MP/M)

FORMATO

REN nuovonome = vecchionome

ARGOMENTI

nuovonome e vecchionome	Argomenti richiesti come nomi di files, comprendenti le estensioni; non sono permessi i prefissi con la lettera dell'unità.

DESCRIZIONE

Il comando (o programma) REN cambia vecchionome in nuovo nome; l'utente deve fornire i nomi di files al completo con le estensioni.

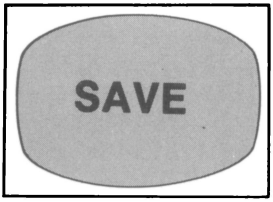
COME USARLO

Nel CP/M, REN è un comando interno che può essere eseguito in qualsiasi momento. Nell'MP/M, dovete avere REN.COM o REN.PRL su un disco accessibile.

ESEMPIO

A > REN NEWTRIC.HEX = OLDDOG.HEX ↵

(Questo comando cambia il nome del file OLDDOG.HEX in NEWTRIC.HEX).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Salva il contenuto della memoria in un file su disco

(Comando interno)

FORMATO

SAVE p nomefile

ARGOMENTI

p — Il numero di "pagine" (segmenti di 256 bytes) da salvare, in decimale.

nomefile — Il nome del nuovo file su disco, con l'estensione.

DESCRIZIONE

Il comando SAVE salva il contenuto della TPA (memoria scratchpad) partendo dall'allocazione 100H per p pagine (segmenti di 256 bytes) in nomefile, che può essere successivamente corretto o eseguito (se il contenuto della TPA era un programma eseguibile prima del salvataggio).

Nell'MP/M, questa operazione è realizzata nei nuovi programmi di messa a punto (debugger) DDT o SID.

COME USARLO

Per calcolare p, dovete prima usare DDT per caricare il programma originale in memoria e usare il valore NEXT. L'indirizzo NEXT sarà l'ultimo indirizzo del programma più uno; comunque, per determinare il numero di pagine, usate questo semplice algoritmo.

Se le ultime due cifre (esadecimale) di NEXT sono 00, sottraete 1H (esempio, 1C00H - 1H = 1BFFH). Se le ultime due cifre non sono 00,

non modificate il numero. Ora, prendete le prime due cifre, cioé "i bits di ordine superiore" (esempio, '1B' dal valore 1BFFH), e convertite quel valore esadecimale in decimale per ottenere il numero di pagine (p).

Poiché SAVE è interno, potete eseguirlo in qualsiasi momento.

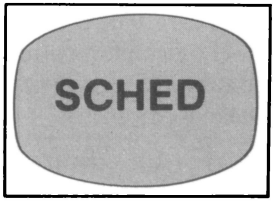
NOTA: nella versione 2.4, l'utente non può realizzare due operazioni consecutive di SAVE sullo stesso contenuto della TPA, poiché il primo SAVE causa un'operazione di direttrice che cambia molte aree della TPA. Nella versione 2.2, invece, questo problema è stato eliminato l'utente può realizzare due operazioni consecutive di SAVE sulla stessa TPA.

ESEMPIO

```
A > DDT SAMPLE.COM ↵
NEXT PC
1D00  00
- GO ↵
```

(Il valore sotto NEXT è 1D00H. Sottraete 1H per ottenere 1CFFH. Prendete il numero 1CH e convertitelo in decimale ottenendo 28).

```
A > SAVE 28 COPY.COM ↵
```



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Predispone un programma per l'esecuzione in una data e un tempo successivi

(Processo residente o SCHED.PRL)

FORMATO

SCHED mm/gg/aa hh:mm programma

ARGOMENTI

mm/gg/aa — Argomenti obbligatori per la data, dove mm è il mese (da 1 a 12), gg è il giorno (da 01 a 31) e aa sono le ultime due cifre dell'anno.

programma — Nome del file con l'estensione '.COM' o '.PRL'; le estensioni non devono essere fornite.

DESCRIZIONE

Il programma SCHED quando è eseguito, si mette in memoria in attesa dell'ora e della data corrispondenti a quelli dati come argomenti. Quando incontra quella specifica ora e quella data, SCHED esegue automaticamente il programma specificato.

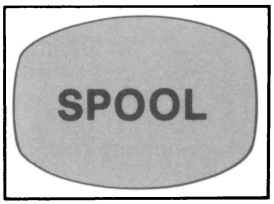
COME USARLO

Il programma SCHED può esistere come file '.PRL' o come processo di sistema residente e dovete fornire gli argomenti richiesti. Si tenga presente che chiunque può modificare l'ora o la data con il comando TOD, per cui è necessaria una certa cooperazione per utilizzare SCHED.

ESEMPI

0A > SCHED 12/31/80 23:59 EIGHTY ↵

(Questo comando predispone l'esecuzione del programma EIGHTY-COM (o EIGHTY.PRL) per il 31 Dicembre 1980 alle 23,59).



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Manda uno o più files su una coda di uscita, solitamente per una stampante

(Processo residente o SPOOL.PRL)

FORMATO

SPOOL nomefile, *nomefile*...

ARGOMENTI

nomefile e *nomefile*	Il primo nomefile è richiesto e i seguenti sono opzionali per altri files da mandare alla coda di uscita. L'utente deve anche specificare le estensioni dei nomi di files.

DESCRIZIONE

Il comando SPOOL manda i files uno per uno alla coda di uscita, dove attendono in ordine di essere ricevuti dal dispositivo LST: (solitamente la stampante, sebbene si possano assegnare altri dispositivi al dispositivo LST: usando STAT). I files deveno essere files di testo ASCII (files sorgenti, testi, ecc.).

COME USARLO

Il programma SPOOL può essere un file '.PRL' nell'unità a dischi corrente o un processo residente che potrete eseguire come un comando fornendo almeno un nome di file. Usate il comando STOPSPLR per interrompere l'operazione di uscita della coda.

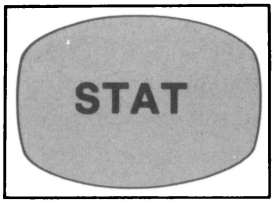
NOTA: Se si usa SPOOL.PRL invece di SPOOL.RSP, l'utente può abortirlo da un altro terminale. Per esempio, se SPOOL.PRL è stato attivato dal terminale 3, può essere interrotto digitando:

```
2B > STOPSPRL 3 ↵
```

ESEMPIO

```
0A > SPOOL PROG.PRN,SAMPLE.TXT,
NOVEL.JNC ↵
```

(Questo comando manda PROG.PRL, SAMPLE.TXT e NOVEL.JNC al dispositivo LST: che è solitamente una stampante o una teletype. I files attendono nella coda di uscita fino a quando possono essere ricevuti dal dispositivo).



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Stampa le informazioni di stato e assegna i dispositivi
(STAT.COM o STAT.PRL)

FORMATI

1. STAT { DEV: / VAL: }

2. STAT gen: = dis:,*gen*: = *dis*:,...

3. STAT *u*: = *R/O*

4. STAT *u*: { nomefile / maschera }

5. STAT *u*: { nomefile / maschera } { $S / $R/O / $R/W / $SYS / $DIR }

6. STAT *u*: { DSK: / USR: }

ARGOMENTI

DEV:
VAL:

Nel formato 1, DEV: produce la stampa degli assegnamenti effettivi dei dispositivi, mentre VAL: produce una stampa degli assegnamenti potenziali dei dispositivi (nel CP/M versione 2.2 VAL: elenca anche possibili comandi di STAT).

gen: = dis;

Nel formato 2, gen: indica un dispositivo generico (CON:, PUN:, RDR:, LST), mentre dis: indica un qualsiasi dispositivo fisico che è adatto per essere assegnato a un dispositivo generico.

u: = *R/O*

Nel formato 3, u: = R/O (dove u: è una lettera di unità) assegna all'unità u: uno stato di sola lettura; mentre se si indica soltanto u:, STAT stampa lo stato dell'unità; se non si indicano argomenti, STAT stampa lo stato dell'unità corrente (cioé sola-lettura o lettura-scrittura).

u: { nomefile / maschera }

Nel formato 4 l'utente può fornire u:nomefile (u: opzionale) per stampare lo stato (dimensione in records bytes, numero di estensioni, ecc.) di un file specifico, o fornire u:maschera per stampare lo stato di più files in una volta.

$S
$R/O
$SYS
$R/W
$DIR

Usando gli argomenti del formato 4, il formato 5 permette nel CP/M versione 2.2 e nell'MP/M di usare il parametro $S per stampare più informazioni sulle dimensioni di un file o di un gruppo di files e di usare i parametri $R/O, $R/W, $SYS e $DIR per assegnare gli attributi dei files. L'attributo $R/O (sola-lettura) impedisce di riscrivere o distruggere il file; viene cancellato dall'attributo $R/W (lettura-scrittura). L'attributo $SYS (sistema) "nasconde" il file al comando DIR; è cancellato dall'attributo $DIR (direttrice).

u:DSK:
USR:

Il formato 6, utilizzabile soltanto nel CP/M versione 2.2 e nell'MP/M, stampa le caratteristiche del disco quando si fornisce l'argomento DSK: (il disco corrente o opzionalmente quello dell'unità alternativa u:). Per stampare l'area utente corrente e quelle attive, usate USR:.

DESCRIZIONE

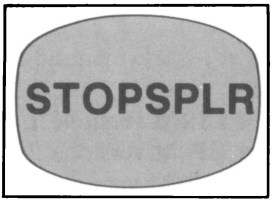
STAT fornisce informazioni statistiche sui files e sui dischi (dischetti) e assegna i dispositivi fisici ai nomi dei dispositivi generici (da usare con PIP). STAT si usa anche per rendere un disco a sola-lettura (formato 3) e un file a sola-lettura (formato 5) nel CP/M versione 2.2 e nell'MP/M e stampa l'area utente corrente e quelle attive.

COME USARLO

Per stampare informazioni statistiche, eseguite semplicemente STAT-.COM (o STAT.PRL) in uno dei suoi vari formati. Per assegnare i dispositivi, dovete usare il formato 2. I nomi dei dispositivi generici sono CON: (dispositivo di console), RDR: (lettore di nastro), PUN: (perforatore di nastro) e LST: (dispositivo di stampa), mentre i dispositivi fisici sono elencati con PIP nel Capitolo 3.

ESEMPI

```
Usando il CP/M versione 2.2:
A > STAT PIP.COM$S ↵
Size      Recs      Bytes     Ext     Acc
  55        55        12K       1     R/O  A:PIP.COM
A > STAT SAMPLE.COM $R/O ↵
A > STAT B: ↵
BYTES REMAINING ON B: 192K
B:      R/O
A >
```



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Interrompe un'operazione di uscita di una coda e svuota la coda di uscita

(Processo residente o STOPSPLR.PRL)

FORMATO

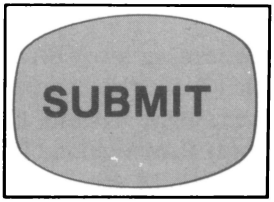
STOPSPLR

DESCRIZIONE

Il comando STOPSPLR interrompe un'operazione di SPOOL e svuota la coda di uscita. (Vedere il comando SPOOL).

ESEMPIO

0A > STOPSPLR ↵



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Esegue un file di comandi
(SUBMIT.COM o SUBMIT.PRL)

FORMATO

SUBMIT nomefile *v1 v2 v3...*

ARGOMENTI

nomefile — Nome obbligatorio di un file di testo con righe di comando, che deve avere l'estensione '.SUB'; '.SUB' non viene fornita nel nome del file.

v1 v2 v3... — Valori opzionali da sostituire alle variabili nel file di comando. Le variabili prendono la forma $1, $2, $3 ecc. e v1 sostituisce $1, v2 sostituisce $2 ecc.

DESCRIZIONE

Il programma SUBMIT accetta il file nomefile.SUB e costruisce il file $$$.SUB che viene eseguito dopo una partenza a caldo (dopo che è terminato il programma SUBMIT). Le righe di comando del file $$$.SUB sono eseguite fino a che il file è vuoto. Per costruire il file $$$.SUB, SUBMIT sostituisce nel file '.SUB' v1 a $1, v2 a $2, ecc. I files di comando sono eseguiti soltanto quando appaiono nell'unità A.

COME USARLO

Con l'uso di ED, create un file '.SUB' che contenga le righe di comando con gli argomenti espressi come variabili ($1, $2, ecc.). Eseguite il file di comando eseguendo SUBMIT.COM sul file '.SUB'.

ESEMPIO

Supponete che il file SMALL.SUB contenga queste righe di testo:

```
DIR $1.*
PIP $2: = $1.BAK
ERA $1.BAK
```

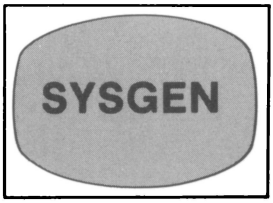
Se l'utente esegue questo file usando il comando SUBMIT:

```
A > SUBMIT SMALL PROG B ↵
```

ottiene in $$$.SUB le seguenti righe di comando:

```
DIR PROG.*
PIP B: = PROG.BAK
ERA PROG.BAK
```

Quando SUBMIT termina la sostituzione costruendo $$$.SUB, il sistema esegue il contenuto di $$$.SUB.



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Genera una copia del CP/M
Porta il sistema in memoria e/o fa una copia del dischetto di sistema

(SYSGEN.COM)

FORMATO

SYSGEN

DESCRIZIONE

Il programma SYSGEN inizializza il dischetto di sistema (scrive le prime due tracce del dischetto con il sistema). SYSGEN porta anche il sistema in memoria e lo esegue.

COME USARLO

Eseguite il programma SYSGEN.COM:

```
A > SYSGEN ↵
SYSGEN VERSION xx.xx
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
```

(Rispondete con la lettera dell'unità dove si trova il sistema, a meno che vogliate saltare l'operazione di lettura del sistema se il sistema è già in memoria per un'operazione di MOVCPM).

```
SOURCE ON A, THEN TYPE RETURN ↵
FUNCTION COMPLETE
```

(L'operazione di lettura del sistema è completa. Il sistema ora è nella memoria principale).

```
DESTINATION DRIVE NAME
(OR RETURN TO REBOOT) B
```

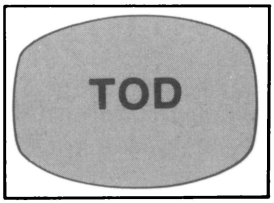
(Rispondete con la lettera dell'unità che contiene il nuovo dischetto di sistema da inizializzare, oppure premete RETURN per eseguire il sistema in memoria).

```
DESTINATION ON B, THEN TYPE RETURN ↵
FUNCTION COMPLETE
```

(Il sistema è scritto sul nuovo dischetto di sistema, che ora è utilizzabile).

```
DESTINATION DRIVE NAME
(OR RETURN TO REBOOT) ↵
A >
```

(Premete RETURN per terminare SYSGEN).



○ CP/M versione 1.4
○ CP/M versione 2.2
● MP/M versione 1.0

Ora del giorno
Stampa o assegna l'ora e la data
(Processo residente o TOD.PRL)

FORMATO

TOD *mm/gg/aa hh:mm:ss*

ARGOMENTI

mm/gg/aa Questo argomento è richiesto soltanto quando si assegna la data, dove mm è il mese, gg il giorno e aa sono le ultime due cifre dell'anno.

hh:mm:ss Questo argomento è richiesto quando si assegna l'ora o la data;hh è l'ora (da 0 a 24), mm sno i minuti (da 00 a 59), ss sono i secondi (da 00 a 59).

DESCRIZIONE

In un sistema MP/M, l'utente può stampare l'ora e la data correnti (compreso il giorno) eseguendo TOD senza argomenti. Se si fornisce la data e l'ora come argomenti, TOD stampa il messaggio "Strike a Key to set time", (premete un tasto per fissare l'ora) e l'utente può premere un tasto qualsiasi quando è pronto.

COME USARLO

Il programma TOD può esistere sull'unità a dischi correnti come file con estensione '.PRL' (rilocabile a pagine) o può essere un processo di sistema residente (comando) caricato con il resto del sistema quando è

stato generato. Dovete cooperare con gli altri utenti in un sistema a più utenti, poiché il calcolatore conosce il tempo soltanto come gli viene assegnato e altri utenti possono aver predisposto i programmi per farli funzionare con SCHED.

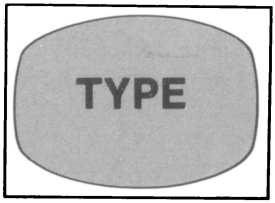
ESEMPI

```
0A > TOD ↵

Sat 12/29/79 02:37:21

0A > TOD 12/29/79 02:38:00 ↵

Strike a Key to set time
Sat 12/29/79 02:38:00
```



- CP/M versione 1.4
- CP/M versione 2.2
- MP/M versione 1.0

Stampa il contenuto di un file sullo schermo video della console (terminale)

(Comando interno del CP/M, TYPE.COM o TYPE.PRL nell'MP/M)

FORMATO

TYPE { nomefile / maschera }

ARGOMENTI

nomefile — L'utente deve fornire o uno specifico nome di file con
maschera — l'estensione o una maschera dei nomi per stampare più files.

DESCRIZIONE

Il comando TYPE stampa il contenuto di un file qualsiasi, ma l'utente sarà in grado di leggere il contenuto di un file di testo ASCII (file sorgente, file '.PRN', ecc.).

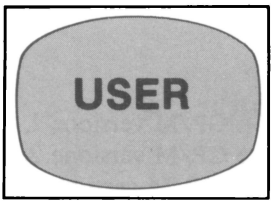
COME USARLO

Nel CP/M TYPE è un comando interno che potete eseguire in qualsiasi momento. Nell'MP/M è fornito o come TYPE.COM (comando transitorio) o come TYPE.PRL (programma rilocabile).

ESEMPI

A > TYPE SMALL.SUB ↵

A > TYPE*.TXT ↵



o CP/M versione 1.4
o CP/M versione 2.2
● MP/M versione 1.0

Cambia l'area utente corrente
o visualizza l'area utente in un sistema MP/M

(USER.COM o USER.PRL)

FORMATO

1. USER *n* *(soltanto MP/M)*

2. USER n

ARGOMENTO

n — Argomento richiesto nel CP/M versione 2.2 e opzionale nell'MP/M, che indica il numero dell'area utente (da 0 a 15).

DESCRIZIONE

Il formato 1 stampa il numero dell'utente corrente se non si fornisce n; se si fornisce n, USER cambia l'area utente nell'area n. Il formato 2 permette soltanto il cambiamento e non la visualizzazione dell'area utente. Il formato 2 cambia l'area utente nell'area n.

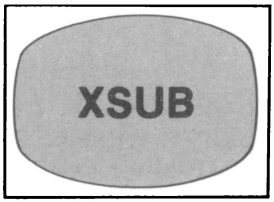
COME USARLO

Il programma USER è fornito o come USER.COM o come USER.PRL.

ESEMPIO

```
A > USER 3 ↵

A >
```



o CP/M versione 1.4
• CP/M versione 2.2
• MP/M versione 1.0

Possibilità estesa di SUBMIT per fornire dati di ingresso ai programmi eseguiti nei files di comando

(XSUB.COM)

FORMATO

XSUB

DESCRIZIONE

Quando XSUB viene posto all'inizio di un file '.SUB' (o se XSUB è eseguito come comando) si colloca nell'area immediatamente sotto il CCP per gestire le righe di comando del file. '.SUB' e quindi fornisce un ingresso da console in un buffer ai programmi eseguiti con l'operazione di SUBMIT. I programmi che leggono dati dalla console li ricevono direttamente dal file '.SUB'.

COME USARLO

Inserite 'XSUB' come prima riga di comando del file '.SUB' e eseguite il file con il programma SUBMIT. XSUB rimane attivo fino alla successiva partenza a freddo e il file '.SUB' è attivo fino a quando si svuota.

ESEMPIO

```
file NUOVO.SUB  XSUB
                DDT
                I$1.HEX
                R
```

GO
SAVE 1 $2.COM

A > SUBMIT NUOVO QUESTO QUELLO ↵

('QUESTO' sostituisce $1 e 'QUELLO' sostituisce $2; il programma XSUB richiama DDT con le righe di comando 'IQUESTO.HEX', 'R' e 'GO'; poi XSUB richiama il CCP con 'SAVE 1 QUELLO.COM'.XSUB rimane attivo fino a una partenza a freddo).

CAPITOLO 7

CONSIGLI PRATICI

INTRODUZIONE

Ora che avete acquisito una buona conoscenza di lavoro del CP/M e delle sue risorse, ecco alcuni consigli pratici per aumentare la vostra efficienza nell'uso del sistema CP/M. Questo capitolo suggerisce il modo di prevenire alcuni dei problemi che possono presentarsi nell'uso di un sistema CP/M. Vengono proposte varie raccomandazioni e soluzioni.

La prevenzione dei problemi è importante quando si usa un sistema di elaborazione. Errori molto semplici e banali fatti dall'utente possono causare errori molto seri e gravi nel programma. Pertanto è essenziale una disciplina da parte dell'utente.

DISCIPLINA DELL'UTENTE

Fate sempre una copia di ogni nuovo programma o dischetto che usate per la prima volta, nel caso che l'originale possa essere danneggiato accidentalmente. Quando inserite un nuovo dischetto sul quale intendete scrivere, eseguite un CTRL-C per catalogarlo nel sistema. Di volta in volta eseguite un comando STAT o un DIR per verificare la direttrice e lo spazio che rimane sul dischetto.

Al termine di una sessione in cui è stato manipolato un file lungo, è raccomandabile che l'utente crei una copia di questo file su un altro dischetto. I settori saranno copiati in sequenza e la nuova copia sarà caricata molto più velocemente del file originale da un programma applicativo. Il file originale dovrebbe essere conservato come versione di sicurezza.

Si dovrebbe organizzare la stanza del calcolatore per incoraggiare la disciplina dell'utente. (Nella sezione delle appendici di questo libro è mostrata una lista di materiali tipici). Dovrebbero essere disponibili tutte le documentazioni utente, i dischetti vuoti e i programmi utente. La procedura di accensione del sistema dovrebbe essere affissa in modo chiaro, insieme alle precauzioni da usare per quel sistema.

TRATTAMENTO DEI DISCHETTI

Rispettate l'integrità magnetica e fisica dei dischetti. Quando usate i dischetti:

- Non mettete oggetti magnetici a contatto o in prossimità di un dischetto. Oggetti magnetici sono i trasformatori (telefoni), i cacciaviti (molti sono magnetizzati dopo poco tempo) e ogni altro oggetto metallico che potrebbe essersi magnetizzato.
- Riponete sempre i dischetti nella loro busta. Non esponeteli alla polvere, al fumo e ad altre particelle. In particolare non graffiateli, né toccateli, e neppure tentate di pulire la superficie del disco.
- Etichettate sempre i dischi in modo completo; indicate la data e il contenuto. Un consiglio pratico è di generare una lista della direttrice di ciascun disco, ritagliarla e incollarla sulla copertina del disco in modo che il contenuto del disco possa essere conosciuto facilmente.
- Tenete una copia dei dischetti importanti in un posto separato in modo da non rischiare di perdere tutte le informazioni in una volta.
- Non incurvate, piegate, mutilate i dischetti.
- Non esponete i dischetti al caldo o alla luce diretta del sole.
- Scrivete sempre sui dischetti con un pennarello, non usate la penna a sfera perché danneggia il dischetto sottostante.
- Assicuratevi che un dischetto non sia inserito completamente nell'unità al momento dell'accensione o dello spegnimento.
- Fate regolarmente delle copie di sicurezza di tutte le informazioni essenziali.

I dischetti devono essere accuratamente protetti dalle contaminazioni della polvere. Questo è particolarmente importante in un ambiente secco dove si deve avere una cura particolare. La polvere accumula una carica statica che l'attacca, insieme ad altre particelle, alla superficie del dischetto.

Se devono prendere molte precauzioni:

- Non mettete i dischetti vicino a una sorgente di polvere. In particolare questo vale per i magazzini, i depositi, i gabinetti dentistici (vicino al trapano), le classi (con la polvere di gesso) gli ambienti industriali.
- Tenete l'ambiente ragionevolmente pulito, usando frequentemente un aspirapolvere e pulendo le superfici dei tavoli e delle apparecchiature. Se necessario, mettete un filtro sul sistema di riscaldamento o installate un depuratore d'aria elettrostatico per la stanza.
- Cercate di prevenire l'accumulo di elettricità statica nella stanza. Si può usare uno speciale spray antistatico sulla moquette. La soluzione più efficace, comunque, è di usare un umidificatore.

— Naturalmente, è anche importante tenere sempre i dischetti nella loro copertina.

LA STAMPANTE

La maggior parte delle stampanti di buona qualità funzionano ottimamente per mesi o anche anni senza guasti. Comunque, devono essere trattate con rispetto, o piuttosto in un modo molto ordinato. Tutte le regolazioni meccaniche devono essere corrette senza eccezione. Se l'utente non impara l'uso di tutte le leve e non fa le regolazioni corrette, la stampante può non funzionare mai correttamente.

Un esempio di una leva che deve essere verificata è la leva dello "spessore della carta" (indicata con le lettere da A a E sulle stampanti IBM). Se si usa la carta sottile nella stampante e la leva è (accidentalmente) lasciata su D o E (le posizioni per la carta più spessa), la stampante può funzionare male in un modo imprevedibile. Può sembrare che vi sia un problema di interfaccia software, mentre, in effetti, si deve semplicemente fare una regolazione nella leva dello spessore della carta. Mettendo la leva su "A" il problema scompare.

TABULATI

I tabulati possono richiedere una quantità di tempo significativa, poiché la stampante è solitamente il dispositivo di ingresso/uscita più lento connesso al calcolatore. Spesso è conveniente per l'operatore fare altro mentre viene stampato un tabulato. Però, è fortemente raccomandato che l'operatore verifichi frequentemente il corretto funzionamento della stampante. Questo è importante specialmente all'inizio di un tabulato, quando è facile che la carta possa accartocciarsi nella stampante (specialmente se si usano etichette autodesive). Inoltre, se si usano nastri inchiostrati di seta, questi si fermano all'improvviso alla fine del rotolo, invece che dare dei caratteri che scompaiono progressivamente. Come risultato, una parte di un tabulato potrebbe essere completamente bianca. A meno che non serva un'elevata qualità di stampa, sono raccomandati i nastri comuni.

Quando si stampano le etichette, è importante che l'operatore sia presente per tutto il tempo. Questo perché ogni malfunzionamento - compreso un malfunzionamento meccanico della stampante - potrebbe provocare il salto di una riga o l'accartocciamento delle etichette. Ciascuno di questi eventi richiederebbe di far partire l'operazione e potrebbe anche causare danni fisici alla stampante (in caso di accartocciamento).

Ogni volta che si verifica un problema durante la stampa di un tabulato, questo deve essere fatto ripartire. Il caso più frequente consiste nel tentativo di far partire la stampa di un tabulato dalla metà di un file.

Se il file è lungo, il procedimento di recupero consiste nell'usare PIP o ED (descritti nei Capitoli 3 e 4) per selezionare nel file iniziale la porzione che deve essere stampata. Se il file non è molto lungo, però, una soluzione veloce e conveniente consiste nel vedere il file ad alta velocità sul video CRT, poi di far partire la stampante al momento giusto con un CTRL-P, se il programma di stampa permette all'utente di farlo. TYPE lo permette, ma è possibile che altri programmi specializzati di stampa non lo permettano.

Se la stampante non funziona quando il sistema è acceso, verificate tutti i comandi della stampante. (Per esempio, potrebbe trovarsi nello stato di funzionamento locale). Inoltre, se sulla vostra installazione sono state usate versioni diverse del CP/M, verificate che il vostro disco corrisponda al tipo di stampante che state usando effettivamente - questo è un errore comune.

Quando si stampano tabulati lunghi, è possibie all'operatore specificare la stampa di più files e allontanarsi poi dalla stampante per un lungo periodo di tempo (usando, ad esempio, un assegnamento PRN).

Infine, per stampe più veloci, usate PIP invece che TYPE (usate CON: = per una stampa su CRT e LST: = per una stampa su stampante).

FILES

Traboccamento dei files

Se l'estensione di un file supera la capacità di un dischetto, dovrà risiedere su due o più dischetti. In generale, non dovreste passare a un secondo dischetto una volta che il primo è pieno. Cercate di ordinare il vostro file sul primo dischetto in accordo con qualche criterio utile. Per esempio, se il vostro file è un elenco di nomi e indirizzi, ordinatelo alfabeticamente o secondo qualche codice. Se si usa l'ordine alfabetico, il primo dischetto conterrà i nomi dalla A alla L, de il secondo dischetto sarà usato per memorizzare i nomi dalla M alla Z. Ciascun dischetto sarà approssimativamente pieno a metà. Questa è una partizione conveniente del dischetto. Come risultato, sarà ancora possibile, usando un programma appropriato, ottenere un elenco secondo un ordine diverso; comunque, ci saranno due elenchi separati uno per dischetto. Essi non possono essere mescolati a meno che non sia disponibile un disco rigido, cioé, un disco a grande capacità.

Un'altra possibilità, una volta che un file cresce troppo, è di separarlo in sottogruppi. Per esempio, se il file contiene nomi e indirizzi di fornitori e clienti, questi possono essere separati e messi su dischetti differenti.

Ancora un altro problema può verificarsi. Supponete di avere un file di grandi dimensioni, diciamo 170K, che volete ordinare. Il vostro programma di ordinamento è su A, e A è abbastanza pieno: contiene 120K di files. Spesso non sarete in grado di effettuare l'ordinamento,

poiché molti programmi di ordinamento richiedono almeno 170K di "spazio di lavoro" sul disco per ordinare un file di 170K.

La soluzione è semplice: create un nuovo disco di sistema che contenga soltanto il programma (di ordinamento) SORT e usatelo. Avrà abbastanza spazio per fare l'ordinamento. Se il programma di ordinamento non funziona ancora, dovete tagliare il file originale in due più piccoli usando un editor o un altro programma che può estrarre una parte del file (come PIP).

Fusione di files

Ricordate che PIP può essere usato per fondere in modo conveniente due o più files.

Parole sbagliate

È possibile che una parola o un codice debbano essere cambiate all'interno di un file. Il programma editor può realizzarlo in modo conveniente. Si veda per i dettagli il Capitolo 4.

Files danneggiati

A causa di un errore di un operatore o di un malfunzionamento del sistema, un file potrebbe essere danneggiato. Questo potrebbe essersi verificato perché l'operatore ha digitato caratteri di controllo non permessi dal programma o perché si è verificato qualche guasto. Comunque, come effetto, il file non è più caricabile o eseguibile. In molti casi, se avete familiarità con la struttura del file, cioé, il modo in cui il file dovrebbe apparire, e se il file contiene un testo, è possibile recuperare il file con appropriate operazioni di chirurgia realizzate con l'editor. I dettagli sono specifici del file su cui si opera. Spesso, files di testo possono essere ripristinati e salvati. Comunque, se il file è piccolo, solitamente è meglio ridigitarlo piuttosto che tentare di salvarlo, a meno che abbiate già familiarità con una tale operazione.

Quando un file che funzionava correttamente all'improvviso appare danneggiato o non funziona correttamente, si può sospettare un errore dell'operatore. Comunque, un'altra causa frequente è un danno al dischetto di sistema o di programma. Se il dischetto di sistema è stato contaminato o danneggiato in qualche modo, l'informazione che contiene sarà stata modificata e il comportamento del sistema diventerà casuale. Può sembrare che i programmi usuali siano eseguiti normalmente. Comunque, in effetti, alcuni dei comandi non funzionano correttamente e possono danneggiare i files esistenti. Sfortunatamente, questo non potrà essere scoperto fino a quando un file non sarà stato danneggiato in modo significativo. In questo caso, passate a un nuovo dischetto di

sistema e a un nuovo dischetto di programma. Create un nuovo file o usate una copia di sicurezza del vecchio file. Se questo funziona, potete sospettare che uno dei dischetti precedenti sia stato danneggiato e scartarlo.

PROGRAMMI UTILI

Editor

Il Capitolo 4 ha mostrato che un editor è un potente mezzo per operare su files e modificarli. ED fornisce una tale possibilità. Sono disponibili altri editor commerciali che possono essere trovati più potenti o più convenienti.

Duplicatore traccia a traccia

Alcuni programmi utili sono anche solitamente forniti dal produttore del calcolatore o dal controllore del disco. Un programma di duplicazione di dischi solitamente copia un disco da un altro, una traccia per volta. È molto più veloce di PIP quando si deve copiare un intero dischetto. Può essere anche disponibile un editor diretto su disco per fare modifiche sul dischetto ed esaminarlo.

Cancellazione di un dischetto

Può essere necessario cancellare un dischetto per due ragioni: se il dischetto semplicemente deve essere cancellato e non è stato danneggiato, o se la direttrice del dischetto è stata modificata, rendendo inutilizzabile il dischetto (assumendo che il dischetto sia ancora fisicamente intatto).

La cancellazione di un dischetto "buona" può essere realizzata con il comando ERA *.*. Altrimenti, in caso di direttrice danneggiata, il dischetto può essere cancellato con un appropriato programma di inizializzazione. Un tale programma, spesso chiamato INIT, è fornito dal produttore del calcolatore o dal controllore del disco ed è differente per ciascun calcolatore. INIT è uno strumento utile per cancellare i dischetti velocemente.

Sequenza di comandi

Quando si esegue frequentemente una sequenza di comandi, si può creare il file di comando SUBMIT. Il file è costituito dalla sequenza di comandi che deve essere eseguita nel calcolatore e deve essere creato da un editor come ED. Digitando SUBMIT, seguito dal nome del programma, la sequenza sarà eseguita automaticamente. Per esempio, se un certo

programma è usato frequentemente e richiede la digitazione di alcuni parametri standard prima di poter essere usato, la sequenza può facilmente essere digitata in un file chiamato START.SUB. L'operatore dovrà soltanto digitare la riga seguente per eseguire questo programma:

```
SUBMIT START ↵
```

STOP

Quando si verifica qualche errore, è possibile che vogliate interrompere tutto. Non togliete la spina dalla presa di corrente. Prima, provate CTRL-C. Se non funziona, usate RESET. Ricordate, comunque, che perderete ogni informazione contenuta nella memoria del calcolatore. Non danneggerete i files. Per interrompere la stampante, potete spegnerla fisicamente.

CONSIGLI VARI

Ricordate che il CP/M stesso "non occupa spazio" sul dischetto. Più esattamente, due tracce sono sempre "occupate" su un dischetto a 8 pollici, che installiate il CP/M o no. Spesso è conveniente mettere il CP/M sulla maggior parte dei dischetti quando sono vuoti. In questo modo, potete fare il bootstrap da qualsiasi dischetto. Inoltre non rischiate di danneggiare i files quando scambiate i dischi con PIP se fate un'operazione sbagliata.

Se lavorate con dischetti duplicati, PIP è l'altro programma di maggior utilità.

I SETTE COMANDAMENTI DOPO UN ERRORE DI SISTEMA

Sospettate dell'operatore per prima cosa

1. Verificate la meccanica:
   — Tutti gli interruttori sono in posizione corretta? (Verificate sistematicamente. Nessuna eccezione).
   — I fusibili sono intatti?
   — I cavi sono tutti attaccati senza connessioni staccate o strappate?
2. Avete dato il comando corretto?
   — Spegnete tutto. Ora, accendete il sistema.
   — Ripetete il comando.

Sospettate del dischetto

3. Usate un dischetto nuovo. (Spesso, il dischetto corrente è stato dan-

neggiato per una gestione scorretta e causerà un comportamento casuale del sistema.

— Usate un dischetto di sicurezza. Non usate programmi sul vostro dischetto solito.

— Se non esiste una singola copia di sicurezza completa, perdete tempo a generarne una.

Sospettate del software

4. Assicuratevi che state usando i programmi corretti:
   — La versione corretta del CP/M se ne avete più di una.
   — Il compilatore/interprete corretto per il vostro programma applicativo. Per esempio, può essere richiesto CBASIC versione 2.
   — Il programa applicativo corretto.

   Molti programmi applicativi, in particolare per il trattamento dei testi, devono essere adattati al terminale e alla stampante. Se questo non viene fatto, è possibile che alcuni tasti sul termiale non funzionino e che non siate in grado di stampare.

Infine sospettate dell'hardware

5. Verificate di nuovo la meccanica, molto accuratamente.
   — In particolare, rimuovete le schede, pulite le connessioni se necessario e inseritele di nuovo.
   — Se la causa del malfunzionamento può essere attribuita a una scheda, rimuovete i componenti dagli zoccoli, pulite le connessioni e reinseriteli.
6. Cercate sempre di identificare il dispositivo di cui si sospetta il malfunzionamento scambiandolo con uno che si sa che funziona: scambiate schede, stampanti, ecc. Questo vi darà risultati positivi e vi farà risparmiare tempo.

   Non accusate mai un dispositivo fino a che non avete verificato sostituendolo con uno buono. Si potrebbero altrimenti sprecare molti sforzi.
7. Da questo momento, usate le tecniche corrette di prevenzione, come spiegato in questo libro. In breve:
   — Siate sempre disciplinati.
   — Non usate scorciatoie. Non fate eccezioni alle regole.

Infine, troverete molti elenchi di regole utili nella sezione delle Appendici di questo libro. Ricordate che la prevenzione è la cosa più importante. La disciplina dell'utente è la chiave essenziale per usare con successo un calcolatore.

CAPITOLO 8

IL FUTURO

STORIA DEL CP/M

Il sistema operativo CP/M è stato creato da Gary Kildall. Quando lavorava come consulente per la Intel Corporation, Gary Kildall scrisse il primo compilatore di linguaggio ad alto livello prodotto dalla Intel, il PL/M. Poi, nel 1974, creò la sua prima versione di un file system CP/M che era progettato (in quel tempo) per fare da supporto a un compilatore residente PL/M.

Il CP/M fece la sua prima apparizione commerciale nel 1975 quando furono stipulati i primi contratti di licenza, ma passò relativamente inosservato per almeno un anno. Durante questo periodo, furono sviluppate le prime versioni dell'editor (ED), dell'assembler (ASM) e del debugger (DDT). Il primo utente commerciale su larga scala di questo sistema operativo fu l'IMSAI (ora scomparso), che ebbe la licenza di distribuire il CP/M versione 1.3 che evolvette in quello che l'IMSAI chiamò IMDOS. Il CP/M è ora evoluto nel CP/M versione 2.2 (e in quelle successive), che è progettata per trarre vantaggio dalle grandi capacità di memorizzazione dei dischi rigidi ora disponibili. L'MP/M è stato progettato per fornire un ambiente multi utente a divisione di tempo per i sistemi a multi-programmazione.

In questo momento, il CP/M è probabilmente uno dei sistemi operativi più frequentemente usati sui microcalcolatori. Sebbene possa essere criticato dagli utenti progettisti di sistemi operativi che hanno familiarità con macchine più potenti, esso svolge bene il suo compito ed è diventato di fatto uno standard per molti utenti di microcalcolatori.

IL CP/M E GLI ALTRI SISTEMI OPERATIVI

Lo sviluppo di un *buon* sistema operativo ha sempre rappresentato un buon investimento. Di conseguenza, sistemi operativi potenti a divisione di tempo sono stati sviluppati soltanto per pochi grandi calcolatori. Il tipo più complesso di sistema operativo è un sistema a divisione di tempo con una potente gestione di processi e strategia di protezione. Sebbene non vi sia consenso su quale sia il miglior sistema operativo, il sistema

UNIX è uno di quelli che ha avuto grande popolarità nel campo dei minicalcolatori a 16-bit. Si è tentato più volte di realizzare una versione di UNIX sui microcalcolatori a 16-bit. Comunque, l'investimento richiesto per rendere questi microcalcolatori completamente compatibili con un sistema tipo UNIX è molto grande e la probabilità di un successo completo è limitata.

Uno dei maggiori vantaggi del CP/M (a parte la convenienza) è il fatto che tutti i programmi e i files compatibili con il CP/M posson essere cambiati fra gli utenti. Il CP/M ha la virtù di ogni sistema operativo standard: la compatibilità. Per questa ragione, il CP/M sarà probabilmente usato ancora per molto tempo - fino a quando saranno costruiti i processi sui quali risiede abitualmente.

EVOLUZIONE

Poiché è sempre possibile che a un grande programma esistente vengano fatti miglioramenti (e correzioni), il CP/M e l'MP/M continueranno ad evolvere. Comunque, le versioni successive di questi due sistemi sono solitamente compatibili con quelle precedenti. Questo significa in pratica, che la maggior parte delle conoscenze che potete aver conseguito leggendo questo libro dovrebbero essere applicabili a qualsiasi versione futura del CP/M e dell'MP/M che sarà realizzata. Inoltre, usando e comprendendo il CP/M, capirete le funzioni di un sistema operativo "standard". Una volta che queste funzioni sono state capite, dovreste essere in grado di adattarle facilmente a un altro sistema operativo.

CONCLUSIONE

Dopo aver letto questo libro, dovreste essere in grado di usare con profitto il vostro calcolatore dotato di CP/M. *Il comando del CP/M (e dell'MP/M)* è stato progettato per insegnarvi come usare il vostro sistema e per aiutarvi a capire come funziona.

Quando imparerete ad usare qualsiasi programma del tipo del CP/M, ricordate che la disciplina è la chiave per usare senza problemi un calcolatore. Seguendo procedure corrette, si eviteranno errori e problemi. Particolarmente all'inizio, seguite tutte le regole presentate nel testo, fedelmente e senza eccezioni. Quando sarete più esperti, potrete modificare o ignorare qualche regola. L'uso corretto di un piccolo calcolatore e delle sue periferiche è il soggetto di un altro libro dell'autore.

Usando questo libro, potete far riferimento a uno qualsiasi dei capitoli per aumentare la vostra comprensione di un argomento specifico. Ad eccezione del primo capitolo, non è necessario che ricordiate a memoria tutto il contenuto di nessun capitolo; potete semplicemente imparare quelle caratteristiche che vi interessano e usare i riassunti nella sezione

delle appendici come riferimenti veloci.

Usando il calcolatore, imparerete a poco a poco quasi tutto sulle sue risorse. Per esempio, anche se non prevedete di usare l'editor adesso, dovreste provarlo. Sarete poi in grado di imparare facilmente ad usare un programma di trattamento dei testi o di adattare o valutare un nuovo programma commerciale.

Una volta che avrete acquistato familiarità con tutti i concetti e le tecniche presentate in questo libro, sarete utenti di calcolatore competenti. Dovreste allora essere in grado di adattarvi rapidamente ad altri programmi o sistemi operativi simili.

APPENDICE A

MESSAGGI DI ERRORE COMUNI DEL CP/M

Ci sono tre condizioni di errore che sono comuni al sistema. Queste condizioni sono indicate attraverso lo stesso messaggio generale:

BDOS ERR ON *u:errore*

dove *u* è una lettera che indica l'unità a dischi in cui si è verificato l'errore e *errore* è uno dei seguenti messaggi di errore:

BAD SECTOR
SELECT
READ ONLY

C'é anche una quarta condizione di errore per il CP/M versione 2.2 dove *errore* è il messaggio:

FILE R/O

(Per le descrizioni di errore di ASM, DDT, ED e di altri programmi, consultate la documentazione fornita con il programma relativo).

BAD SECTOR

Un errore di "bad sector" (settore difettoso) si verifica se il controllore del disco non riesce a trovare l'informazione sul dischetto. Questo accade quando il dischetto è consumato (ha un settore difettoso) o se il controllore a dischi presenta dei malfunzionamenti. Un'altra causa può essere che il dischetto non è caricato nell'unità quando cercate di accedervi. Potete avere questo errore anche se cercate di leggere files che sono stati scritti sul dischetto da un controllore diverso da quello che usate attualmente. Anche se i controllori del disco sono dischiarati "compatibili IBM", potrebbero esservi piccole differenze nei formati dei records. Per esempio, files scritti su dischetti usando il controllore dell'Intel MDS-800 potrebbero essere letti da un altro controllore, ma files scritti da un altro controllore potrebbero produrre l'errore BAD SECTOR quando si cerca di leggerli sul controllore dell'MDS-800. Questo errore

si verifica anche se le informazioni in un file sono state danneggiate a causa di un uso errato del dischetto o per un programma danneggiato o errato.

Quando si verifica questo errore, potete fare un ↑ C (CTRL-C per rilanciare il sistema), che interrompe il programma o la gestione del file e vi riporta al sistema, oppure potete decidere di ignorare l'errore e continuare l'esecuzione del programma o la gestione del file, premendo RETURN, che dice al sistema di ignorare il settore difettoso.

Potrebbe non essere sicuro ignorare l'errore! Se il vostro programma o l'operazione sul file coinvolge operazioni di scrittura della direttrice, ignorando l'errore potreste distruggere l'integrità del dischetto. Assicuratevi di avere copie di sicurezza adeguate.

SELECT

Questo errore si verifica quando selezionate un'unità a dischi che non esiste. Il valore di *u* è l'unità selezionata, che è errata. Il sistema automaticamente viene ricaricato quando premete qualsiasi tasto sul terminale.

READ ONLY

Questo errore si verifica quando cercate di scrivere su un dischetto che è stato designato come dischetto a "sola lettura" pre mezzo del comando STAT (o da un programma che usa la funzione BDOS). Questo errore può anche verificarsi se inserite un dischetto nuovo senza fare un ↑ C per ricaricare il sistema (e cambiare la mappa del dischetto); dovete fare un ↑ C ad ogni inserzione di un nuovo dischetto per poter scrivere su di esso (riscrivere, cancellare, creare o aggiornare files).

Premendo un qualsiasi tasto sul terminale, potete uscire da questa condizione di errore e automaticamente realizzare un caricamento del sistema (↑ C), che cambia *anche* il dischetto in un dischetto a lettura-scrittura (cioé, un dischetto su cui potete leggere e scrivere).

FILE R/O

Questo errore si verifica solo nelle ultime versioni del CP/M (CP/M versione 2.2 e successive), quando cercate di scrivere (riscrivere, modificare o cancellare) un file che ha l'attributo $R/O (sola-lettura, assegnato dal comando STAT o da un programma utente). L'attributo $R/O è descritto a fondo nel Capitolo 2, nella sezione sul CP/M versione 2.2 e sull'MP/M.

Per uscire da questo errore, potete premere un qualsiasi tasto sul terminale. L'operazione che coinvolge il file a sola-lettura è interrotta e dovete cambiare l'attributo $R/O in $R/W per poter scrivere sul file. Usate il comando STAT per cambiare gli attributi del file.

APPENDICE B

TABELLA DI CONVERSIONE ESADECIMALE

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

APPENDICE C

TABELLA DI CONVERSIONE ASCII

NUMERO DEI BIT →				0	0	0	0	1	1	1	1
				0	0	1	1	0	0	1	1
				0	1	0	1	0	1	0	1
b_7	b_6	b_5	b_4	b_3	b_2	b_1	HEX 1 / HEX 0	0	1	2	3	4	5	6	7
			0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
			0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
			0	0	1	0	2	STX	DC2	"	2	B	R	b	r
			0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
			0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
			0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
			0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
			0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
			1	0	0	0	8	BS	CAN	(	8	H	X	h	x
			1	0	0	1	9	HT	EM	)	9	I	Y	i	y
			1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
			1	0	1	1	11	VT	ESC	+	;	K	[	k	{
			1	1	0	0	12	FF	FS	,	<	L	\	l	¦
			1	1	0	1	13	CR	GS	-	=	M	]	m	}
			1	1	1	0	14	SO	RS	.	>	N	^	n	~
			1	1	1	1	15	SI	US	/	?	O	_	o	DEL

I SIMBOLI ASCII

NUL — Nullo
SOH — Partenza del primo blocco
STH — Partenza del testo
ETX — Fine del testo
EOT — Fine della trasmissione
ENQ — Interrogazione
ACK — Avvisatore di ricevimento
BEL — Richiamo
BS — Ritorno indietro
HT — Tabulazione orizzontale
LF — Avanzamento di un'interlinea
VT — Tabulazione verticale
FF — Alimentazione di carta
CR — Ritorno carrello
SO — Codice semplice
SI — Codice normale
DLE — Uscita trasmissione
DC — Controllo della periferica
NAK — Avvisatore di ricevimento-negativo
SYN — Aspettativa sincronizzazione
ETB — Fine del blocco di trasmissione
CAN — Annullamento
EM — Fine del supporto
SUB — Rimpiazzamento
ESC — Cambiamento di codice
FS — Separatore di file
GS — Separatore di gruppo
RS — Separatore di record
US — Separatore di unità
SP — Spazio (black)
DEL — Cancellazione

APPENDICE D

CARATTERI DI CONTROLLO DI ED

Tasti	Significato
CTRL-C (↑ C)	Rilancio del sistema (partenza a caldo), restituisce il prompt di sistema.
CTRL-E (↑ E)	Muove il cursore alla riga successvia per continuare la riga di comando (senza eseguire o trasmettere la riga).
*CTRL-H (↑ H)	*Riporta il cursore sul carattere precedente per cancellare l'ultimo carattere digitato.
CTRL-I (↑ I)	Muove il cursore dello spazio di una tabulazione (lungo 7 colonne).
*CTRL-J (↑ J)	*Realizza un RETURN.
*CTRL-M (↑ M)	*Realizza un RETURN.
CTRL-L (↑ L)	Sostituisce la sequenza di ritorno carrello generata da un RETURN nelle stringhe usate con il comando di ricerca e di sostituzione.
*CTRL-R (↑ R)	*Ristampa la riga corrente (stampa una riga pulita).
CTRL-U (↑ U)	Cancella la riga corrente.
*CTRL-X (↑ X)	*Riporta il cursore all'inizio della riga corrente e la cancella.
**CTRL-D (↑ D)	**Distacca il programma corrente dal terminale.
RETURN (↵)	Trasmette (esegue) la riga corrente, o genera un ritorno carrello per separare le righe del file di testo.
RUBOUT o DELETE	Cancella l'ultimo carattere digitato (con l'eco del carattere).
CTRL-Z (↑ Z)	Termina l'inserzione con il comando I o separa stringhe di testo nella ricerca o nella sostituzione o mette una marca alla fine del file di testo.
CTRL-P (↑ P)	Fa l'eco sulla stampante di tutto quello che viene digitato o visualizzato.
CTRL-S (↑ S)	Interrompe temporaneamente una stampa lunga (premere un tasto qualsiasi per continuarla).
BREAK	Interrompe l'esecuzione di un qualsiasi comando ED.

*CP/M 2.2
**MP/M

APPENDICE E

COMANDI DI ED

Comandi	Significato
nA	Appende n righe (o una riga se manca n) al buffer di edit, prendendolo dal file sorgente. Un "#" al posto di n appende 65535 righe (riempie il buffer) e uno zero al posto di n appende righe fino a riempire metà del buffer (il numero di righe dipende dall'estensione del buffer e dal sistema).
+/- nB	Muove il CP* all'inizio del buffer nel caso +B e alla fine nel caso -B.
+/- nC	Muove il CP avanti di (+) n caratteri o indietro di (-) n caratteri. Il CP conta una sequenza di "ritorno carrello" come due caratteri (RETURN e LINE FEED).
+/- nD	Cancella n caratteri avanti (+), incluso il CP; o cancella n caratteri indietro (-), escluso il CP. Se manca n, cancella soltanto il carattere puntato dal CP.
E	Termina normalmente la sessione di ED. Il comando E salva il testo nel buffer e il resto del testo nel file sorgente in un file di uscita temporaneo, poi cambia il nome del file di uscita nel nome del file sorgente (copiando il file sorgente in un file di sicurezza ".BAK" per salvare la copia originale). Quindi ED termina, passando il controllo al sistema.
nFstringa {↑Z ⊐}	Trova la stringa di caratteri n volte (se manca n, una sola volta). F cerca nel buffer dopo il CP e sposta il CP alla fine della riga trovata. Fate seguire la stringa dal terminatore Z se volete aggiungere altri comandi di ED; altrimenti usate RETURN per terminare la stringa.
H	Termina la sessione di ED, realizzando un comando E, quindi esegue ED nuovamente sul nuovo file sorgente (salva gli aggiornamenti al file e ritorna in editing).

NOTA: potete sostituire il carattere "#" a n in qualsiasi comando ED, dando così ad n il più alto valore che può avere: 65535.

* CP è il puntatore di carattere.

I ⏎	Inserisce una nuova riga di testo dopo il CP, spostandolo alla fine dell'ultima riga inserita. Nel CP/M 2.2: - Se nel comando si usa la I maiuscola, tutto il testo inserito sarà in caratteri maiuscoli. - Se si usa la i minuscola, allora il testo sarà inserito in caratteri maiuscoli e minuscoli.
Itesto ↑ Z	Inserisce caratteri dopo il CP, spostando il CP alla fine dell'ultimo carattere inserito.
nJstringa1 ↑ Zstringa2 ↑ Zstringa3 {↑ Z / ⏎}	Giustappone stringhe di testo cercando stringa1, inserendo stringa2 alla fine di stringa1 e cancellando tutti i caratteri fino a stringa3 esclusa (giustappone le tre stringhe di testo). CP è spostato all'inizio di stringa3.
+/- nK	Cancella (+) n righe seguenti, compreso il CP e i caratteri che lo seguono sulla riga corrente oppure cancella (-) n righe precedenti compresi i caratteri che precedono il CP sulla riga corrente.
+/- nL	Sposta il CP all'inizio della riga corrente se n è zero; altrimenti sposta il CP all'inizio della riga corrente e lo muove in avanti (+) n righe o indietro di (-) n righe.
nMstringa {↑ Z / ⏎}	Ripete l'esecuzione della stringa di comandi di ED n volte, se n è maggiore di 1. Se n è 0 o 1, M esegue la stringa di comandi ripetutamente fino a quando si verifica un errore.
nNtesto {↑ Z / ⏎}	Cerca l'ennesima occorrenza di testo nel buffer e nel file sorgente (terminate il testo con RETURN o Z per aggiungere altri comandi). N muove il CP alla fine del testo trovato. N appende righe del file sorgente fino a quando trova il testo.
O	Chiude la sessione di ED e torna al file sorgente originale.
+/- nP	Muove il CP e manda su video e su stampante pagine di testo contenute nel buffer. n è il numero di pagine (24 righe per pagina) stampate; +n stampa n pagine a partire dal CP e -n stampa n pagine precedenti il CP. 0P (0 al posto di n) stampa la riga e la pagina corrente (le prime 23 righe dopo la riga corrente). CP è spostato all'inizio della pagina stampata.
Q	Chiude senza modifiche i files (Lascia infatti il file temporaneo, il file sorgente e il buffer). Q restituisce il controllo al sistema. Non viene creato un file ".BAK" con lo stesso nome, viene cancellato. (Attenzione!).

R	Legge dal file X$$$$$$$.LIB e inserisce le righe a partire dal CP, portandolo alla fine delle righe inserite (non svuota il file ".LIB").
Rnomefile	Legge righe dal file nomefile.LIB e le inserisce, a partire dal CP, spostandolo alla fine delle righe inserite (non svuota il file ".LIB").
nSvecchiotesto ↑ Znuovotesto {↑ Z / ↵}	Cerca vecchiotesto nel buffer a partire dal CP e lo sostituisce con nuovotesto; ripete la sequenza n volte se n è maggiore di 1.
+/- nT	Se n non è specificato o se è 1, stampa i caratteri seguenti il CP fino alla fine della riga. Se n è 0, stampa i caratteri sulla riga corrente fino al CP escluso. Se n è positivo (+), stampa le n righe seguenti compresa la riga corrente. Se n è negativo (-) stampa le n righe precedenti esclusa la riga corrente e i caratteri sulla riga corrente fino al CP escluso. La sequenza di comandi "B#T" stampa l'intero buffer.
+/- U	Traduce tutti i caratteri minuscoli di ingresso (digitati o inseriti) in maiuscoli nel caso +U, oppure termina la traduzione nel caso -U.
V	Attiva la stampa dei numeri di riga per le righe nel buffer (i numeri di riga non sono effettivamente nel file).
0V	Stampa il numero di bytes liberi rimasti nel buffer e l'estensione totale di memoria nel buffer (numeri decimali). Per esempio, nella stampa "27648/28832", "27648" è il numero di bytes liberi e "28832" è il numero totale di bytes nel buffer corrente.
nW	Scrive nel buffer temporaneo di uscita con estensione ".$$$" le n righe seguenti a partire dal CP (compresa la riga corrente). Se manca n, scrive soltanto la riga corrente.
nX	Copia le n righe seguenti di testo nel file X$$$$$$$.LIB (non cancella le righe originali). Potete rileggere le righe usando il comando R. Se n è 0, il comando cancella tutti i files X$$$$$$$.LIB.
**nZ	Sospende il programma ED per n secondi (approssimativamente).
+/- n	Realizza la sequenza di comandi "+/- nLT".
n:	Sposta il CP all'inizio del numero di riga n.
n1::n2	Specifica un campo di numeri di riga a partire da n1 fino a n2. Se manca n1 o n2, li sostituiscono con la riga corrente.
* CP è il puntatore di carattere.	

APPENDICE F

NOMI DEI DISPOSITIVI DI PIP

DISPOSITIVI LOGICI	
CON:	"console" o terminale comprendente tastiera o video (ingresso/uscita).
RDR:	lettore di nastro di carta o di schede (soltanto ingresso).
PUN:	perforatore di nastro di carta o di schede (soltanto uscita).
LST:	dispositivo di stampa di tabulati, ad esempio una stampante (soltanto uscita).

DISPOSITIVI FISICI	
TTY:	per una console o un terminale, un lettore, un perforatore o un dispositivo di stampa (teletype).
CRT:	una console, un terminale o un dispositivo di stampa (Tubo a raggi catodici).
PTR:	un dispositivo di lettura di nastro di carta o di schede.
PTP:	un dispositivo di perforazione di nastro di carta o di schede.
LPT:	un dispositivo di stampa (stampante).
UC1:	una console o un terminale definito dall'utente.
UR1:	un lettore definito dall'utente.
UR2:	un secondo lettore definito dall'utente.
UP1:	un dispositivo di uscita (perforatore) definito dall'utente.
UP2:	un secondo dispositivo di uscita (perforatore) definito dall'utente.
UL1:	un dispositivo di stampa definito dall'utente.
NOTA:	BAT: non è compreso, perché riassegna soltanto i valori a RDR: e LST: (si veda "Assegnamento dei dispositivi").

APPENDICE G

PAROLE CHIAVE DI PIP

NUL:	manda 40 "nulls" (codice ASCII 0) al dipositivo di uscita, solitamente un perforatore. Esempio, dove PROG.HEX è mandato al perforatore: * PUN: = PROG.HEX,NULL: ↵
EOF:	manda un "end-of-file" (fine del file, carattere ASCII ↑Z) al dispositivo (mandato automaticamente da PIP durante il trasferimento di un file di testo ASCII e necessario soltanto in casi speciali). Esempio: * PUN: = NUL:,X.ASM EOF:,NULL: ↵ Questo esempio manda 40 nulls al dispositivo di perforazione seguito da una copia del file X.ASM, seguito dal carattere end-of-file (↑Z) e da altri 40 nulls.
PRN:	lo stesso di LST: (manda alla stampante), tranne che le tabulazioni sono espanse ogni otto caratteri, le righe sono numerate (come nel programma ED) e vengono inseriti salti pagina ogni 40 righe (per fare avanzare la carta della stampante alla pagina successiva), con un salto pagina iniziale. Esempio: * PRN: = SAMPLE.TXT ↵
INP:	codice speciale di un dispositivo di ingresso che può essere collegato al programma PIP stesso (dovete scrivere il programma di collegamento in linguaggio assembler e aggiungerlo a PIP). PIP riceve i caratteri di ingresso uno per uno chiamando una locazione in memoria (103H) e memorizzando i dati a partire dalla locazione 109H (il bit di parità deve essere 0 - usate il parametro Z).
OUT:	codice speciale di un dispositivo di uscita che può essere collegato al programma PIP stesso, come INP: descritto sopra. PIP chiama la locazione 106H e manda il dato nel registro C (ciascun carattere). Nota per i programmatori in linguaggio assembler: le locazioni dell'immagine di memoria di PIP da 109H a 1FFH non sono usate e possono essere sostituite con un codice per pilotare dispositivi speciali (usate DDT - il debugger del CP/M fornito dalla Digital Research con il CP/M o l'MP/M). Esempi: * GIZMO.CLK = INP: ↵ *(i caratteri di ingresso del dispositivo speciale sono memorizzati nel file GIZMO.CLK).* * OUT: = GIZMO.CLK ↵ *(una copia di GIZMO.CLK è mandata al dispositivo speciale).*

APPENDICE H

PARAMETRI DI PIP

B	Trasferimento a blocchi. PIP mette i dati in un buffer fino a quando legge il carattere ASCII "x-off" (↑ S) dal dispositivo. PIP a questo punto svuota il buffer e torna a ricevere gli altri dati. L'estensione del buffer dipende dalle dimensioni del sistema (vedi la documentazione fornita con il sistema). Usate questo parametro per trasferire dati da un dispositivo a lettura continua con un lettore di cassetta. Esempio: * ENUFF.TXT = RDR: [B] ↵
Dn	PIP, nel copiare files di testo, cancella i caratteri che si trovano dopo la colonna n (colonna verticale sul terminale). Usatelo per troncare le righe se mandate un file a un dispositivo stretto. Esempio: * PRN: = LONG.TXT [D52] ↵
E	Fa l'eco (ristampa) sul terminale video di tutte le operazioni di copiatura, al momento in cui vengono realizzate. Esempio: * COPY.TXT = SOURCE.TXT,S2,TXT,S3.TXT, S4.TXT [E] ↵
F	PIP filtra, nel file, i caratteri di salto pagina (cioé, li toglie). Potete anche usare il parametro P per inserire nuovi salti pagina.
Gn	Prende un file dall'area utente n (CP/M versione 2.2 e MP/M).
H	Trasferimento di dati esadecimali: PIP verifica che tutti i dati siano nel formato esadecimale Intel corretto (vedi "Note sulla copiatura di files in codice macchina (HEX)").
I	Ignora i records ":00" nel trasferimento dei files in formato esadecimale Intel (attiva automaticamente il parametro H).
L	Traduce tutti i caratteri maiuscoli in caratteri minuscoli.
N	Aggiunge i numeri a ciascuna riga copiata dal nuovo file (a partire dalla riga 1). Ciascun numero di righe è seguito da un :. Gli zeri a sinistra (ad esempio 003) sono cancellati, a meno che specifichiate il parametro "N2". "N2" lascia gli zeri a sinistra e inserisce uno spazio di tabulazione dopo il numero. Potete espandere gli spazi di tabulazione usando il parametro T.
O	Trasferimento di un file oggetto (per i files non ASCII): PIP ignora la fine fisica del file durante la concatenazione (vedi "Concatenazione di files" nel Capitolo 2).

Pn	PIP inserisce i salti pagina ogni n righe (con un salto pagina iniziale). Se n è 1 (o se non specificate n) i salti pagina si verificano ogni 60 righe. Se usate anche il parametro F, PIP rimuove i salti pagina precedenti inserendo di nuovi.
Qstringa ↑ Z	PIP smette di copiare dal dispositivo o dal file quando trova la stringa di caratteri specificata (una stringa è un gruppo di caratteri; ad esempio, STRINGA105%). Terminate la stringa con un ↑ Z (CTRL e Z contemporaneamente). Vedi "Copiatura di porzioni di files" nel Capitolo 2.
R	Legge (copia) files di sistema ($SYS); realizza anche l'operazione corrispondente al parametro "W" (CP/M versione 2.2 e MP/M).
Sstringa ↑ Z	PIP incomincia a copiare dal dispositivo o dal file quando trova la stringa di caratteri specificata. Terminate la stringa con un ↑ Z. Vedi "Copiatura di porzioni di files" nel Capitolo 2.
Tn	Espande gli spazi di tabulazione ogni n colonne, durante il trasferimento di files di testo. Potete creare uno spazio di tabulazione in un file di testo usando ↑ I;questo parametro espande gli spazi di tabulazione rispetto alla solita quantità fissa di colonne.
U	Traduce tutti i caratteri minuscoli in caratteri maiuscoli durante la copiatura di files di testo.
V	PIP verifica che i dati siano stato copiati correttamente rileggendo successivamente la copia del file (il file di copia non può essere un dispositivo) e stampando un messaggio se l'operazione ha avuto successo.
W	Riscrive (cancella) files a sola lettura (ignora l'attributo $/R). (Soltanto CP/M versione 2.2 e MP/M).
Z	Mette a zero il bit di parità nei caratteri di ingresso ASCII. Usate questo parametro specialmente se state inserendo dati dal dispositivo INP:.

Ecco alcuni esempi di espressioni PIP con parametri:

```
* LST: = ESEMPIO.TXT [NT8P60] ↵
```

Questa espressione manda il file ESEMPIO.TXT al dispositivo di stampa (LST:), con i numeri di riga, le tabulazioni espanse ogni otto colonne e i salti pagina ogni 60 righe.
NOTA: il dispositivo PRN: assomma automaticamente questi caratteri; se il dispositivo di stampa era stato assegnato a PRN:, l'esempio sopra potrebbe essere riscritto:

```
* PRN: = ESEMPIO.TXT ↵
```

APPENDICE I

RIASSUNTO DEI COMANDI DEL CP/M (E DELL'MP/M)

COMANDI	CP/M VERSIONE 1.4	CP/M VERSIONE 2.2	MP/M VERSIONE 1
ABORT			X
ASM	X	X	X
ATTACH			X
CONSOLE			X
DDT	X	X	X
DIR	X	X	X
DSKRESET			X
DUMP	X	X	X
ED	X	X	X
ERA	X	X	X
ERAQ			X
GENHEX			X
GENMOD			X
GENSYS	X	X	X
LOAD	X	X	X
MOVCPM	X	X	X
MPMLDR			X
MPMSTAT			X
PIP	X	X	X

COMANDI	CP/M VERSIONE 1.4	CP/M VERSIONE 2.2	MP/M VERSIONE 1
PRLCOM			X
REN	X	X	X
SAVE	X	X	X
SCHED			X
SPOOL			X
STAT	X	X	X
STOPSPLR			X
SUBMIT	X	X	X
SYSGEN	X	X	X
TOD			X
TYPE	X	X	X
USER			X
XSUB		X	X

APPENDICE J

TASTI DI CONTROLLO PER LA DIGITAZIONE DEI COMANDI

CONTROLLI COMUNI

rubout/delete	cancella l'ultimo carattere facendone l'eco
CTRL-U o CTRL-X	cancella la riga
CTRL-R	ristampa la riga
CTRL-E	continua sulla riga successiva
CTRL-C	rilancia il CP/M

ALTRI

CTRL-D	distacca la console (MP/M)
CTRL-H	spostamento indietro di una posizione
CTRL-J (line feed)	fine ingresso dati
CTRL-M (carriage return)	fine comando
CTRL-P	attivazione/disattivazione stampante
CTRL-Q	acquisisce la stampante (MP/M)
CTRL-S	interruzione/rilancio dell'uscita su console

APPENDICE K

TIPI DI ESTENSIONE

Estensione	Tipo	Esempio
COM	Obbligatorio per un file che contiene un comando transitorio (programma).	PIP.COM LOAD.COM
ASM	Obbligatorio per i files sorgente (testi) in linguaggio assembler usati con il comando ASM.	PROG.1.ASM PATCH.ASM
PRN	Obbligatorio per i files di stampa dei programmi in linguaggio assembler.	PROG1.PRN PATCH.PRN
PRL	Obbligatorio per i programmi rilocabili MP/M.	RDT.PRL
HEX	Obbligatorio per i files che contengono programmi in formato esadecimale (linguaggio macchina), pronti per essere caricati.	PROG1.HEX PATCH.HEX
RSP	Obbligatorio per i "programmi di sistema residenti" dell'MP/M.	SPOOL.RSP
BAS	Obbligatorio per i files sorgente (testi) dei programmi BASIC.	PROGBAS.BAS
INT	Obbligatorio per i files intermedi dei programmi BASIC, pronti per l'esecuzione (già compilati).	PROGBAS.INT
BAK	Creato da ED come copia di backup di un file prima di modificarlo.	LETTER.BAK
$$$	Files temporanei creati e normalmente cancellati da ED e da altri programmi.	LETTER.$$$
SUB	Files di testo con comandi interni e transitori del CP/M o programmi; da eseguirsi in modo "batch" per mezzo del programma SUBMIT.	TRANSFORM.SUB

APPENDICE L

LISTA DEI MATERIALI

- ☐ Dischetti vuoti
- ☐ Testine di stampa
- ☐ Nastri inchiostrati
- ☐ Carta per stampante
- ☐ Manuale del calcolatore
- ☐ Manuale della stampante
- ☐ Manuale del terminale CRT
- ☐ Documentazione del CP/M
- ☐ Documentazione dei programmi applicativi
- ☐ Dischetto di sistema
- ☐ Dischetto dei programmi applicativi

APPENDICE M

ORGANIZZAZIONE DELLA STANZA DEL CALCOLATORE

- ☐ Ventilazione sufficiente
- ☐ Nessun oggetto sulle uscite della ventilazione del calcolatore
- ☐ Contenitori non metallici per i dischetti
- ☐ Materiale del calcolatore in quantità sufficiente (vedere l'elenco separato)
- ☐ Tutti i manuali necessari
- ☐ Promemoria della configurazione corretta del terminale CRT
- ☐ Promemoria della configurazione corretta della stampante
- ☐ Promemoria di manutenzione
- ☐ Numeri di telefeno della manutenzione e dell'assistenza
- ☐ Nessun telefono vicino all'area di lavoro (un telefono che suona sopra un'unità a dischetti o a dischi distrugge il contenuto dei dischetti)
- ☐ Nessun cacciavite vicino all'area di lavoro (a causa della magnetizzazione)
- ☐ Nessun liquido nella stanza del calcolatore a meno che non siate ben assicurati
- ☐ Non fumare in prossimità dell'unità a dischi
- ☐ Non muovere o scuotere l'unità a dischi
- ☐ Procedure di accensione ben in vista
- ☐ Pavimentazione antistatica

VERIFICHE IN CASO DI ERRORE

NIENTE FUNZIONA

☐ Verifica le connessioni meccaniche:

- ☐ Cavi di alimentazione.
- ☐ Cavi di connessione.
- ☐ Interruttori.
- ☐ Fusibili.

STAMPANTE NON FUNZIONANTE

☐ Provare la stampante in "locale".
☐ Eseguire CTRL P dalla console.
☐ Verificare la configurazione.
☐ Reinserire la carta in modo corretto.
☐ Verificare i fusibili.

LA STAMPANTE NON SI FERMA

☐ Premere CTRL P.
☐ Premere CTRL C.
☐ Spegnere la stampante.

IL SISTEMA NON FUNZIONA

☐ Rilanciare (CTRL C).
☐ Fermare il sistema e ripartire completamente.

UNITA' A DISCHI CONTINUAMENTE ATTIVA

☐ Non è caricato il dischetto. Inserirne uno.
☐ Togliere il dischetto e far ripartire il sistema.

COMPORTAMENTO CHIARAMENTE ANOMALO

☐ Sospettare di un errore di operatore. Riprovare. Verificare se il dischetto di sistema è corretto e se la configurazione della stampante è corretta.
☐ Sospettare che il dischetto di sistema sia danneggiato. Sostituirlo con un altro.
☐ Sospettare che il programma applicativo sia danneggiato. Sostituirlo con un altro.
☐ Spegnere tutto. Riprovare.
☐ Sospettare un guasto hardware.

APPENDICE O

REGOLE DI BASE PER LA LOCALIZZAZIONE DEI GUASTI

- In questo ordine:
  1. Sospettare un errore di operatore.
  2. Sospettare che i dischetti siano danneggiati.
  3. Sospettare del software.
  4. Sospettare dell'hardware.
- Annotare una descrizione dettagliata del malfunzionamento.
- Tentare di nuovo dall'inizio. Usare dischetti nuovi. Verificare tutta la configurazione meccanica e le connessioni.

IL LIBRO

Il sistema operativo CP/M è stato progettato per rendere semplice l'uso di un microcomputer. Questo libro vi renderà semplice l'uso del CP/M. (Le versioni esaminate del CP/M sono: il CP/M 1.4 - il CP/M 2.2 - il nuovo sistema operativo multiutente MP/M).
La maggior parte degli utenti di microcomputer dovrà, infatti, un giorno o l'altro, fare ricorso al CP/M, disponibile su quasi tutti i computer basati sul microprocessore 8080 o Z80, come pure su certi sistemi utilizzanti il 6502.
Il libro, senza presupporre alcuna conoscenza di un calcolatore, inizia con la descrizione, passo-passo, delle procedure di inizializzazione del sistema: accensione del computer, inserimento dei dischetti, esecuzione delle più comuni operazioni su file, compresa la duplicazione dei dischetti. Prosegue con il PIP (programma di trasferimento dei file), il DDT (programma di messa a punto - debugger) e ED (programma editor). Per entrare sempre più fornendo numerosi consigli pratici all'interno del CP/M e delle sue operazioni, al fine di comprenderne appieno le risorse ed eventualmente dare gli strumenti per successive modifiche.

L. 22.000

Cod. 510 P

ISBN 88-7056-103-8

GRUPPO
EDITORIALE
JACKSON

Rodnay
Zaks

CP/M CON MP/M™

510 P