DOS

Indice

1 Il Sistema Operativo **4**

1.1 Il Kernel del DOS 4

1.2 I Device Drivers del DOS 5

1.3 La Shell del DOS 5

1.4 Devices Speciali 6

2 Comandi del DOS: Generalità **7**

2.1 Nomi dei Files 8

2.2 Batch Files 9

2.3 Ridirezione Input e Output 10

3 Comandi del DOS: Comandi Relativi alle Directories **11**

3.1 Dir 11

3.1.1 Opzioni 11

3.1.2 Esempi 11

3.2 Cd 12

3.2.1 Note 13

3.2.2 Esempi 13

3.3 Md 13
3.3.1 Note 13
3.3.2 Esempi 13
3.4 Rd 14
3.4.1 Note 14
3.4.2 Esempi 14

4 Comandi del DOS: Comandi Relativi ai Files 14

4.1 Copy 14
4.1.1 Opzioni 15
4.1.2 Esempi 15
4.2 Del 17
4.2.1 Note 17
4.2.2 Opzioni 17
4.2.3 Esempi 17
4.3 Ren 18
4.3.1 Note 18
4.3.2 Esempi 18
4.4 Type 18
4.4.1 Note 19
4.4.2 Esempi 19

5 Comandi del DOS: Altri Comandi 19

5.1 Set 19
5.1.1 Esempi 19
5.2 Echo 19
5.2.1 Note 20
5.2.2 Esempi 20

5.3 For 20
5.3.1 Note 21
5.3.2 Esempi 21
5.4 If 21
5.4.1 Note 21
5.4.2 Esempi 21
5.5 ctty 22

6 Comandi del DOS: Comandi dei Batch Files 22

6.1 Pause 22
6.1.1 Esempi 23
6.2 Goto 23
6.2.1 Note 23
6.3 Esempi 23
6.4 Call 23
6.4.1 Note 24
6.4.2 Esempi 24

1 Il Sistema Operativo

La sigla DOS sta per Disk Operating System, ovvero sistema operativo basato sui dischi. Ci occupiamo in questa sede dell'MS-DOS, ovvero del DOS prodotto dalla Microsoft.

Un **sistema operativo** è un insieme di programmi e di procedure in linguaggio macchina che si occupano di caricare in memoria gli applicativi dell'utente, di eseguirli, di effettuare gli accessi alle periferiche (**devices**) del computer. Si tratta perciò di un'interfaccia fra il computer e l'utente.

Sono tre le principali componenti di un sistema operativo:

1. Il **kernel** ("nocciolo).
2. I **device drivers** ("gestori di periferiche).
3. La **shell**, o **command interpreter** ("interprete dei comandi).

Il kernel organizza i processi, carica i programmi in memoria, si occupa di eseguirli[1] e predispone delle "difese nei confronti di programmi instabili o errati. Per caricare i programmi in memoria il kernel chiede al device driver preposto alla gestione del disco dove risiede il programma di leggere il disco e di passargli i dati. La shell è il programma che accetta i comandi dalla tastiera, li analizza e poi passa le informazioni al kernel. Possiamo rappresentare shell, kernel e device drivers come tre strati di astrazione fra utente e computer.

utente
⋮
shell
kernel
device drivers
⋮
computer

1.1 Il Kernel del DOS

Il kernel dell'MS-DOS è molto semplice, in quanto può solo eseguire un programma alla volta e non ha nessun sistema di protezione nei confronti dei

[1] In taluni casi il kernel esegue più programmi contemporaneamente, nel qual caso si parla di sistema operativo *multi-tasking*. L'MS-DOS non ha questa proprietà.

programmi che esegue[2]. Originariamente poteva indirizzare fino ad un massimo di 640 kb RAM; qualche anno fa è stato aggiunto un device driver per accedere alla cosiddetta "memoria alta, ovvero la quantità di memoria al di là dei 640 kb. Fisicamente il kernel dell'MS-DOS si trova nei files nascosti `c:\msdos.sys` e `c:\io.sys`.

1.2 I Device Drivers del DOS

La situazione del DOS per quanto concerne la gestione dei floppy e dei dischi fissi attaccati alla scheda IDE[3] è poco usuale, infatti viene affidata direttamente al kernel. Il DOS dispone tuttavia di parecchi device drivers. In genere sono dei files binari avente l'estensione `.sys` o `.exe`, come per esempio

`ramdrive.sys`	per usare la memoria RAM come se fosse un disco
`mouse.sys`	per la gestione del mouse
`smartdrv.exe`	per dedicare parte della RAM a cache[4] di un disco
`himem.sys`	per accedere alla memoria RAM oltre i 640 kb di base
`interlnk.exe`	per collegare due computer con cavo diretto

A parte quelli appena menzionati ed altri che vengono distribuiti direttamente con l'MS-DOS, in genere i device drivers vengono scritti dai fabbricanti di hardware.

1.3 La Shell del DOS

La shell, o interprete dei comandi, che viene fornita con l'MS-DOS è contenuta nel file eseguibile `c:\command.com`. La shell presenta un **prompt** all'utente e attende un comando. Solitamente il prompt indica quali sono il drive e la directory correnti, per esempio[5]

`c:\user>`■

indica che ci troviamo sul drive `c:` nella directory `\user`. I comandi in genere sono delle stringhe di più parole separate da spazi. In generale la sintassi di un comando DOS è la seguente:

[2]Se un programma contiene un errore blocca l'intero sistema MS-DOS.

[3]Vi sono parecchi modi di connettere delle memorie di massa come i dischi al computer. Fra i più diffusi appaiono le schede IDE, che vengono fornite di serie con quasi ogni computer, e le schede SCSI.

[5]Notazione: una scritta contornata da una cornice indica una stringa stampata sullo schermo dal computer, mentre il carattere ■indica la posizione del cursore.

nome_comando [*opzioni*] [*argomenti*]

dove solitamente le opzioni sono lettere o parole precedute da una barra (/) o da un segno meno (-). Facciamo qualche esempio:

`dir /p`	lista i contenuti della directory corrente sul monitor con una pausa dopo ogni schermata per facilitare la lettura
`md mydir`	crea la directory `mydir`
`copy c:\autoexec.bat autoexec.bak`	copia il file `c:\autoexec.bat` nella directory corrente con il nome `autoexec.bak`

1.4 Devices Speciali

L'MS-DOS dà all'utente la possibilità di usare alcuni devices in modo diretto, sulla riga di comando della shell. La convenzione che il DOS usa per nominare questi devices è di terminare il nome con il simbolo "due punti (:). Per esempio, alle unità di *memoria di massa* come dischi fissi, floppy, ecc., vengono assegnate delle lettere dalla A alla Z. Tradizionalmente, `a:` è il disco floppy da 3 pollici e mezzo, `b:` è il secondo floppy, `c:` è il disco fisso su cui si trova il sistema operativo[6], `d:`, ..., `z:` sono dischi fissi supplementari, dischi che si trovano su altri computer collegati in rete, eccetera.

Altri devices speciali sono: le porte seriali, di solito due e al massimo quattro, chiamate `com1:, com2:, com3:, com4:`, le porte parallele, di solito una e al massimo tre, chiamate `lpt1:, lpt2:, lpt3:`, la stampante predefinita `prn:`, la console (ovvero la tastiera e il monitor) `con:` e il device nullo `nul:`. Tutti i dati mandati a `nul:` vengono ignorati.

La differenza fra le unità di memoria di massa e i devices `com`*n*`:`, `lpt`*n*`:`, `prn:, con:, nul:` risiede nel fatto che mentre i dati mandati alle memorie di massa possono venire organizzati in un **filesystem** (cioè una struttura ad albero di directories e files), i dati mandati agli altri devices speciali non hanno questa proprietà. Così si potrà parlare di un file `c:\user\text.txt` ma non di uno `com1:\autoexec.bat`. I dati mandati ai devices che non dispongono di filesystem devono venire smistati via via che arrivano dall'hardware che sta dall'altro capo del cavo connesso alla porta del computer. Per esem-

[6] Cioè il disco fisso da cui è stato eseguito il processo di **bootstrap**, o più semplicemente **boot**, ovvero accensione del computer.

pio, i dati mandati a `con:` devono venire stampati dal monitor via via che arrivano.

2 Comandi del DOS: Generalità

I comandi del DOS si dividono in due grandi categorie: comandi interni e comandi esterni. I comandi interni vengono caricati con il bootstrap (accensione) della macchina, e risiedono quindi in memoria, sempre pronti per essere usati. Un comando esterno è un qualsiasi file eseguibile (estensioni .EXE, .COM, .BAT), e quindi molto difficilmente catalogabile. La sintassi generica di un comando esterno non dipende dal sistema operativo bensì dal programmatore che ha prodotto il file eseguibile. Basti sapere che in genere la sintassi di un comando esterno è

`[`*drive:\path*`]` *nome_comando argomenti*

dove *drive:\path* è la locazione del file eseguibile. Molto spesso si può usare l'opzione "`/?` per avere qualche spiegazione su come funziona il comando.[7] Essendo i comandi esterni così difficili da catalogare, ci limiteremo in questa sede ad una discussione dei comandi interni. A loro volta questi possono essere divisi in comandi per le directories, comandi per i files, comandi per i batch files e altri comandi.

Per ogni comando interno del DOS daremo la sintassi e la descrizione. È necessario prestare attenzione al fatto che un comando sintatticamente corretto non necessariamente funziona, ovvero è semanticamente giusto[8]. Seguire i costrutti della sintassi è necessario ma non sufficiente per usare il DOS con successo.

Per quanto concerne la notazione della sintassi, le parole scritte in carattere `teletype` vanno digitate così come appaiono, le parole scritte in *italics* sono delle variabili a cui si deve dare un valore che dipende dal contesto in cui ci si trova; gli elementi fra parentesi quadre sono opzionali. Un'indicazione del tipo

[*elemento1* |*elemento2*]

[7] Questa è una cosa da tener presente quando siamo dalla parte del programmatore: sarebbe opportuno se i programmi che scriviamo offrissero all'utente l'opzione "`/?`.

[8] Il termine "sintassi si riferisce alla correttezza grammaticale di una frase, mentre il termine "semantica indica il significato della frase. Ad esempio, la frase "la mia gatta è incinta di un dalmata è sintatticamente corretta e semanticamente scorretta perché i cani e i gatti non sono interfecondi.

significa che si può introdurre un elemento dei tipi citati oppure nessun elemento. Un'indicazione del tipo

<elemento1 |elemento2 >

significa che è necessario introdurre esattamente un elemento fra *elemento1* e *elemento2*. Un'indicazione del tipo

elemento1, ...

o

elementi

significa che è necessario introdurre almeno un elemento ma che non si specifica il numero massimo degli elementi da introdurre[9]. Dove possibile tenteremo anche di dare indicazioni semantiche, per esempio

`copy` [*opzioni*] *file_sorgente* [*file_destinazione*]

2.1 Nomi dei Files

I nomi dei files consistono di un nome di minimo un carattere e massimo otto, di un punto (.), e di un'estensione di minimo zero caratteri e massimo 3. I nomi e le estensioni possono contenere lettere, numeri, i simboli *underscore* (_), meno (-), parentesi graffe, punto esclamativo (!) e *tilde* (~); non possono contenere quasi nessun altro simbolo, in particolare non possono contenere spazi e punteggiatura al di fuori di quella già detta. Un nome di percorso (*pathname*) è una catena di directories, ad esempio `bin\i386\`, `c:\user\`, `c:\home\lsl\maths\projects\`. Un nome di file completo (*complete filename*) è un nome di file preceduto dal nome del percorso in cui si trova a partire dal drive in cui si trova, per esempio `c:\user\file.txt`, `c:\autoexec.bat`, `a:\lsl\maths\cv.tex`. Quando ci si riferisce ad una directory è possibile usare delle speciali directories "di sistema. Queste sono \ (root), `.` (current) e `..` (parent). La directory \ indica la prima directory in cima alla struttura ad albero, la directory `.` indica la directory corrente e la directory `..` indica la directory che sta un livello più in alto della directory corrente. Per esempio, se la directory corrente è `c:\user\bin`, la directory `..` si riferisce a `c:\user`. Attenzione: la directory \ (root), essendo quella che sta più in alto nell'albero, non dispone della directory `..` (parent).

I caratteri `*` e `?` hanno un significato particolare quando vengono usati nella riga di comando nei nomi dei files. Il carattere `?` sta per "qualsiasi

[9]C'è comunque la limitazione abbastanza pesante che una riga di comando deve essere lunga meno di 256 caratteri.

carattere o nessun carattere; il carattere `*` sta per "qualsiasi quantità di qualsiasi carattere, ovvero "qualsiasi parola. È evidente che `*` è la stessa cosa di `????????`. Per esempio, `*.txt` indica tutti i files con l'estensione `.txt`; `a??def.g*` indica tutti i files con qualsiasi carattere nella seconda e terza lettera del loro nome, con la prima lettera uguale ad "a, la quarta, quinta e sesta lettera uguali rispettivamente a "d, "e, "f e l'estensione che inizia con "g.

2.2 Batch Files

Un insieme di comandi del DOS può venire riunito in un unico comando usando la tecnica dei batch files. I batch files (estensione .BAT) sono dei files di testo in cui ogni riga è un comando valido DOS. Ad esempio, se prima di cambiare i contenuti di un file volessimo conservare l'originale, dovremmo fare:

```
c:\user> copy file.txt file.bak
c:\user> edit file.txt
```

Possiamo creare un batch file chiamato `editbak.bat` che automatizza questa procedura:

```
c:\user> copy con:  editbak.bat
@echo off
copy %1.txt %1.bak
edit %1.txt
^Z
```
[10]

La linea iniziale dei batch files è quasi sempre "`@echo off` per motivi tecnici che hanno a che vedere con la gestione dello schermo. Ai numeri preceduti da % viene sostituito il corrispondente argomento nella riga di comando, cosicché se lanciamo il batch file con

```
c:\user> editbak file
```

i comandi eseguiti saranno

```
c:\user> copy file.txt file.bak
```
[11]

[10] Il carattere ^Z indica che bisogna tenere premuto il tasto Ctrl mentre si schiaccia Z.

[11] Perché a %1 viene sostituito il primo argomento della riga di comando `editbak file`, che è la stringa `file`.

```
c:\user> edit file.txt
```

Gli argomenti dati sulla riga di comando del batch file possono venire referenziati dai comandi del batch file stesso attraverso le variabili %1, ..., %9, mentre la variabile %0 indica il nome del comando: ad esempio, nella riga di comando `editbak file` la variabile %0 assume il valore testo "editbak.

2.3 Ridirezione Input e Output

Quando un comando DOS stampa qualcosa sullo schermo in realtà lo manda attraverso un flusso di dati che si chiama **stdout**[12] al device `con:` (si ricorda che il device `con:` indica la console, ovvero la tastiera e il monitor). Del pari, quando un comando DOS chiede un input dalla tastiera in realtà lo accetta attraverso un flusso di dati chiamato **stdin** dal device `con:`. I termini stdin e stdout si leggono "standard input e "standard output. Se non viene specificato altrimenti, questi flussi puntano automaticamente al device `con:`. Questa predefinizione può essere cambiata con il simbolo >[13]: perché un comando usi un file o un device alternativo a `con:` per il flusso stdout la sintassi è

nome_comando [*opzioni*] [*argomenti*] > *device*

dove *device* può essere un device speciale, come `com1:, nul:, prn:, lpt1:` eccetera, oppure il nome di un file (questo permette di scrivere un file il contenuto del quale è il testo che sarebbe apparso sullo schermo senza ridirezione). Si noti che, se il nome del file a cui si sta ridirigendo l'output dovesse esistere già sul disco, il file esistente verrà cancellato prima di procedere alla scrittura. Per ovviare a questo inconveniente si può sostituire il segno > con due segni >>: in questo modo i dati provenienti dal flusso stdout verranno scritti in coda al file esistente.

La corrispondente sintassi per la ridirezione dello stdin è

nome_comando [*opzioni*] [*argomenti*] < *device*

Il simbolo | (chiamato *pipe*) serve a passare lo stdout di un comando allo stdin del comando successivo. Per esempio,

```
c:\user> type c:\autoexec.bat | more
```

[12] Certi messaggi che appaiono sullo schermo, come per esempio i messaggi di errore, usano un flusso diverso chiamato **stderr** che non si può ridirigere.

[13] Attenzione: > non è un simbolo per indicare una sintassi. Bisogna in effetti immettere un carattere > sulla riga di comando.

prende il contenuto del file `c:\autoexec.bat` e lo manda a stdout. Anziché finire sullo schermo i dati vengono passati allo stdin del comando `more` che li mette sullo schermo fermandosi ad ogni schermata per permettere all'utente di leggere comodamente.

3 Comandi del DOS: Comandi Relativi alle Directories

3.1 Dir

Il comando `dir` serve a mostrare il contenuto di una directory o le specificazioni (spazio occupato in bytes, data e ora di creazione) di un file. L'output del comando `dir` viene messo su stdout. Se non si usa ridirezione viene mostrato sullo schermo.

`dir` [*opzioni*] [*nome_directory1* |*nome_file1, ...*]

3.1.1 Opzioni

/?	Guida
/P	Pausa dopo ogni schermata
/W	Stampa la lista in cinque colonne
/A:[[-]drhas]	Lista solo i files con gli attributi specificati: d: directory, r: sola lettura, h: nascosto a: archivio, s: sistema (il prefisso - indica "senza gli attributi specificati)
/O:[[-]nsedg]	Lista in ordine: n: per nome, s: per spazio occupato (i più piccoli prima) e: per estensione, d: per data e ora (i più vecchi prima) g: raggruppa directories all'inizio della lista (il prefisso - usa l'ordine inverso)
/S	Lista contenuti della directory specificata e di tutte le sue sottodirectory

3.1.2 Esempi

Il primo esempio mostra i contenuti della directory corrente (la directory corrente e il drive corrente sono quelli indicati nel prompt).

```
c:\user> dir

 Volume in drive C is Euryalus
 Volume Serial Number is CB51-3D8F

 Directory of C:\user

11/04/97  05:50p        <DIR>          .
11/04/97  05:50p        <DIR>          ..
10/29/97  11:35p                    80 AUTOEXEC.BAT
11/04/97  05:50p                   160 CONFIG.SYS
               4 File(s)            240 bytes
                             43,695,616 bytes free
```

Il prossimo esempio cerca tutti i files nascosti nel disco corrente.

```
c:\user> dir /s /a:h \*.*

 Volume in drive C is XANTHUS
 Volume Serial Number is 3F63:1CE4

Directory of C:\*.*

IO       SYS        40,982 05-31-94   6:22a
MSDOS    SYS        38,172 05-31-94   6:22a
DRVSPACE BIN        66,294 05-31-94   6:22a
        3 File(s)        145,448 bytes

Listed file(s):
        3 File(s)        145,448 bytes
```

Si noti che il formato dell'output è diverso in quanto il primo comando è stato dato su una macchina con Windows NT come sistema operativo mentre il secondo è stato dato su un MS-DOS 6.22.

3.2 Cd

Il comando `cd` serve a cambiare la directory corrente oppure, se usato senza alcun argomento, a stampare la directory corrente. Nel caso in cui sia usato senza argomento, l'output viene messo su stdout e quindi può essere ridiretto.

`cd` [*nome_directory*]

3.2.1 Note

- Il nome della directory sulla riga di comando (*nome_directory*) deve riferirsi ad una directory esistente sul disco corrente.
- *nome_directory* non può riferirsi ad una directory esistente su un disco diverso.

3.2.2 Esempi

L'esempio seguente esce dalla directory \`user` e poi ci rientra e stampa sullo schermo il nome della directory corrente.

```
c:\user> cd ..
c:\> cd user
c:\user> cd
c:\user
c:\user>
```

3.3 Md

Il comando `md` crea una nuova directory, opzionalmente anche su un drive diverso da quello corrente.

`md` [*drive:\path*]*nome_directory*

3.3.1 Note

Questo comando manda la stringa "Directory già esistente al flusso stderr (vedi nota 12 a piè pagina per la definizione di stderr) nel caso che la directory specificata da *nome_directory* esista già. Il comando `mkdir` è sinonimo di `md`.

3.3.2 Esempi

L'esempio seguente crea una directory al livello inferiore rispetto alla directory corrente e ci entra.

```
c:\user> md prova
```

```
c:\user> cd prova
c:\user\prova>
```

3.4 Rd

Il comando `rd` rimuove una directory vuota, opzionalmente anche su un drive diverso da quello corrente. Il comando `rmdir` è sinonimo di `rd`.

`rd` [*drive:\path*]*nome_directory*

3.4.1 Note

Per ogni tipo di errore che può accadere nell'esecuzione del comando, la stringa di notifica mandata a stderr è "Percorso non valido, non è una directory o la directory non è vuota.

3.4.2 Esempi

L'esempio seguente crea una directory al livello inferiore rispetto alla directory corrente, ci entra, esce nuovamente e la rimuove.

```
c:\user> md prova
c:\user> cd prova
c:\user\prova> cd ..
c:\user> rd prova
c:\user>
```

4 Comandi del DOS: Comandi Relativi ai Files

4.1 Copy

Il comando `copy` consente di copiare il contenuto di uno o più files in un'altra locazione. Se si copia un singolo file si può anche cambiare il nome durante la copia. Il comando `copy` supporta l'uso delle wildcards (i caratteri * e ? nei nomi dei files) per indicare gruppi di files.

`copy` [*opzioni*] *file_sorgente* [*file_destinazione* |*percorso*]

4.1.1 Opzioni

/?	Guida
/V	Verifica che i nuovi files siano scritti correttamente (impiega più tempo della semplice copia)
/-Y	Chiede conferma prima di sovrascrivere un file di destinazione già esistente
/Y	Il contrario di /-Y

4.1.2 Esempi

Gli esempi seguenti copiano il file `c:\autoexec.bat` alla directory `c:\user`. Le differenti forme in cui appare il comando dipendono dalla directory corrente (quella indicata nel prompt). In particolare nell'esempio 2 non appare la destinazione: quando viene omessa viene presa come destinazione la directory corrente.

1)
```
c:\> copy autoexec.bat user
    1 file copiato/i
c:\>
```
2)
```
c:\user> copy ..\autoexec.bat
    1 file copiato/i
c:\user>
```
3)
```
a:\dos> copy c:\autoexec.bat c:\user
    1 file copiato/i
a:\dos>
```

Il prossimo esempio mostra come cambiare il nome ad un file mentre viene copiato ad un'altra locazione.

```
c:\> copy autoexec.bat user\altro.nom
    1 file copiato/i
c:\>
```

L'esempio seguente mostra come copiare gruppi di files.

```
c:\> copy c:\dos\*.* c:\user
    22 file copiato/i
c:\>
```

Vediamo ora come copiare un file alla stampante predefinita.

```
c:\user> copy myfile.txt prn:
    1 file copiato/i
c:\user>
```

...oppure alla stampante collegata alla prima porta parallela.

```
c:\user> copy myfile.txt lpt1:
    1 file copiato/i
c:\user>
```

Il comando `copy` può anche essere usato per creare un file: si copia l'input proveniente dalla tastiera (console) ad un file.

```
c:\user> copy con:  myfile.txt
Questo è il contenuto di myfile.txt
aaaiiighh!!
^Z
    1 file copiato/i
c:\user>
```

Nell'esempio seguente il file viene copiato allo schermo (alla console). Questo è un modo per mostrare i contenuti di un file di testo sullo schermo.

```
c:\user> copy myfile.txt con:
Questo è il contenuto di myfile.txt
aaaiiighh!!
    1 file copiato/i
c:\user>
```

4.2 Del

Il comando `del` serve per cancellare uno o più files. Il comando `del` supporta l'uso delle wildcards (i caratteri * e ? nei nomi dei files) per indicare gruppi di files. Il comando `erase` è sinonimo di `del`.

`del` [*opzioni*] *file*

4.2.1 Note

Se *file* è il nome di una directory, tutti i files in quella directory verranno cancellati. Inoltre, ogni volta che vengono cancellati tutti i files in una directory il DOS chiede conferma esplicita prima di procedere.

4.2.2 Opzioni

/? Guida
/P Chiede conferma prima di procedere all'eliminazione del file

4.2.3 Esempi

L'esempio seguente svuota una directory, esce al livello più alto e poi la rimuove.

```
c:\user> del *.*
Tutti i file della directory verranno eliminati!
Continuare? (S/N) s
c:\user> cd ..
c:\> rd user
c:\>
```

Nell'esempio seguente creiamo una directory, copiamo un file dentro la directory appena creata e poi lo cancelliamo.

```
c:\> md user
c:\> copy autoexec.bat user
    1 file copiato/i
c:\> del user\autoexec.bat
```

```
c:\>
```

4.3 Ren

Il comando `ren` serve a cambiare il nome a uno o più files. Il comando supporta l'uso di wildcards (i caratteri * e ?) per indicare gruppi di files. Il comando `rename` è sinonimo di `ren`.

`ren` [*drive:\path*]*nome_file1 nome_file2*

4.3.1 Note

Si noti che i nomi *nome_file1* e *nome_file2 non* comprendono drive e percorsi. In particolare non si può spostare un file ad un'altra locazione.

4.3.2 Esempi

L'esempio seguente copia un file in `c:\user` e lo rinomina; poi copia ancora un file in `c:\user` e li rinomina entrambi.

```
c:\user> copy ..\autoexec.bat .
    1 file copiato/i
c:\user> ren autoexec.bat myfile.txt
c:\user> copy ..\autoexec.bat myfile2.txt
c:\user> ren *.txt *.bat
c:\user>
```

4.4 Type

Il comando `type` serve a mettere il contenuto di un file di testo su stdout per la stampa sullo schermo o la ridirezione dell'output. Il comando *non* supporta le wildcards.

`type` *file*

4.4.1 Note

Se *file* è il nome di un file non esistente, la stringa "File non trovato - *file* viene messa su stderr.

4.4.2 Esempi

L'esempio seguente stampa un file di testo sullo schermo.

```
c:\user> type c:\autoexec.bat
@echo off
set path=c:\dos
c:\user>
```

5 Comandi del DOS: Altri Comandi

5.1 Set

Il comando `set` assegna un valore (testo) ad una variabile. Se viene usato senza argomenti mostra i valori di tutte le variabili assegnate.

`set` [*nome_var*=[*testo*]]

5.1.1 Esempi

Nell'esempio seguente si assegna un valore alla variabile di sistema PATH.

```
c:\user> set path=c:\dos c:\user>
```

5.2 Echo

Il comando `echo` serve a passare una stringa di caratteri allo stdout; se non vi è alcuna ridirezione dell'output la stringa viene stampata sullo schermo. Se manca l'argomento del comando, viene indicato lo status di ECHO. Lo status di ECHO può assumere due valori: ON e OFF. Quando ECHO è OFF il prompt non viene stampato sullo schermo; viceversa, quando ECHO è ON, il prompt viene mostrato. Questo è il motivo del comando `@echo off`

all'inizio dei batch files: non vogliamo che durante l'esecuzione del batch venga mostrato il prompt. Alla fine del batch file lo status di ECHO viene ripristinato a ON.

`echo` [*testo*]

5.2.1 Note

- L'argomento *testo* può contenere una o più variabili (vedi comando `set`). Per stampare il contenuto di una variabile si include il nome della variabile fra due segni di percento, per esempio `%path%`.
- Per stampare una linea bianca, il comando è `echo.`, cioè il comando `echo` seguito da un puntino. Attenzione: questa è l'unica eccezione alla regola della sintassi generica dei comandi che vuole che si separino con degli spazi i comandi e gli argomenti. *Non c'è uno spazio fra* `echo` *e il puntino.*

5.2.2 Esempi

Il seguente esempio scrive sullo schermo il valore della variabile `path`.

```
c:\user> echo Il valore di PATH è %path%
Il valore di PATH è C:\DOS
c:\user>
```

Nel prossimo esempio passiamo lo stdout di un comando `echo` allo stdin di un comando `del`, e mandiamo al device `nul:` lo stdout di `del`. Questo serve a sopprimere il messaggio "Tutti i file nella directory verranno cancellati. Proseguire con l'operazione (S/N)? che appare quando si cancellano tutti i files in una directory. Attenzione: questa forma è da usare con cautela.

```
c:\user> echo s | del *.*
c:\user>
```

5.3 For

Esegue un comando specificato per ogni file in un gruppo di files.

`for` %%*nome_var* in (*files...*) do *nome_comando*

5.3.1 Note

Durante il ciclo la variabile *nome_var* assume come valori i nomi dei files contenuti nel gruppo di files indicato da *files...*, e può essere usata nel comando *nome_comando*.

5.3.2 Esempi

Nell'esempio che segue creiamo un batch file `typetxt.bat` in cui usiamo il comando `for` e il comando `type` con la ridirezione di stdout per listare, schermata per schermata, tutti i files con l'estensione .TXT nella directory corrente.

```
c:\user> copy con:  typetxt.bat
@echo off
for %%a in (*.txt) do type %%a >>file.tmp
more <file.tmp
del file.tmp
^Z
c:\user>
```

5.4 If

Esegue un comando se la condizione specificata è vera.

`if [not]` <*testo1*==*testo2* | `exist` *file* | `errorlevel` *numero*> *nome_comando*

5.4.1 Note

- Nella prima condizione le stringhe *testo1* e *testo2* possono contenere delle variabili.
- Lo status di ERRORLEVEL dipende dalla conclusione del comando eseguito *prima* del comando `if`.
- La parola chiave `not` verifica la falsità della condizione.

5.4.2 Esempi

Nel seguente esempio si verifica che il valore di una variabile non sia nullo.

```
c:\user> if %path%=‘‘ echo La variabile PATH non è stata
immessa.
c:\user>
```

Si noti l'uso delle virgolette. Quando in *testo1* o *testo2* appaiono delle variabili bisogna essere sicuri che, anche se non sono state definite, da nessuna delle parti dell'uguale appaia la stringa vuota. Se non avessimo messo le virgolette e %PATH% fosse vuota, il comando diventerebbe

```
if == echo La variabile PATH non è stata immessa.
```

che è un errore di sintassi. Le virgolette non hanno nulla di particolare se non che sono "estetiche. Sarebbe analogo dire

```
if !%path%==!  echo La variabile PATH non è stata immessa.
```

Verifichiamo adesso l'esistenza di un file. Se esiste lo stampiamo sullo schermo.

```
c:\user> if exist c:\autoexec.bat type c:\autoexec.bat
c:\user>
```

5.5 ctty

Serve a trasferire il controllo del computer ad un device diverso dalla console (`con:`). Il comando viene spesso usato nella connessione di due computer tramite cavo seriale.

`ctty` *device*

6 Comandi del DOS: Comandi dei Batch Files

I seguenti comandi possono essere usati esclusivamente nei batch files. Negli esempi non appariranno i prompt bensì i testi dei batch files di esempio.

6.1 Pause

Il comando `pause` sospende l'esecuzione finché l'utente non preme un tasto. Il seguente messaggio viene messo su stdout durante l'attesa: "Premi un tasto

per continuare .... Se viene premuto CTRL-C (^C) l'esecuzione del batch file viene cancellata.

`pause`

6.1.1 Esempi

Vediamo come sopprimere il messaggio predefinito e metterne uno a piacere.

```
echo Premi un tasto per continuare o CTRL-C per smettere.
pause > nul:
```

6.2 Goto

Il comando `goto` trasferisce il controllo ad un'altra parte del batch file.

`goto` *etichetta*

6.2.1 Note

L'*etichetta* è una linea del batch file in cui compare una stringa preceduta dai due punti (:).

6.3 Esempi

Vediamo come mimare il comportamento di un blocco logico if-then-else con i comandi `if` e `goto`.

```
if not %1==‘‘ goto else
echo Il batch file è stato lanciato senza argomenti
goto endif
:else
echo L’argomento è %1
:endif
```

6.4 Call

Esegue una chiamata ad un altro batch file e alla fine di quest'ultimo il controllo ritorna al primo batch file (quello in cui appare il comando `call`).

`call` *nome_batch_file*

6.4.1 Note

Il comando `call` viene spesso eseguito come *comando* di `for` per far sì che ad ogni ciclo vengano eseguiti più comandi.

6.4.2 Esempi

Il seguente file esegue l'installazione di un software da un drive ad un altra locazione. Il comando `call` viene eseguito nel corpo di un `for`.

```
-----------INSTALL.BAT-----------
@echo off
echo Welcome to Woderful Software installation!
if "%1"=="" goto no1
if "%2"=="" goto no2
if "%1"=="/?" goto help
if not exist %1\lic.txt goto nodrive1
md %2 > nul:
echo nulla > %2\tmp
if not exist %2\tmp goto nodrive2
del %2\tmp > nul:
type %1\lic.txt | more
echo Sei sicuro di volere installare?
echo Premi ctrl-c per uscire
echo Premi qualsiasi tasto per installare
pause > nul:
copy %1\*.* %2 > nul:
for %%a in (%2\*.txt) do call %1\inst2.bat %%a %2
type %2\tmp | more
del %2\tmp
goto fine
:no1
echo %0: ERRORE: Mancano gli argomenti della riga di comando
goto help
:no2
echo %0: ERRORE: Manca il secondo argomento
goto help
:nodrive1
echo %0: ERRORE: Drive %1 non accessibile alla lettura
```

```
goto fine
:nodrive2
echo %0: ERRORE: Drive o percorso %2 non accessibili alla scrittura
goto fine
:help
echo %0 - Leo Liberti 29/10/1997
echo Sintassi:
echo [drive:\path]install drive1: drive2:\path
goto fine
:fine

-------------INST2.BAT-------------
@echo off
if not %1==%2\LIC.TXT type %1 >> %2\tmp
```