Università degli Studi di Pisa

SCRIPT

Dr. RICCARDO MEDVES

30

CNUCE

Copyright C ottobre 1972
by CNUCE, Centro Nazionale Universitario di Calcolo Elettronico
dell'Università di Pisa.

S C R I P T

a cura di
Riccardo Medves

CNUCE
14/5/1973

INDICE

Questo manuale e' stato redatto in base agli appunti utilizzati per vari corsi di SCRIPT effettuati al CNUCE e si puo' logicamente suddividere in tre parti.

Una prima parte (Cap. 1-2-3-4) descrive molto schematicamente la struttura di un calcolatore; tali concetti non sono evidentemente esaurienti, in quanto presupporrebbero la presenza di un istruttore e di un dialogo docente-discente per il chiarimento di molti aspetti del problema, ma servono semplicemente a fornire il significato di alcune parole che ricorrono spesso nella terminologia dei calcolatori.

Una seconda parte (Cap. 5-6-7) concerne la creazione e l'aggiornamento di files CMS. Tale parte e' rivolta a chi non ha mai usato un terminale e quindi non sa come comportarsi di fronte ad esso. Chi ha gia' lavorato in CMS (anche se non in SCRIPT), potra' saltare le prime due parti, mentre per chi non ha mai lavorato in CMS ritengo essenziale in questa fase l'uso pratico di un terminale in parallelo alla lettura del manuale.

Nella terza parte infine (Cap. 8-9) si descrivono i comandi SCRIPT propriamente detti e il loro uso.

1) Possibilita' e caratteristiche del linguaggio di comandi SCRIPT.

Per prima cosa cerchiamo di capire a che cosa serve lo SCRIPT.
Supponiamo di dover battere con una macchina da scrivere un testo qualsiasi: ogni tasto premuto provoca sul foglio di carta l'immediata stampa di un carattere, giusto o sbagliato che sia.
Quando saremo arrivati in fondo al testo da battere, oltre al tempo perso per correggere gli errori di battitura man mano che si verificavano, dovremo controllare di non aver saltato parole o righe intere. Guai a dover inserire una frase o a dover fare una correzione che comporti un accorciamento o un allungamento del testo per piu' di qualche parola. Per non parlare poi della fatica che si deve fare per rispettare la struttura tipografica prestabilita: non si sa mai quando andare a capo, se lasciare la riga che stiamo scrivendo piu' corta delle altre o se inserire un'altra parola, e cosi' via.
Lo SCRIPT, usato in connessione con un calcolatore, fa si' che del testo battuto vengano fatte due copie: una direttamente sul foglio di carta del terminale a cui si lavora, e questa corrisponde alla stampa su carta che si ottiene con una macchina da scrivere normale, l'altra registrata su un disco magnetico del calcolatore.
Da questa copia del nostro testo registrata su disco possiamo poi ottenere tante stampe su carta quante vogliamo.
Se poi la copia su disco non ci soddisfa possiamo correggerla, allungarla o accorciarla in modo semplice, ottenendo cosi' stampe corrette: pensera' il calcolatore a sostituire o aggiungere le frasi che noi gli indichiamo, variando automaticamente la lunghezza dell'intero testo.
Non solo, ma inserendo dei comandi particolari si possono ottenere dei vantaggi tipografici non indifferenti: quello dell'allineamento automatico delle righe sulla destra, per esempio.
Le righe battute a terminale e inviate su disco possono essere di lunghezza varia, ma le righe che compaiono in stampa hanno tutte lunghezza costante: cio' e' ottenuto dallo SCRIPT in fase di stampa spostando le ultime parole a destra di una riga troppo lunga nella riga sottostante, o completando una riga troppo corta con le prime parole a sinistra della riga sottostante.
Inoltre in fase di stampa si possono far eseguire dei salti pagina, numerazioni progressive delle pagine, centraggio di parole nel mezzo delle righe o della pagina, spostamenti rispetto al margine sinistro di stampa di interi capoversi, inserimento automatico di linee bianche, ecc.
L'insieme dei comandi per ottenere tutto cio' costituisce lo SCRIPT.

2) Struttura generale di un calcolatore.

unita' di ingresso/uscita: permettono di colloquiare col calcolatore, di inserire i dati che formano il nostro programma, di ottenere la stampa dei risultati desiderati.

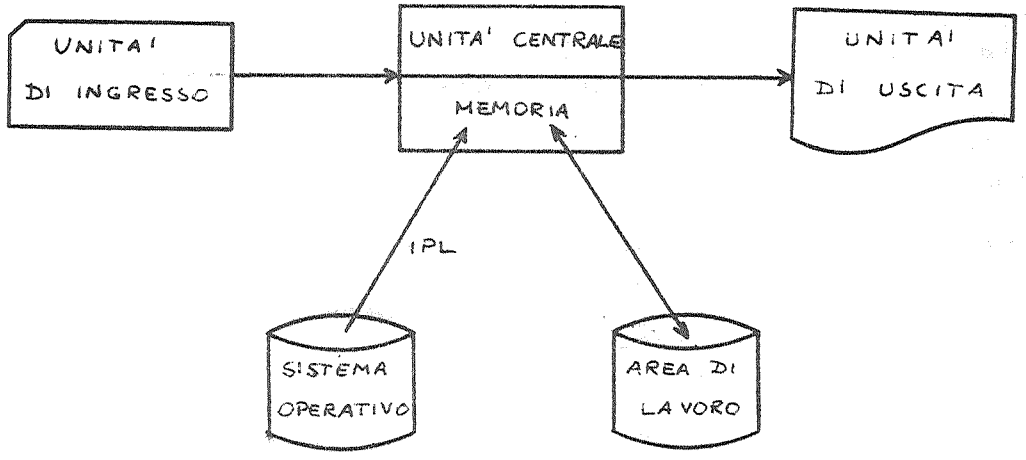
memoria: magazzino in cui vengono tenuti i dati che il calcolatore deve elaborare.

unita' centrale: parte di controllo e gestione di tutte le attivita' del calcolatore (leggere, scrivere, eseguire calcoli, ecc).

programma: insieme di istruzioni scritte in un particolare tipo di linguaggio, che indicano al calcolatore cosa deve fare, quali tipi di calcoli deve eseguire ecc. (tipi di linguaggi: FORTRAN, ASSEMBLER, PL/1 ....).

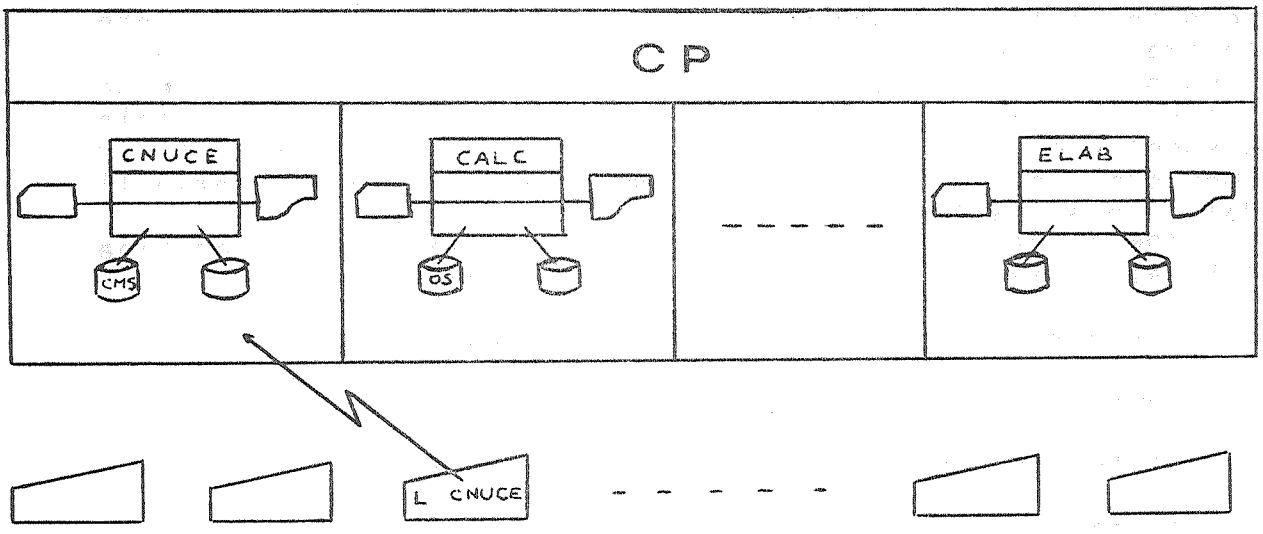
sistema operativo: e' un particolare tipo di programma, mantenuto su un determinato supporto esterno (disco, nastro...), che viene caricato in memoria all'inizio del lavoro e che gestisce tutti i programmi presentati degli utenti, fornendo loro le risorse richieste, passando il controllo dall'uno all'altro ecc.. (tipi di sistemi operativi: IBSYS, CMS, OS, APL...).

I.P.L.: (Initial Program Loading) e' la fase in cui il sistema operativo viene caricato dal supporto che lo contiene (disco, nastro, ecc.) in memoria.



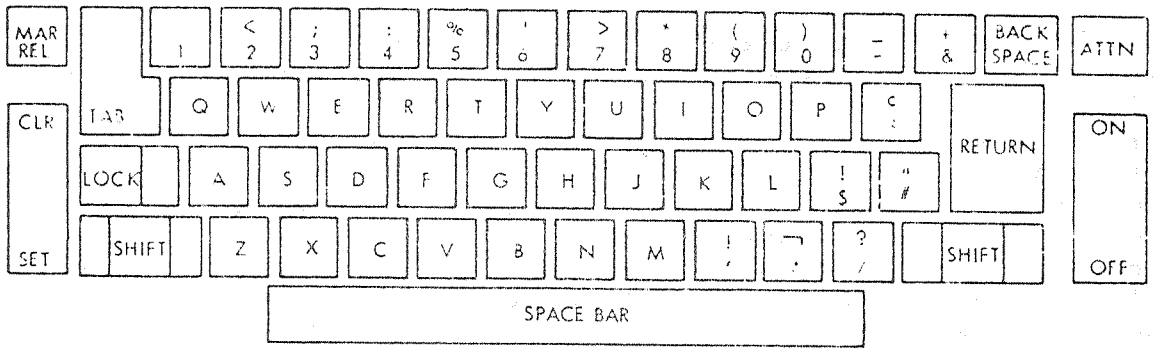
3) Struttura del calcolatore 360/67.

Il 360/67 e' un calcolatore un po' particolare: infatti invece di avere in memoria un unico sistema operativo che gestisce i lavori degli utenti, possiede un programma di controllo che si chiama CP (Control Program), il quale ha la caratteristica di poter simulare in memoria vari calcolatori, chiamati Macchine Virtuali e contraddistinti ciascuno da un certo nome.
Ciascuno di tali calcolatori ha simulata una sua struttura, cioe' una sua configurazione di unita' e puo' essere gestito da uno qualsiasi dei sistemi operativi sopra descritti (OS, CMS, APL, ecc)



4) Collegamento ad una macchina virtuale.

Il programmatore comunica al sistema cio' che vuole fare per mezzo di un terminale, costituito da una telescrivente collegata col calcolatore, la cui tastiera e' la seguente:



Da uno qualsiasi di tali terminali, il programmatore puo' collegarsi ad una macchina virtuale, cioe' ad un suo proprio calcolatore (identificato da un nome e da una parola d'ordine) e successivamente caricare il proprio sistema operativo: le elaborazioni che il programmatore fa eseguire sono del tutto <u>indipendenti</u> da quelle degli altri utenti che lavorano contemporaneamente sulle altre macchine virtuali. Questo tipo di accesso viene chiamato in linguaggio tecnico "time-sharing", in quanto il tempo di macchina reale disponibile viene suddiviso fra i programmatori presenti in modo che l'esecuzione dei programmi passi a brevi intervalli dall'uno all'altro.
Il programmatore che vuole collegarsi alla propria macchina virtuale, deve compiere le seguenti operazioni:

a) Accendere il terminale (tasto a destra in ON).

b) premere il tasto ATTN per mettersi in comunicazione con il CP.
Sul foglio comparira' una scritta del tipo:

```
CP-67 online   xd. 65 qsyosu
restart
```

c) Scrivere LOGIN (abbreviato L) seguito dal nome della sua macchina virtuale.
Premere il tasto RETURN per inviare il messaggio al calcolatore.
Sul foglio comparira' la scritta:

```
ENTER PASSWORD:
```

Questa ultima riga e' ottenuta dalla scrittura di tre linee di otto caratteri, l'una sull'altra, e serve a coprire la parola d'ordine in modo che nessun altro, non autorizzato, possa accedere a tale macchina virtuale.

d) Scrivere la parola chiave, che viene cosi' a sovrapporsi alla riga scritta dal terminale, e premere il tasto RETURN.
Sul foglio comparira' la scritta:

```
READY AT  ora  ON  data
CP
```

A questo punto il programmatore ha a disposizione un proprio calcolatore virtuale, sul quale carichera' il sistema operativo che gli permettera' di gestire i suoi programmi.
Il sistema operativo per poter lavorare in linguaggio SCRIPT e' il CMS.

e) Scrivere IPL CMS (abbreviato in I CMS) e premere il tasto RETURN.
Sul foglio comparira' la scritta:

```
CMS..VERSION 3.0 (data)
```

A questo punto il programmatore ha a disposizione un calcolatore virtuale proprio, su cui e' stato caricato il sistema operativo CMS.

f) Alla fine della sessione di lavoro, per interrompere il collegamento, premere di nuovo il tasto ATTN.
Il calcolatore rispondera' con:

```
CP
```

Scrivere allora LOGOUT (abbreviato LOG).
Il calcolatore rispondera' con alcune informazioni riguardanti il tempo di connessione e interrompera' il collegamento.

g) Spengere il terminale (tasto a destra in OFF).

Tutti i comandi che si danno a terminale possono essere indifferentemente scritti con lettere maiuscole o minuscole, benche' in questo manuale siano mostrati a lettere maiuscole per necessita' visiva.

La forma di tutti i comandi e' quella di un "codice operativo" (es. LOGIN, IPL ecc) seguito da uno o piu' "parametri" (es. nome della macchina virtuale, CMS ecc): codice operativo e parametri devono essere tra loro separati da uno o piu' spazi bianchi.

5) Struttura dei dati CMS su disco.

Un qualsiasi insieme di dati fornito al calcolatore e messo su un disco di lavoro si chiama "file".
Per distinguerlo dagli altri files che gia' sono su quel disco, ciascun file CMS e' identificato da tre etichette: un "nome", un "tipo" e un "modo".
In particolare per i file SCRIPT, si dovra' codificare:

nome: una qualsiasi sequenza di caratteri alfanumerici (da 1 a 8).
tipo: la parola SCRIPT.
modo: la parola P1 che indica il disco su cui si trova il file e che, nella maggior parte dei casi sara' omessa, lasciando al sistema operativo la ricerca.

Il CMS fornisce vari comandi che permettono di elaborare files memorizzati su disco: esistono per esempio comandi per la creazione, la correzione e la stampa di files.
Per ottenere l'esecuzione di un comando CMS si deve scrivere a terminale la parola chiave che indica il comando, seguita dal nome, tipo ed eventuale modo di quel file che si vuol trattare.

Comandi (abbreviazioni) ed esempi:

```
LISTF  CASA  SCRIPT
(L)
```

con tale comando si effettua la ricerca, su disco, di un file che ha nome CASA e tipo SCRIPT; se il file esiste, vengono stampati come risposta il nome, il tipo, il modo del file, la sua dimensione (in records di 829 bytes) e la data dell'ultimo aggiornamento.
Se il file non esiste viene data la risposta
FILE NOT FOUND.

```
LISTF  *  SCRIPT
(L)
```

si ottiene la ricerca di tutti i files che hanno tipo SCRIPT, qualunque sia il loro nome: per ciascuno di essi vengono stampate le informazioni come descritto nel precedente comando.

```
LISTF  *  *     oppure semplicemente     LISTF
(L)                                      (L)
```

si ottiene la ricerca di <u>tutti</u> i files esistenti su disco e la stampa per ciascuno di essi delle informazioni descritte in precedenza.

PRINTF CASA SCRIPT 3 9
(P)

si ottiene la stampa a terminale delle righe dalla 3 alla 9 del file indicato (nel nostro caso il file CASA SCRIPT).
Se i due numeri sono omessi, si ottiene la stampa a terminale di tutto il file, mentre se e' omesso solo il secondo si intende raggiungere la fine del file a partire dalla linea indicata.
Se, una volta inviato il comando di stampa, si desidera interrompere momentaneamente la scrittura del file, basta premere il tasto ATTN una volta e portarsi cosi' sotto CP.
Se si vuol riprendere la stampa, basta premere il tasto ATTN una seconda volta, seguito dal RETURN; altrimenti, se si vuol interrompere definitivamente la stampa, basta, dopo aver premuto il tasto la seconda volta, dare il comando KT prima di premere il tasto RETURN.

ERASE CASA SCRIPT

si ottiene la cancellazione dal disco di lavoro del file indicato, rendendo cosi' disponibile l'area su disco precedentemente occupata.

SPLIT nome1 SCRIPT nome2 SCRIPT 3 5

serve per creare un nuovo file (nome2 SCRIPT) prelevando cinque linee a partire dalla terza compresa del file nome1 SCRIPT. Se il secondo parametro e' omesso, si intende raggiungere la fine del file. Se il file nome2 esiste gia', la parte indicata viene semplicemente aggiunta alla fine.

COMBINE nome1 SCRIPT P1 nome2 SCRIPT P1 ... nomeN SCRIPT P1

serve per creare un nuovo file (nome1 SCRIPT) unendo insieme gli altri files che seguono (da nome2 a nomeN). Se il file nome1 esiste gia', viene cancellato prima di ricreare il nuovo file nome1.

ALTER nome1 SCRIPT P1 nome2 SCRIPT P1

serve per cambiare il nome "nome1" di un file esistente su disco in "nome2".

Nota: dopo l'invio di un comando al calcolatore (invio che si ottiene sempre premendo il tasto RETURN) si attenda che il comando sia eseguito ed accettato.
Il calcolatore rispondera' con
R; tempo di elaborazione
oppure
E(errore); tempo di elaborazione
a seconda che il comando sia stato eseguito correttamente o meno.

6) Creazione di un file SCRIPT.

Per ottenere la registrazione su disco di un file con un certo nome e tipo, si invia al calcolatore il comando

```
EDIT  nome tipo
(E)
```

e si preme il tasto RETURN. Esempio:

```
EDIT  CASA  SCRIPT
```

Sul foglio compare la scritta:

```
NEW FILE.
INPUT:
```

con la quale il calcolatore indica che il file e' nuovo, che non esisteva gia' su disco, e si predispone ad accettarlo in ingresso.
A questo punto l'utente puo' battere a terminale il testo del suo file.
Non ha importanza la forma con cui le frasi vengono scritte a terminale, in quanto, come gia' abbiamo accennato, una caratteristica dello SCRIPT e' quella di allineare a destra le frasi del file: e' pero' consigliabile tenere le righe abbastanza corte, affinche' durante la fase di correzione, la stampa a terminale di una riga, non porti via un tempo eccessivamente lungo.
Dopo ogni riga premere il tasto RETURN per inviarla al calcolatore.
Le righe che vengono in tal modo inviate al calcolatore, non vanno direttamente su disco, ma restano momentaneamente nella sua memoria.

Puo' accadere che per un improvviso guasto di macchina o mancanza di corrente o altro inconveniente il calcolatore si blocchi o, come si dice, il sistema operativo vada in "RESTART".
In tal caso tutto cio' che e' in memoria viene perduto, mentre cio' che e' su disco rimane inalterato.
Allora, per evitare di lavorare a vuoto, e' buona norma, di tanto in tanto, salvare su disco il lavoro fatto in modo che il file si conservi anche in caso di malfunzionamento del sistema.
In caso di restart non sara' piu' necessario riscrivere il file dall'inizio, ma solo quella porzione del file che e' stata battuta dopo l'ultimo salvataggio.

Per salvare su disco il file che si trova in memoria si procede nel modo seguente: dopo avere inviato una linea di testo al calcolatore premendo il tasto RETURN, si deve, senza battere alcun carattere sulla linea, premere ancora una volta il tasto RETURN.
Il calcolatore risponde con:

EDIT:

Si deve allora dare il comando per memorizzare temporaneamente il file su disco, e cioe'

SAVE

Il sistema risponde con:

INPUT:

permettendoci cosi' di continuare a scrivere il resto del file dal punto in cui avevamo provveduto al salvataggio.
Tale procedura, per i motivi su accennati, e' bene sia ripetuta di tanto in tanto.
Alla fine della sessione di lavoro vogliamo memorizzare definitivamente il file su disco, per poterlo ritrovare al successivo collegamento
Per questo, dopo l'ultima riga inviata premendo RETURN, si prema ancora il tasto RETURN.
Il calcolatore risponde con

EDIT:

Diamo il comando:

FILE

Il calcolatore memorizza definitivamente il file su disco e risponde con
R; tempo elaborazione

Nota1: durante la stesura del testo, cioe' in fase di INPUT, oppure mentre scriviamo un comando CMS, puo' accadere di battere un tasto sbagliato, o anche di sbagliare una parola o tutta una riga. Se ce ne accorgiamo subito, possiamo porvi rimedio facendo uso dei due caratteri di controllo @ e ¢, descritti in dettaglio in Appendice A.

@: permette di cancellare dalla riga corrente l'ultimo carattere scritto (se ripetuto n volte di seguito cancella gli ultimi n caratteri scritti).

¢: permette di cancellare tutta la riga corrente scritta fino a quel momento.

Nota2: oltre al testo che vogliamo scrivere e che viene trattato successivamente dallo SCRIPT possiamo far uso di altri comandi particolari che permettono di gestire il testo secondo determinate richieste, ad esempio saltare delle righe bianche, andare a pagina nuova, vietare lo spostamento delle parole da una riga all' altra ecc. Tali comandi vengono inseriti nel testo nei punti in cui essi devono agire: iniziano sempre con un punto a colonna 1, e vengono inviati al calcolatore premendo il tasto RETURN come per una qualsiasi riga normale. E' chiaro che qualsiasi linea inizi con un punto a colonna 1 viene interpretata come comando SCRIPT: nel testo non devono pertanto comparire linee normali che inizino con un punto in colonna 1. I comandi dello SCRIPT saranno descritti in dettaglio in seguito.

Esempio:

```
                                        <----- (ATTN)
cp-67 online     xd.65 qsyosu

l cnuce
ENTER PASSWORD:
********
READY AT 10.41.36 ON 04/11/73
CP
i cms
CMS..VERSION 3.0 (31 mag 72)

e casa script
NEW FILE.
INPUT:
Questa e' una prova di
creazione di un file SCRIPT.
.comando SCRIPT
                                        <----- (2 RETURN)
EDIT:
save
INPUT:
Questo fy@ile non
esisteva prima su disco.
Ogni volt¢Ogni riga del file
.comando SCRIPT
viene inviata al calcolatore
premendo il tost@@@asto RETURN.
                                        <----- (2 RETURN)
EDIT:
file
R; T=0.24/0.71 10.46.37

l casa script
FILENAME FILETYPE MODE NO.REC. DATE
CASA     SCRIPT    P1       1    4/11
R; T=0.02/0.03 10.46.47

p casa script 6 8

Questo file non
esisteva prima su disco.
Ogni riga del file

R; T=0.03/0.06 10.47.04

erase casa script
R; T=0.01/0.03 10.47.35

p casa script
FILE NOT FOUND.
E(00003); T=0.01/0.02 10.47.56
                                        <----- (ATTN)
CP
log
CONNECT= 00:06:41  VIRTCPU= 000:00.35  TOTCPU= 000:01.15
LOGOUT AT 10.48.09 ON 04/11/73
```

7) Aggiornamento di un file esistente.

Supponiamo di aver gia' scritto il file CASA SCRIPT, di averlo memorizzato su disco e di volerlo ora modificare, cioe' di voler aggiungere o togliere righe o di voler correggere errori commessi in fase di scrittura.
Per prima cosa e' necessario portare in memoria dal disco il file desiderato, mediante il comando:

```
EDIT  nome  tipo
(E)
```

nel nostro caso:

```
EDIT CASA SCRIPT
```

Sul foglio compare ora la scritta:

```
EDIT:
```

in quanto il file esiste gia' su disco, non deve essere creato di nuovo, ma solo corretto.
A questo punto il calcolatore e' pronto a ricevere qualsiasi comando EDIT per l'aggiornamento del file stesso.
Associato al file richiamato c'e' un indicatore di linea, che all'inizio e' posizionato in testa al file; per poter fare una modifica su una certa riga del file, si deve spostare l'indicatore in corrispondenza all'inizio di tale riga.

a) I comandi per eseguire spostamenti dell'indicatore di linea all'interno del file sono in forma estesa ed abbreviata i seguenti:

```
TOP
(T)
```

posiziona l'indicatore all'inizio del file, prima della prima riga in modo da permettere l'eventuale inserzione di altre righe in testa al file.

```
BOTTOM
(B)
```

posiziona l'indicatore sull'ultima riga del file.

```
UP n
(U)
```

posiziona l'indicatore all'n-ma riga precedente quella corrente. Se n e' omesso viene assunto uguale a 1 ed avviene il posizionamento sulla riga immediatamente precedente quella corrente.

```
UP /stringa/
(U)
```

ricerca la sequenza di caratteri "stringa" nelle righe

precedenti quella corrente, a partire da quella immediatamente precedente, fino all'inizio del file. La ricerca si arresta quando tale stringa viene trovata per la prima volta, e l'indicatore viene posizionato su tale riga.

NEXT n
(N)

posiziona l'indicatore sull'n-ma riga seguente quella corrente. Se n e' omesso, viene assunto uguale a 1 ed avviene il posizionamento sulla riga immediatamente seguente quella corrente.

NEXT /stringa/ oppure LOCATE /stringa/
(N) (L)

ricerca la sequenza di caratteri "stringa" nelle righe seguenti quella corrente a partire da quella immediatamente seguente fino alla fine del file.
La ricerca si arresta quando tale stringa viene trovata per la prima volta, e l'indicatore viene posizionato su tale riga.

FIND ␣␣----␣ stringa
(FI)

dove ␣␣----␣ e' una sequenza di n caratteri bianchi, ricerca nelle righe seguenti a quella corrente fino alla fine del file, la sequenza di caratteri "stringa" posizionata a partire dall'(n+1)-esimo carattere della riga. Cioe' ogni carattere non bianco della stringa viene confrontato col carattere che si trova nella colonna corrispondente della linea. La ricerca si arresta quando tale stringa viene trovata per la prima volta e l'indicatore viene posizionato alla riga trovata.

LINENO
(LI)

fornisce il numero della linea corrente a partire dall'inizio del file.

GOTO n
(G)

permette il trasferimento dalla linea corrente alla n-ma linea del file.

b) I comandi che servono ad elaborare le linee di un file sono i seguenti:

CHANGE /stringa1/stringa2/ n
(C)

sostituisce la sequenza di caratteri "stringa1" con "stringa2" nella linea corrente e nelle (n-1) linee successive.
Se n e' omesso e' assunto uguale a 1 ed il cambiamento interessa la sola linea corrente; se n e' posto uguale ad * si intende raggiungere la fine del file.
Se le linee interessate contengono piu' di una sequenza "stringa1" solo la prima di esse (da sinistra) viene cambiata. Le due stringhe possono essere di lunghezza diversa. Dopo l'esecuzione di tale comando si resta posizionati sull'ultima riga corretta.

```
Esempi :        AMARE
c /m/r/       : ARARE
c /r/lt/      : ALTARE
c /lta/tto/   : ATTORE
c /tto//      : ARE          nota 1
c /r/var/     : AVARE
c /va/lte/    : ALTERE   }
c /e/o/       : ALTORE   } nota 2
c /o/e/       : ALTERE   }
c /re/ro/     : ALTERO
c /alt/lo /   : IO ERO       nota 3
```

nota1: la stringa di lunghezza nulla (due barre consecutive) permette di eliminare una sequenza dalla riga corrente.
nota2: per effettuare un cambiamento occorre fornire delle stringhe di lunghezza sufficiente ad individuare univocamente il punto da modificare.
nota3: si noti che lo spazio bianco e' un carattere come un altro all'interno della stringa.

CHANGE /stringa1/stringa2/ n *
(C)

sostituisce tutte le sequenze di caratteri uguali a "stringa1" con "stringa2" nella linea corrente e nelle altre (n-1) successive. Se n e' posto uguale ad * si intende raggiungere la fine del file. (n non puo' essere omesso e deve essere esplicitamente scritto uguale a 1 nel caso si voglia modificare la sola linea corrente).

REPLACE linea
(R)

sostituisce la linea scritta a quella corrente. Dopo l'esecuzione l'indicatore resta posizionato sulla stessa riga.

DELETE n
(D)

cancella n linee a partire da quella corrente (compresa). Dopo l'esecuzione l'indicatore risulta posizionato alla prima riga non cancellata.
Se n e' omesso e' assunto uguale a 1; avviene allora la cancellazione della sola linea corrente e il posizionamento sulla successiva.

DELETE /stringa/
(D)

cancella le linee del file a partire da quella corrente fino a quella che contiene la sequenza "stringa" (esclusa).

OVERLAY ßß----ß stringa
(O)

ciascun carattere non bianco della stringa va a sostituire il carattere che si trova nella colonna corrispondente della linea corrente.
(ß indica uno spazio bianco)

BLANK ßß----ß stringa
(BL)

ciascun carattere non bianco della stringa fa sostituire uno spazio bianco al carattere che si trova sulla colonna corrispondente della linea corrente.
(ß indica uno spazio bianco)

INSERT linea
(I)

inserisce la linea scritta a lato dopo la linea corrente, prima della successiva e posiziona l'indicatore sulla linea inserita.

INPUT
(I)

non seguito da alcun carattere, fa si' che il calcolatore entri in fase di INPUT. Si possono allora battere quante righe si vogliono e tutto questo nuovo testo viene inserito fra la linea corrente e la successiva. In tale fase si possono usare tutte le possibilita' offerte e descritte nel capitolo 6. Quando si vuol tornare in fase di correzione basta premere due volte il tasto RETURN e attendere la risposta EDIT: per continuare normalmente. L'indicatore risulta posizionato sull'ultima riga inserita.

PUT n nome

crea un nuovo file identificato da "nome" e tipo SCRIPT costituito dalla riga sulla quale siamo posizionati e dalle (n-1) righe successive. Se il file "nome" esiste gia', le linee vengono aggiunte alla fine del file.

PUT /stringa/ nome

crea un nuovo file identificato da "nome" e tipo SCRIPT, costituito dalla riga sulla quale siamo posizionati e dalle seguenti fino a quella (esclusa) che contiene la sequenza di caratteri "stringa". Se la stringa non e' trovata il comando non ha effetto. Se il file "nome" esiste gia', le linee vengono aggiunte alla fine del file.

PUT n oppure PUT /stringa/

crea un file temporaneo, come descritto sopra, e inseribile in altra posizione del file corrente tramite il comando GET n1 n2.
Il file resta su disco fino all'esecuzione di un nuovo comando PUT dello stesso tipo.

GET nome tipo n1 n2

preleva le righe da n1 a n2 dal file definito con "nome" e "tipo", le inserisce dopo la linea corrente, prima della successiva e si posiziona sull'ultima riga inserita. Se n1 e n2 sono omessi, viene prelevato l'intero file.

GET n1 n2

preleva le righe da n1 a n2 del file temporaneo creato in precedenza col comando PUT n o PUT /stringa/. Se n1 e n2 sono omessi preleva tutto il file.

c) Comandi vari:

AGAIN n
(A)

permette la ripetizione n volte dell'ultimo comando effettuato, senza la necessita' di riscriverlo esplicitamente. Se n e' omesso e' assunto uguale ad 1.

PRINT n
(P)

stampa a terminale la riga corrente e le (n-1) successive. Se n e' omesso e' assunto uguale a 1, e si ha la stampa della sola riga corrente.

ZONE n1 n2
(Z)

indica che il comando che segue dovra' operare solo tra le colonne n1 e n2 (n1 minore di n2) delle linee costituenti il file (mentre per default qualsiasi comando inizia sempre dalla colonna 1 della linea e termina, per i files SCRIPT, con la colonna 132).

Nota1: nelle descrizioni precedenti, i due caratteri / che racchiudono la sequenza di caratteri cercata, possono essere due qualsiasi caratteri uguali che non compaiano nella stringa.

Nota2: i comandi precedenti provocano la scrittura, a terminale, dell'intera linea ricercata o modificata, permettendo cosi' di verificare se tutto si e' svolto regolarmente ed eventualmente di tornare a correggere. Si puo' sopprimere la scrittura di tale linea facendo seguire il comando dal carattere punto (.), senza spazi bianchi intermedi. Esempio:
N. /stringa/
verra' ricercata la sequenza di caratteri "stringa"; se trovata , la tastiera si sblocchera' in attesa di nuovi comandi, senza che la linea in cui compare la sequenza venga stampata a terminale.

Valgono, durante la fase di correzione di un file (EDIT) le stesse regole di salvataggio descritte nel capitolo 6.
L'unica differenza e' che durante la fase di correzione ci troviamo gia' in EDIT: pertanto sara' sufficente dare di tanto in tanto il comando SAVE senza prima aver bisogno di premere il tasto RETURN.
Il sistema memorizzera' su disco le modifiche fatte e rispondera' con INPUT:
Se si vogliono continuare le correzioni bastera' allora premere tasto RETURN per ritornare in fase di EDIT.
Alla fine e' sufficiente scrivere FILE perche' il file sia memorizzato permanemtemente su disco.

8) Comandi SCRIPT.

Tutti i comandi Script cominciano con un punto in colonna 1. L'operando, se c'e', e' separato dal comando tramite uno o piu' spazi bianchi.
Alla fine del comando premere il tasto RETURN.

a) Comandi che descrivono il formato della pagina di stampa:

.HM n (heading margin)

fissa la distanza della eventuale intestazione (vedi .HE) dall'inizio del foglio (default .HM 1).

.TM n (top margin)

fissa la distanza del testo scritto dall'inizio del foglio (default .TM 5).

.BM n (bottom margin)

fissa la distanza dell'ultima riga dalla fine della pagina (default .BM 3).

.LL m (line length)

fissa la lunghezza in caratteri di una linea (default .LL 60).

.PL n (page length)

fissa la lunghezza in linee di una pagina (default .PL 66).

Tali comandi hanno automaticamente i valori segnati come default, e pertanto non e' necessario inserirli esplicitamente in un testo, a meno che non se ne vogliano cambiare le specifiche.

.HE stringa (heading)

stampa la sequenza di caratteri "stringa" in testa ad ogni pagina, ad eccezione della prima.
Il comando resta in effetto finche' non compare un nuovo comando .HE stringa1, che si sostituisce al primo.
Poiche' l'intestazione viene scritta in alto a sinistra nella pagina, sulla stessa linea della numerazione, la sequenza deve essere 10 caratteri piu' corta del valore definito col comando .LL .

b) Comandi che descrivono il formato del testo :

.PN ON (page numbering)

produce la numerazione di tutte le pagine eccetto la prima (default).

.PN OFF

sopprime la stampa della numerazione, ma continua la numerazione interna delle pagine.
Se successivamente si riprende la stampa della numerazione col comando .PN ON, tale numerazione comincera' dal numero di sequenza interno corrente.
Se si vuol alterare la numerazione sia interna che esterna delle pagine si deve usare il comando .PA n (vedi nel seguito).

.PN OFFNO

sopprime qualsiasi numerazione.

.PA (page)

produce un salto pagina, cioe' la linea che segue il comando e' stampata all' inizio di una nuova pagina.

.PA n

produce un salto a pagina nuova, come il comando precedente, rinumerando le pagine successive a partire dal numero n indicato.
Per esempio puo' accadere che un testo contenga delle pagine di introduzione e si voglia pertanto cominciare la numerazione delle pagine con la prima pagina del documento effettivo: si devono allora dare il comando .PN OFF all'inizio del file e i due comandi .PN ON e .PA 1 dopo aver scritto tutte le pagine di introduzione. Si otterra' in tal modo che la prima pagina del testo sia numerata col numero 1.

```
.PN OFF
---pagine di introduzione ---
.PN ON
.PA 1
--- pagine del testo ---
```

.CP n (conditional page)

causa un salto pagina se mancano meno di n linee alla fine della pagina.
Tale comando e' ad esempio utile prima di una intestazione di capitolo, per evitare che tale intestazione venga stampata alla fine della pagina precedente.

.SP n (space)

causa l'inserimento di n linee bianche nel punto in cui si trova il comando.

.DS (double space)

fa stampare il testo in doppia spaziatura e raddoppia l'effetto di successivi comandi .SP n .

.SS (single space)

annulla l'effetto del comando precedente.

.CF (center)

fa si' che le informazioni sulla riga di stampa che segue vengano centrate in mezzo alla linea, invece che allineate a sinistra.

Abbiamo gia' accennato come una delle caratteristiche dello Script sia quella di allineare a destra le righe, spostando opportunamente le parole del testo da una linea all'altra.
Cio' puo' a volte non essere desiderato come ad esempio nella costruzione di tabelle o quando si inizia un nuovo capoverso.
Vi sono vari metodi e comandi per ottenere un fatto del genere: in pratica tutti i comandi Script eccetto .PN e .CP causano una rottura di continuita' tra le due righe alle quali si riferiscono.
Un altro metodo per tenere separate due linee e' quello di iniziare la seconda di esse con almeno uno spazio bianco.
Vi sono inoltre due specifici comandi che permettono di fare quanto sopra descritto:

.BR (breaking)

mantiene separate la linea che precede da quella che segue tale comando, non spostando parole dall'una all'altra.

.NF (no format)

tutte le linee che seguono tale comando non vengono gestite dallo Script. Non si ha cioe' allineamento automatico a sinistra e destra ma la stampa ricalca esattamente quello che viene scritto nella fase di battitura del testo, senza alcuna modifica.

.FO (format)

annulla l'effetto del comando precedente.

.IN n (indent)

fa si' che tutte le linee seguenti, prima compresa, inizino dopo n spazi bianchi dal margine sinistro. Ha validita' finche' non viene incontrato un nuovo comando .IN n o il comando di chiusura .IN .

.IN

annulla l'effetto del comando precedente.

.UN m (undent)

fa si' che la sola riga seguente inizi m colonne indietro ripetto al valore n definito da un precedente comando .IN n
Esempi :

```
.NF                    -ABCDE
.IN 1                   FGHIJ
.UN 1                   KLMNO
-ABCDE          ->     -PQRST
FGHIJ                   UVWXY
KLMNO
.UN 1
-PQRST
UVWXY
.IN
.FO
```

Nota che il valore di m non puo' superare il valore di n dato precedentemente.

.OF n (offset)

fa si' che tutte le righe seguenti, <u>eccetto la prima</u> inizino dopo n spazi bianchi dal margine sinistro. Esempio:

```
.NF                    -ABCDE
.OF 1                   FGHIJ
-ABCDE                  KLMNO
FGHIJ           ->     -PQRST
KLMNO                   UVWXY
.OF 1
-PQRST
UVWXY
.OF
.FO
```

Ha validita' finche' non si trova un nuovo comando .OF n oppure il comando di chiusura .OF
Si noti che se e' in azione un comando .IN m, il comando .OF n parte dalla colonna m+1.

.OF

annulla l'effetto del comando precedente.

Puo' a volte accadere che, a causa dell'allineamento a destra delle righe, l'inserimento di spazi bianchi nella linea influisca negativamente sul formato di stampa.
Ad esempio, in una lista di bibliografie puo' accadere che sia allineata solo la prima parola della frase, (in questo caso il numero di ordine) mentre sarebbe richiesto l'allineamento anche di altre parole, quali l'autore, ecc.

```
1. Bianchi, A ....
2.  Rossi, B ....
```

Si fa allora uso del tasto di tabulazione TAB CMS e del comando Script .TB per allineare le parti volute della frase.

```
.TB n1 n2 ... nM
```

definisce la spaziatura logica all'interno di una linea, inserendo certe posizioni logiche di stop (n1, n2 ...) che vengono prese in considerazione con l'uso del tasto TAB (valori di default : 5,10,15,.....75).
Esempio:

```
.TB 5 8
.OF 8
(TAB) 1. (TAB) Bianchi, A ...
.OF 9
(TAB) 2. (TAB) Rossi, B ....
```

Si noti che se e' in vigore un precedente comando .IN m, tutti i valori della tabulazione vengono spostati a destra di m spazi.

c) Altri comandi che servono ad unire insieme piu' file di tipo SCRIPT durante la stampa:

.IM nome (imbed)

serve ad inserire nel posto desiderato un altro file "nome" SCRIPT che si trova sul disco di lavoro della propria macchina virtuale. Al momento in cui il comando .IM viene incontrato, in fase di stampa del file corrente, il controllo passa al file "nome" SCRIPT, che viene stampato di seguito al primo; Alla fine il controllo ritorna al file corrente e la stampa continua dalla riga successiva al comando.

.AP nome (append)

ha lo stesso scopo del comando .IM, solo che si applica soltanto per aggiungere files alla fine di quello corrente, non in mezzo alle istruzioni. Al momento in cui il comando viene incontrato in fase di stampa del file corrente, il controllo passa al file "nome" SCRIPT che viene stampato di seguito al primo; alla fine la stampa si interrompe senza ritornare al file corrente.

.RD n (read)

causa l'interruzione della stampa, e permette l'inserimento da terminale di n linee (se n e'omesso e' assunto uguale ad 1), dopo di che il processo di stampa riprende regolarmente.

9) Stampa di un file SCRIPT.

Abbiamo visto che un file Script si compone delle normali linee di stampa del testo e di particolari comandi (.XX) che servono a gestire in modo particolare la stampa a terminale del file.
Vi sono due modi di ottenere la stampa di un file Script.

stampa senza formattamento : si ha la stampa del file cosi' come e' stato battuto a terminale, compresi tutti i comandi Script; e' cioe' una lista pura e semplice di quanto battuto prima a terminale ed equivale al comando PRINTF del CMS.

stampa con formattamento : i comandi Script non vengono listati esplicitamente, ma vengono interpretati per ottenere il formattamento della stampa.

Una volta che abbiamo memorizzato il nostro file su disco, il comando di stampa sotto CMS e' il seguente:

```
SCRIPT  nome  opzioni
```

nome: e' il nome del file che si vuole stampare.
opzioni: un insieme di parole chiave, separate tra loro da uno o piu' spazi bianchi. Esse possono essere scelte tra le seguenti:

```
NO              (nowait)
```

fa partire immediatamente la stampa del file su terminale.
Se l'opzione NO e' omessa, appare invece il messaggio:
LOAD PAPER; HIT RETURN.
e l'esecuzione si interrompe fino a che non viene premuto il tasto RETURN, permettendo eventualmente di aggiustare la carta.

```
ST              (stop)
```

interrompe la stampa del file su terminale alla fine di ogni pagina, permettendo ad esempio di cambiare carta. Per far ripartire la stampa con la pagina successiva, basta premere il tasto RETURN.
Se l'opzione ST e' omessa, non c'e' pausa fino a quando tutto il file non sia stato stampato.

PAGEnnn (page)

fa stampare il file a partire dalla pagina nnn.
Il numero nnn deve comparire esplicitamente con gli zeri non significativi.
Tale opzione e' attiva anche con la specifica .PN OFF, mentre non e' attiva con la specifica .PN OFFNO.
Se l'opzione e' omessa, la stampa comincia dall'inizio del file.

SI (single)

fa stampare una sola pagina selezionata eventualmente da una opzione PAGEnnn.
Se l'opzione SI e' omessa, la stampa prosegue fino alla fine del file.

UN (unformatted)

fa stampare il testo non formattato dallo SCRIPT.
Se l'opzione UN e' omessa, il file viene formattato in fase di stampa.

TR (translate)

esegue una conversione di tutti i caratteri da minuscoli a maiuscoli, prima di inviare il file in stampa.
Se l'opzione TR e' omessa, viene eseguita la stampa cosi' come il file e' stato creato.

OF (offline)

la stampa viene effettuata sulla stampante centrale del calcolatore, invece che sul proprio terminale.
Se tale stampante non e' equipaggiata con una particolare catena, comprendente ad esempio i caratteri minuscoli, si deve anche effettuare una conversione con la opzione TR.
Se l'opzione TR e' omessa, la stampa avviene sul proprio terminale.

CE (center)

la stampa sulla stampante centrale viene centrata sulla carta.
Se l'opzione CE e' omessa, la stampa e' allineata a sinistra. Esempi:

a) SCRIPT CASA

Il calcolatore risponde a terminale:
LOAD PAPER; HIT RETURN.
e si ferma, dando cosi' la possibilita' di cambiare carta o di aggiustarla. Per iniziare si preme il tasto RETURN. Il calcolatore stampa il file CASA SCRIPT a terminale, con le opportune specifiche di formato, salti pagina ecc.

b) SCRIPT CASA NO

si ha la stampa come nell'esempio a), ma il calcolatore non richiede la sostituzione della carta (NO): appena premuto il tasto RETURN la stampa inizia immediatamente.

c) SCRIPT CASA NO UN

si ha a terminale la stampa non formattata del file CASA SCRIPT, cioe' una lista pura e semplice del file come e' stato precedentemente battuto a terminale.

d) SCRIPT CASA NO PAGE019 SI

si ottiene la sola stampa, formattata, della pagina 19 del testo.

e) SCRIPT CASA NO PAGE019 ST

si ottiene la stampa del file a partire da pagina 19 (PAGE019). Alla fine di ogni pagina il calcolatore si ferma (ST) e, per continuare, si deve premere il tasto RETURN.

f) SCRIPT CASA TR OF

si ottiene la stampa formattata in caratteri maiuscoli (TR) del file sulla stampante centrale del calcolatore (OF).

Se, una volta inviato il comando di stampa, si desidera interrompere la scrittura del file, basta premere il tasto ATTN una volta e portarsi cosi' sotto CP.
Se si vuol riprendere la stampa, basta premere il tasto ATTN una seconda volta, seguito dal RETURN; se invece si vuol interrompere la stampa, basta, dopo aver premuto il tasto la seconda volta, scrivere KT prima di premere il tasto RETURN.

APPENDICE A

Caratteri con significato particolare.

- Cancellazione di un carattere : @
  Il segno @ scritto subito dopo un carattere di una linea, cancella tale carattere. Si usa generalmente per cancellare un carattere errato in fase di battitura. Se il segno @ e' ripetuto n volte di seguito, cancella gli ultimi n caratteri scritti. Logicamente e' come se tale comando ci riportasse indietro nella linea e ci permettesse di sovrapporre ai vecchi i nuovi caratteri battuti. Si tenga presente che lo spazio bianco (tasto SPACE) e' considerato dallo SCRIPT un "carattere" al pari degli altri: se si vogliono percio' eliminare degli spazi bianchi sovrabbondanti, si deve sempre fare uso del carattere @ e mai del tasto BACKSPACE, che come vedremo serve per altri scopi. Esempi:

```
SV@CRIPT          =  SCRIPT    (cancella la lettera V)
SVRI@@@CRIPT      =  SCRIPT    (cancella le lettere VRI)
SC  @@RIPT        =  SCRIPT    (cancella i due bianchi)
```

- Cancellazione di una linea : ¢
  Il carattere ¢ permette di cancellare tutta la linea corrente scritta (prima di premere il tasto RETURN) e di riscrivere di seguito quella corretta. Esempio:

```
SVRI ¢SCRIPT      =  SCRIPT
```

  logicamente e' come se tale comando ci riportasse all'inizio della linea, cancellando quanto precedentemente scritto.

- Backspace logico : %
permette di tornare indietro logicamente in una linea di tanti caratteri quanti sono i segni % scritti. A questo punto i nuovi caratteri non bianchi battuti di seguito vanno a sovrapporsi ai vecchi, mentre gli spazi bianchi fanno restare invariato il vecchio carattere della linea. Esempi:

```
ABCD%%H     =  ABHD
ABCD%%HP    =  ABHP
ABCD%%H P   =  ABHDP
```

Si noti la differenza col carattere di cancellazione @ che, quando ripetuto n volte, perde completamente il ricordo degli n caratteri precedenti, anche se successivamente vengono battuti caratteri bianchi. Esempi:

```
ABCD@@H     =  ABH
ABCD@@H P   =  ABH P
```

- Tasto di backspace fisico : BACKSPACE
durante la codifica di un testo in SCRIPT puo' essere utile sovrapporre due o piu' caratteri per ottenere dei segni particolari (ad esempio il segno ∅ indicante lo spazio bianco e' ottenuto dalla sovrapposizione dei due caratteri b e /) o, come nella maggior parte dei casi, per ottenere delle parole sottolineate. Ci si serve in tal caso del tasto BACKSPACE, il cui funzionamento e' il seguente: una volta battuta la parola da sottolineare (ad esempio) si prema il tasto BACKSPACE tante volte quanto necessario per riportare fisicamente la pallina di stampa sotto la prima lettera della parola e si batta il carattere di sottolineatura "_" tante volte quante necessarie; poi si riprenda la codifica normale.
C'e' solo da notare una cosa, e cioe' che mentre i caratteri @, ¢ e % non vengono inviati al calcolatore, in quanto agiscono subito sulla linea corrente, ed e' la linea gia' corretta che viene inviata al calcolatore (RETURN), il tasto BACKSPACE causa invece l'inserimento nella linea di certi caratteri particolari, che verranno poi interpretati in fase di stampa dallo SCRIPT. Esempio:

La parola SCRIPT battuta a terminale come:
SCRIPT(BACK)(BACK)(BACK)(BACK)(BACK)(BACK)_____
viene inviata al calcolatore come:
S(BS)_C(BS)_R(BS)_I(BS)_P(BS)_T(BS)_
dove con (BACK) abbiamo indicato l'uso del tasto BACKSPACE e con (BS) abbiamo indicato quel particolare carattere che lo SCRIPT interpretera' in seguito per ricreare la sottolineatura.
Pertanto e' da tener presente che le parole sottolineate occupano nel file il triplo della loro lunghezza.

- Carattere di tabulazione logica : >
  permette di spostarsi logicamente ad una determinata colonna della linea a seconda delle specifiche standard del CMS e del tipo di file trattato. Se ad esempio si prevedono le colonne 1 5 10 come colonne di tabulazione, la codifica
  >>linea
  viene interpretata come una linea che inizia a colonna 10.
- Tasto di tabulazione fisica : TAB
  agisce come il carattere di tabulazione logica.

- Carattere di fine linea logica : #
  permette di scrivere su una linea fisica del terminale piu' linee logiche contenenti comandi CMS, separate tra loro dal segno #. Esempio:

  LISTF CASA SCRIPT # PRINTF CASA SCRIPT (RETURN)
  equivale a

  LISTF CASA SCRIPT (RETURN)
  PRINTF CASA SCRIPT (RETURN)

Per poter listare i caratteri particolari in un contesto SCRIPT senza cioe' che essi assumano quei particolari significati, si possono usare diversi comandi.
Il primo di essi si deve usare in fase di EDIT:

SETCHAR carattere numero

permette di introdurre in un file dei caratteri che non esistono sulla tastiera del terminale e in particolare di introdurre in un file i caratteri speciali descritti in precedenza:

numero: e' la configurazione decimale del carattere che si vuol inserire nel file.
carattere: e' il carattere che compare nella tastiera del terminale e che viene logicamente associato alla configurazione descritta da "numero".
Esempi:

SETCHAR & 2 : permette premendo il tasto & di inserire in un file la configurazione decimale 2 (esadecimale 2) che corrisponde alle perforazioni 12-2-9 su scheda, di cui non esiste riscontro sulla tastiera.

SETCHAR & 123 : permette, premendo il tasto & di inserire su un file la configurazione decimale 123 (esadecimale 7B) che corrisponde al carattere # , che altrimenti sarebbe interpretato come carattere di controllo fine di linea logica e non inserito nel file.

Ogni qual volta si preme cioe' il tasto "carattere", viene inserito nel file non quello, ma il carattere la cui configurazione decimale e' "numero".

Diamo qui la configurazione decimale dei 5 caratteri particolari descritti in precedenza:

@ = 124
¢ = 74
% = 108
> = 110
= 123

Le definizioni fornite mediante il comando SETCHAR rimangono valide finche' non si esce dall'ambiente EDIT.

Gli altri comandi devono essere usati in CMS:

CHARDEF C carattere1

sostituisce il carattere @ col carattere1 fornito, per la funzione di cancellazione carattere.

CHARDEF L carattere2

sostituisce il carattere ¢ col carattere2 fornito, per la funzione di cancellazione linea.

CHARDEF B carattere3

sostituisce il carattere % col carattere3 fornito, per la funzione di backspace logico.

CHARDEF T carattere4

sostituisce il carattere > col carattere4 fornito, per la funzione di tabulazione logica.

LINEND carattere5

sostituisce il carattere5 # col carattere fornito, per la funzione di fine linea logica.

Tali definizioni rimangono valide finche':

1) Si rifa' IPL del sistema.
2) Si emette un altro comando CHARDEF.

APPENDICE B

- Riassunto dei comandi.
(Sotto i titoli sono indicate le pagine alle quali sono descritti tali comandi).

Comandi CP (4-5)

- LOGIN nome (L)
- IPL CMS (I)
- LOGOUT (LOG)

Comandi CMS (6-7-8-9) (31)

- LISTF { nome | * } { tipo | * } (L)
- PRINTF nome tipo { n1 n2 | Ø } (P)
- ERASE nome tipo
- SPLIT nome1 tipo1 nome2 tipo2 n m
- COMBINE nome1 tipo1 P1 nome2 tipo2 P1 ... nomeN tipoN P1
- ALTER nome1 tipo1 P1 nome2 tipo2 P1
- EDIT nome tipo
  - INPUT → definizione del file; comandi SCRIPT (vedi seguito)
  - (2 RET) ↕
  - EDIT → SAVE; FILE; comandi EDIT (vedi seguito)
- CHARDEF funzione carattere
- LINEND carattere

caratteri particolari (27-28-29)

- @ = cancellazione carattere
- ¢ = cancellazione linea
- % = backspace logico
- > = tabulazione logica
- # = fine linea logica

- Comandi EDIT:

spostamenti (12-13)

TOP		
(T)		
BOTTOM		
(B)		
UP	{ /stringa/ n }	
(U)		
NEXT	{ /stringa/ n }	o LOCATE /stringa/
(N)		(L)
FIND	␣␣---␣ stringa	
(FI)		
LINENO		
(LI)		
GOTO	n	
(G)		

elaborazioni (14-15-16)

CHANGE	/stringa1/stringa2/	{ n * } { ␣ * }
(C)		
REPLACE	linea	
(R)		
DELETE	{ /stringa/ n }	
(D)		
OVERLAY	␣␣---␣ stringa	
(O)		
BLANK	␣␣---␣ stringa	
(BL)		
INSERT	linea	
(I)		
INPUT		
(I)		
PUT	{ /stringa/ n }	{ nome ␣ }
GET	{ nome tipo ␣ }	{ n1 n2 ␣ }

varie (17)-(30)

AGAIN	n
(A)	
PRINT	n
(P)	
ZONE	n1 n2
(Z)	
SETCHAR	carattere numero

- Comandi SCRIPT:

formattamento della pagina (18)	.HM n .TM n .BM n .LL m .PL n .HE stringa
formattamento del testo (19-20-21-22)	.PN ON .PN OFF .PN OFFNO .PA .PA n .CP n .SP n .DS .SS .CE .BR .NF .FO .IN n .IN .UN m .OF n .OF .TB n1 n2 ... nM
unione di files (23)	.IM nome .AP nome .RD n
stampa (24-25)	SCRIPT nome opzioni opzioni: NO ST PAGEnnn SI UN TP OF CE