

Comandi UNIX

Classi di comandi UNIX

- reperire informazioni su comandi, programmi, file....
- gestire filesystem
- operare su file e directory
- elaborare testi
- sviluppare software
- comunicare tra utenti (sullo stesso host)
- operare in remoto
-(*comandi “di utilità” vari*)
- amministrare il sistema
 - utenti e gruppi
 - dispositivi
 - software

Formato dei comandi

`comando [argomento ...]`

Gli argomenti possono essere:

- **opzioni o flag (-)**
- **parametri**

separati da almeno un separatore (di default il carattere spazio)

Esempio d' uso

```
ls -l -F file1 file2 file3
```

Forme equivalenti:

```
ls -F -l file1 file2 file3
```

```
ls -lF file1 file2 file3
```

```
ls -F1 file1 file2 file3
```

Reperire informazioni

man
which
whereis

Accede al manuale in linea di Unix

Cerca il path di un file eseguibile

Determina il percorso di binari, sorgenti e documentazione per un programma

Il manuale in linea di Unix

- Unix ha un manuale di riferimento molto completo, accessibile “in linea” mediante il comando **man** ;
- Il manuale è in genere distribuito su più directory, ma appare come un unico grande volume virtuale, organizzato in **sezioni** e **sottosezioni**;
- Ogni sezione e sottosezione dispone di una pagina chiamata **intro**, in cui sono disponibili informazioni introduttive;
- Ogni (sotto)sezione è composta di **pagine** (logiche);
- Ogni pagina descrive **un singolo argomento**;
- Gli **argomenti** possono essere comandi, file di configurazione, funzioni di libreria, etc...

Le sezioni del manuale

Sezione	Contenuti
1	Commands
2	System Calls
3	Library Functions
4	Administrative Files
5	Miscellaneous Information
6	Games
7	I/O and Special Files
8	Maintenance Commands
.....	<i>Optional sections</i>

Il formato di una pagina

User Commands

which(1)

NAME

which - locate a command; display its pathname or alias

SYNOPSIS

which [filename...]

DESCRIPTION

which takes a list of names and looks for the files which would be executed had these names been given as commands. Each argument is expanded if it is aliased, and searched for along the user's path. Both aliases and path are taken from the user's .cshrc file.

FILES

~/.cshrc source of aliases and path values
/usr/bin/which

Il formato di una pagina

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

csh(1), attributes(5)

DIAGNOSTICS

A diagnostic is given for names which are aliased to more than a single word, or if an executable file with the argument name was not found in the path.

Il formato di una pagina

NOTES

which is not a shell built-in command; it is the UNIX command, /usr/bin/which

BUGS

Only aliases and paths from ~/.cshrc are used; importing from the current environment is not attempted. Must be executed by csh(1), since only csh knows about aliases.

To compensate for ~/.cshrc files in which aliases depend upon the prompt variable being set, which sets this variable to NULL. If the ~/.cshrc produces output or prompts for input when prompt is set, which may produce some strange results.

man (manual)

```
man [options] [-M path] [-s section] name...
```

```
man [-M path] -k keyword
```

```
man [-M path] -f file
```

Nella prima forma, visualizza le pagine del manuale specificate tramite **name**.

Nella seconda e terza forma visualizza, rispettivamente, un listato delle pagine disponibili che corrispondono alla parola chiave **keyword**, oppure di nome **file**

-M permette di specificare il path in cui cercare le pagine del manuale

-s permette di specificare la sezione

options sono opzioni sia di ricerca che di visualizzazione

NOTE:

Nella versione GNU del comando le sezioni si specificano senza il flag -s.

Se la sezione non è specificata, viene selezionata la prima occorrenza.

man – esempi

Documentazione sul comando man

```
man man
```

Ricerca delle pagine del manuale relative alla parola chiave “firewall”

```
man -k firewall
```

Documentazione sui comandi vimtutor e vim dal manuale /opt/sfw/man

```
man -M /opt/sfw/man vimtutor vim
```

Visualizza l'introduzione relativa alla sezione “syscalls”

```
man -s 2 intro
```

Ricerca delle pagine del manuale di nome intro

```
man -f intro
```

which

```
which [filename...]
```

simula la ricerca che farebbe la shell per avviare i programmi indicati come argomento e determina la posizione di quelli che verrebbero scelti.

Utile per sapere:

- dove si trova un comando determinato;
- quale programma viene scelto effettivamente nel caso ne esistano diversi con lo stesso nome collocati in posizioni differenti nell'albero di directory.

Esempi di esecuzione

```
gio$ which ls pippo  
/usr/bin/ls  
no pippo in /usr/bin /usr/ucb /usr/sfw/bin /usr/openwin/bin  
gio$
```

whereis

whereis [*options*] **file...**

localizza i file binari, i sorgenti e/o le pagine di manuale per i file specificati come argomento

Alcune opzioni

- b** cerca solo i file binari
- m** cerca solo le pagine del manuale
- s** cerca solo i sorgenti

Esempi di esecuzione

```
gio$ whereis ls pippo
ls: /usr/bin/ls /usr/ucb/ls /usr/man/man1/ls.1
   /usr/man/man1b/ls.1b
pippo:
gio$
```

Gestione di file system

`mount`

Collegamento (innesto) di un file system in quello globale.

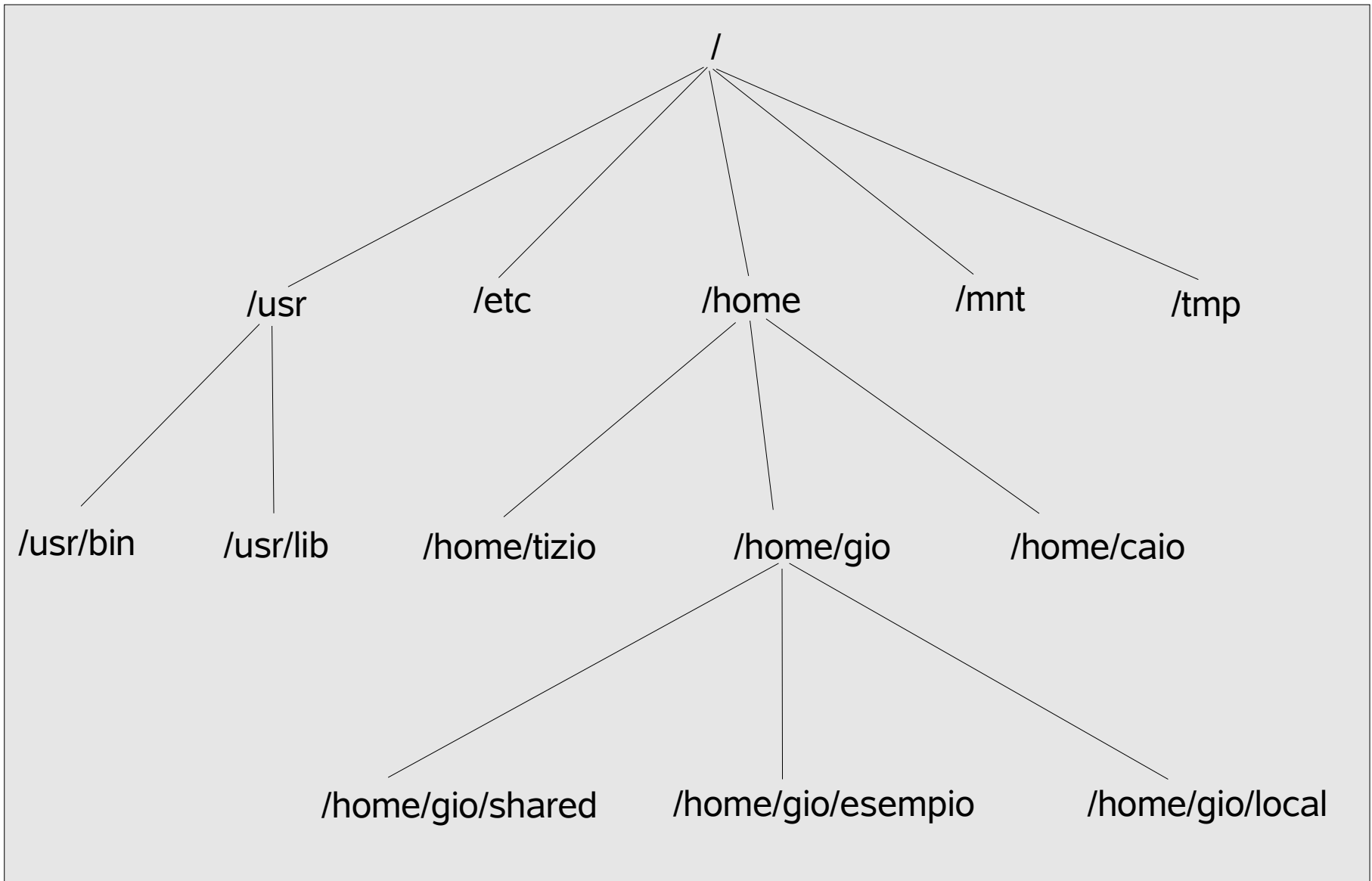
`umount`

Distacco di un file system da quello globale.

`df`

Utilizzo del disco.

Il file system di Unix



mount

mount -a [options]

mount [options] [device] [directory]

Nella prima forma vengono montati tutti i file system elencati in un file di configurazione (/etc/fstab o /etc/vfstab) che non abbiano l'opzione “no auto”

Nella seconda forma, permette di attaccare il file system presente sul dispositivo *device* al file system globale di Unix, nella posizione specificata da *directory*. Senza argomenti permette di visualizzare tutti i file montati.

Alcune opzioni

- t (-F)** Specifica il tipo di file system. Se il nome del tipo di file system viene preceduto dalla sigla **no**, si intende che quel tipo deve essere escluso.
- v** mostra diagnostica dettagliata
- f (-V)** Simula solo il montaggio; utile per valutare gli effetti del comando, soprattutto se usata insieme all'opzione precedente
- (-p)** mostra i file system montati in un formato più semplice

umount

```
umount -a [options]
```

```
umount [options] [device | directory]
```

esegue l'operazione inversa di **mount**: smonta i file system.

Nella prima forma, smonta i file system elencati nell'apposito file di configurazione (/etc/mstab o /etc/mnttab), eventualmente in base alle opzioni, con l'eccezione di quelli necessari al funzionamento del sistema.

Nella seconda forma, smonta il file system individuato dal device *device* o dal punto di attacco *directory*.

L'operazione (a meno di usare l'opzione **-f**, valida solo per alcuni sistemi) avviene solo se non ci sono più attività in corso su quei file system.

Nota:

Le operazioni di (s)montaggio con (u)mount richiedono privilegi speciali.

mount e umount – esempi

Esempi d'uso

Monta il floppy su un sistema Linux x86 in /tmp

```
mount /dev/fd0 /tmp
```

Smontaggio del dispositivo precedente

```
umount /tmp
```

Monta un CD sul lettore ottico primario di un sistema Solaris x86 nella directory /mnt/cdrom

```
mount -F hsfs /dev/dsk/c0t0d0s0 /mnt/cdrom
```

Forza lo smontaggio del precedente dispositivo

```
umount -f /mnt/cdrom
```

df (disk free)

df [*options*] [*device...*]

permette di conoscere lo spazio a disposizione di una o di tutte le partizioni che risultano montate.

Se non vengono indicati i nomi dei dispositivi, si ottiene l'elenco completo di tutti i dispositivi attivi, altrimenti l'elenco si riduce a quelli specificati.

Alcune opzioni

- (-g)** Fornisce informazioni molto dettagliate, riportando su diverse linee dati relativi a blocchi, frammenti, file, tipo di file system, etc.
- h** Emette le informazioni su dimensione, spazio occupato e spazio libero in un formato facilmente leggibile

Gestione delle Directory

`mkdir`

Crea una directory

`rmdir`

Elimina una directory vuota

`pwd`

Emette il percorso della directory corrente

`basename`

Emette l'ultimo nome di un percorso

`dirname`

Emette il nome della directory estraendolo da un path

`ls`

Elenca il contenuto di una o più directory

`du`

Calcola lo spazio utilizzato da una serie di directory e subdirectory

mkdir (make directory)

```
mkdir [options] directory...
```

Crea le directory passate come argomento, secondo le opzioni specificate

Alcune opzioni

- m *mode*** Specifica il modo *mode* per la directory da creare
- p** Genera anche tutte le directory gerarchicamente superiori necessarie. Il modo per le directory “genitrici” sono impostati al valore della maschera di modo, modificato in modo da aggiungere i permessi **w** ed **x** per il proprietario

Esempi d' uso

Crea la directory *pippo* a partire da quella corrente con modo 777

```
mkdir -m 755 pippo
```

Crea le directory *esami prove giudizi* a partire da quella corrente

```
mkdir -p esami/prove/giudizi
```

rmdir (remove directory)

```
rmdir [options] directory...
```

Rimuove le directory indicate secondo le opzioni specificate.

Per poter essere rimosse, le directory devono essere vuote e la loro directory genitrice deve avere il permesso in scrittura

Alcune opzioni

- p** rimuove ricorsivamente directory e subdirectory contenute nel path **directory** (ammesso che non contengano altri file)
- (-s)** non mostra eventuali messaggi di errore se è attiva l'opzione **-p**

Esempi d' uso

Rimuove la directory pippo a partire da quella corrente

```
rmdir pippo
```

Rimuove le directory esami prove giudizi a partire da quella corrente

```
rmdir -p esami/prove/giudizi
```

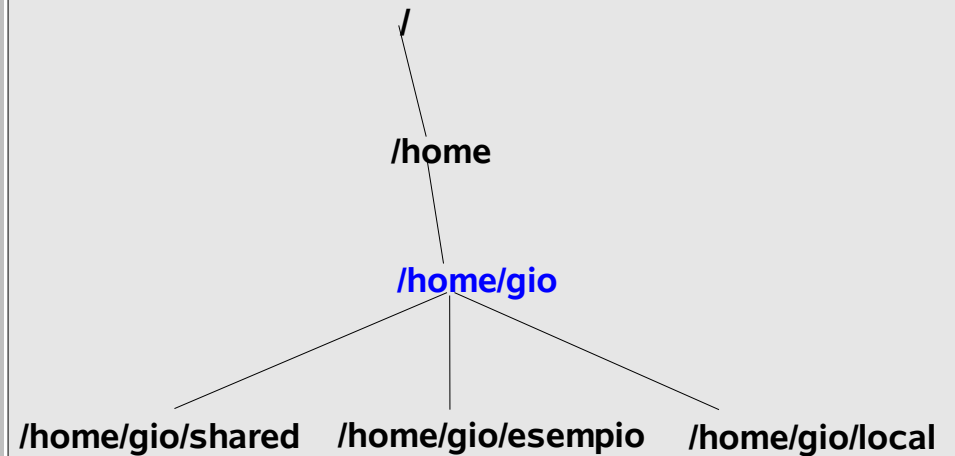
pwd (print working directory)

pwd

stampa il path della directory di lavoro corrente

Esempi di esecuzione

```
gio$ pwd
/home/gio
gio$
```



cd (change directory)

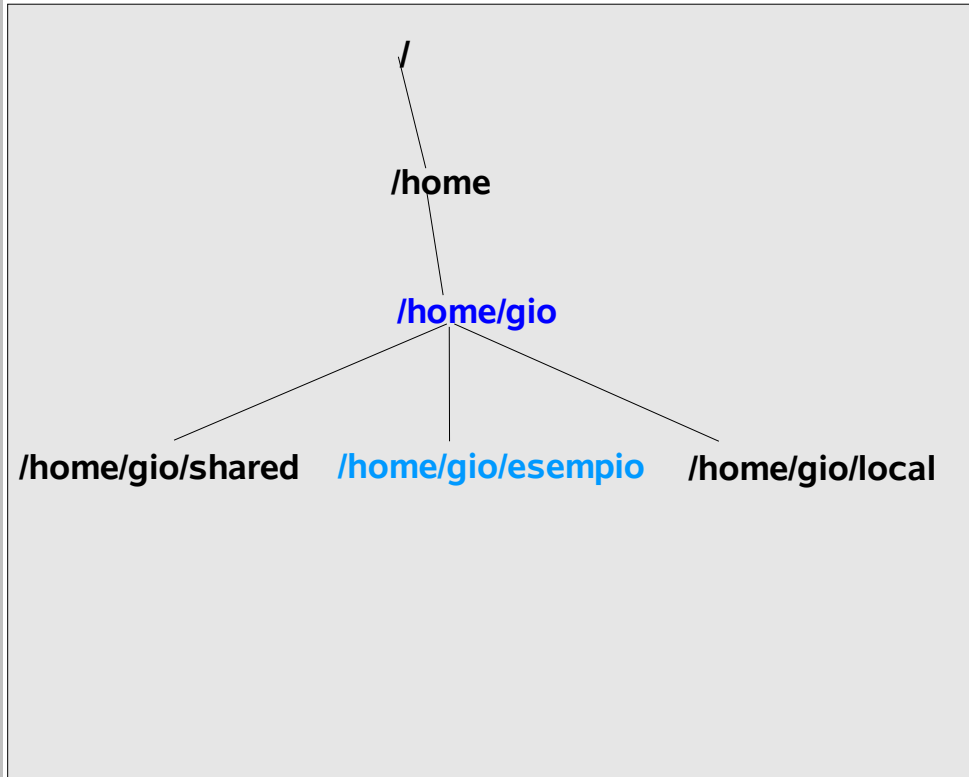
`cd [directory]`

la directory specificata diviene la **working directory**

se nessuna directory è specificata, si viene posizionati nella **home directory**

Esempi di esecuzione

```
gio$ cd /home/gio/esempio
gio$ pwd
/home/gio/esempio
gio$ cd
gio$ pwd
/home/gio
gio$ cd esempio
gio$ pwd
/home/gio/esempio
gio$
```



basename

basename *pathname* [*suffix*]

rimuove dal percorso la parte anteriore contenente l'informazione sulla directory

rimuove anche il suffisso indicato dalla parte finale del nome rimanente

Esempi di esecuzione

```
gio$ basename /idrogeno/eliografia  
eliografia  
gio$ basename /idrogeno/eliografia grafia  
elio  
gio$
```

dirname (directory name)

dirname *pathname*

rimuove dal percorso la parte finale a partire dall'ultima barra obliqua (/) di divisione tra l'informazione della directory e il nome del file

se il percorso contiene solo un nome di file, il risultato è un punto singolo (.)

Esempi di esecuzione

```
gio$ dirname /idrogeno/eliografia  
/idrogeno  
gio$ dirname pippo  
.  
gio$
```

ls (list)

```
ls [options][directory...]
```

lista, in una modalità dipendente dalle opzioni selezionate, il contenuto delle directory indicate

se nessuna directory è indicata, lista il contenuto della working directory

Alcune opzioni

- a mostra anche i **file nascosti** (file il cui nome incomincia col punto)
- l formato esteso, con informazioni su modo, proprietario, dimensione, etc dei file
- s fornisce la dimensione in blocchi dei file
- t lista i file nell'ordine di modifica (prima il file modificato per ultimo)
- 1 elenca i file in una singola colonna
- F aggiunge / al nome delle directory e * al nome dei file eseguibili
- R si chiama ricorsivamente su ogni subdirectory
- i fornisce l'**i-number** (numero di i-nodo) dei file

ls – esempi

Esempi di esecuzione

```
gio$ ls /home/gio/esempio
```

```
b c d
```

```
gio$ ls -1ai esempio
```

```
30657 .
```

```
4 ..
```

```
30658 .a
```

```
30659 b
```

```
30660 c
```

```
30661 d
```

```
gio$ cd esempio
```

```
gio$ ls -at
```

```
. .. c b .a d
```

```
gio$ ls -l esempio
```

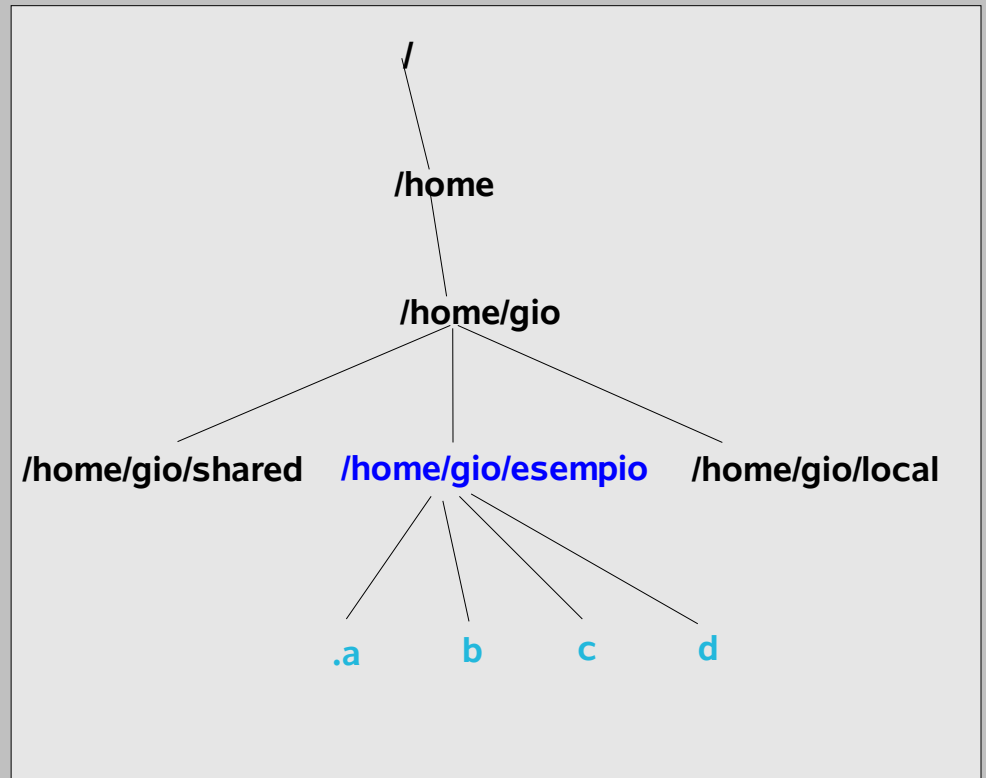
```
totale 8
```

```
-rw-r--r-- 1 gio staff 10 mar 25 22:11 b
```

```
-rw-r--r-- 1 gio staff 15 mar 25 22:11 c
```

```
drwxr-xr-x 2 gio staff 512 mar 25 22:10 d
```

```
gio$
```



ls – campi del formato esteso

dimensione totale occupata dai file (in blocchi)

link (riferimenti al file)

dimensione (byte)

nome

```
totale 8
-rw-r--r-- 1 gio staff 10 mar 25 22:11 b
-rw-r--r-- 1 gio staff 15 mar 25 22:11 c
drwxr-xr-x 2 gio staff 512 mar 25 22:10 d
```

tipo

permessi

proprietario

gruppo primario

data ultima modifica

(**r**)ead, (**w**)rite, e(**x**)ecute, (**s**)et uid bit, s(**t**)icky bit, ...

(**d**)irectory, (**l**)ink, (**c**)haracter special file, (**b**)lock special file, (-) ordinary file, ...

du (disk usage)

`du [options] file...`

emette informazioni sull'utilizzo dello spazio disco da parte di *file*, che può rappresentare un file o una directory.

se *file* è una directory, l'output standard prevede l'elencazione di tutti i file e le subdirectory in modo ricorsivo, con il calcolo della dimensione di ciascuno.

Alcune opzioni

- s** invece dell'output standard, mostra solo lo spazio disco occupato complessivamente da *file*.
- k** esprime le dimensioni in kibi byte (1024 byte), invece che in blocchi (512 byte)

Gestione dei File

cp	Copia un file in un altro
ln	Crea un collegamento
mv	Rinomina un file
rm	Cancella un file
chown	Cambia proprietario e gruppo primario di un file
chmod	Cambia i permessi di un file
touch	Cambia il tempo di ultimo accesso ad un file
file	Determina il tipo di file in base al magic number
find	Ricerca file in base ad opportuni criteri, permettendo di eseguirvi (opzionalmente) delle operazioni
tar	archivia un file
gzip, gunzip	Comprime (Decomprime) un file

cp (copy)

cp [options] *source...* *target*

copia un file in un altro file oppure uno o più file in una directory

- se vengono specificati solo i nomi di due file, il primo viene copiato sul secondo
- se vengono specificati solo due nomi, e se il secondo nome indicato è una directory, *source* viene copiato con lo stesso nome nella directory *target*. Se *source* è una directory, la copia avviene solo con opzioni particolari.

Se vengono indicati più di due nomi, il file *target* deve essere una directory e vengono generate le copie dei *source* in *target*. In mancanza di opzioni particolari, le directory non vengono copiate.

Alcune opzioni

- r se *source* e *target* sono directory, copia ricorsivamente *source*, i suoi file e le sue subdirectory in *target*
- i opera in modo interattivo, chiedendo una conferma se la copia comporta la cancellazione di un *target* preesistente

cp - esempi

Esempi d' uso

Copia il file pippo nella directory corrente nel file /tmp/pippo.back

```
cp pippo /tmp/pippo.back
```

Copia il file /tmp/pippo.back e la directory dir nella directory nuovadir

```
cp -r /tmp/pippo.back dir nuovadir
```

Copia il file pippo nel file pippo2

```
cp pippo pippo2
```

Nota:

Una copia duplica i dati relativi ai *source* in *target*. Se *target* rappresenta un nuovo file, viene impegnato un nuovo inodo.

ln (link)

ln [*options*] *source*... *destination*

crea un collegamento tra la preesistente *source* e la nuova *destination*.

Se vi sono più di due argomenti, *destination* deve indicare il path di una directory esistente, ed i link saranno creati sotto tale directory con gli stessi nomi dei file *source*.

In assenza di opzioni il link generato è di tipo hard e l'operazione ha successo solo se *source* e *destination* afferiscono allo stesso file system. In questo caso il collegamento consiste nel fatto che il path di *source* e quello ottenuto tramite *destination* fanno riferimento allo stesso i-nodo.

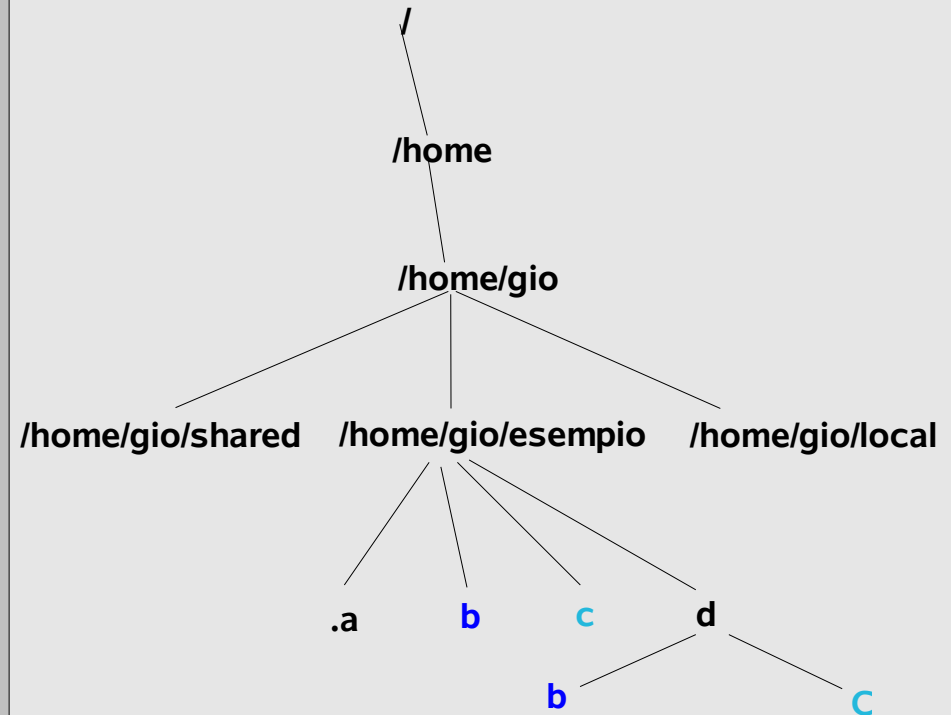
Alcune opzioni:

- s genera un link di tipo soft, e in tal caso l'operazione viene effettuata anche se *source* e *destination* afferiscono a file system diversi. Il file ottenuto come *destination* è in questo caso rappresentato da nuovo i-nodo, i cui dati consistono nel path di *source*.

ln – esempi

Esempi di esecuzione

```
gio$ ln b d
gio$ ln -s c d/C
gio$ ls -liR
..
 30659 b
 30660 c
 30661 d
./d:
 30659 b
 30648 C
gio$ ls -l d
totale 4
-rw-r--r--  2 gio      staff    10 mar 25 22:11 b
lrwxrwxrwx  1 gio      staff     1 mar 26 20:00 C -> c
gio$
```



mv (move)

mv [*options*] *source... destination*

rinomina (sposta) file o directory.

Se vengono specificati solo i nomi di due elementi, *source* viene rinominato in *destination*, oppure in *destination/source*, a seconda che *destination* indichi un file o una directory. Qualora *destination* denoti un file preesistente, questo non sarà più accessibile come tale, e non sarà più accessibile in alcun modo se *destination* era il suo unico nome.

Se vengono indicati più di due elementi, *destination* deve essere una directory, e *source_1...source_n* vengono rinominati come *destination/source_1...destination/source_n*.

Nel caso che *source* e *destination* appartengono a due diversi file system, il comando effettua un vero e proprio spostamento dati tra i due file system. In tal caso vengono spostati solo i file ordinari, quindi: né collegamenti, né directory.

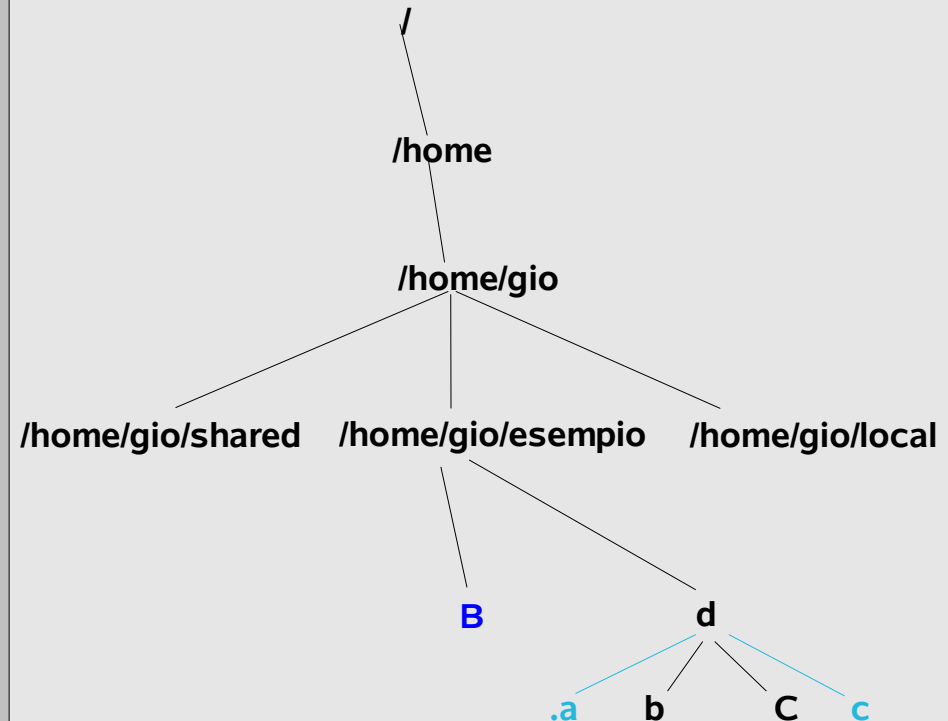
Alcune opzioni:

-i il comando chiede conferma all'utente qualora *destination* è un file preesistente

mv – esempi

Esempi di esecuzione

```
gio$ mv b B
gio$ mv .a c d
gio$ ls -a1iR
..
 30657 .
    4 ..
 30659 B
 30661 d
./d:
 30661 .
 30657 ..
 30658 .a
 30659 b
 30648 C
 30660 c
gio$
```



rm (remove)

```
rm [options] file...
```

```
rm -R [options] dirname... [file....]
```

elimina i file o le directory indicati come argomento.

Nella prima forma elimina uno o più file.

Nella seconda forma elimina ricorsivamente una o più directory con i loro contenuti; inoltre, opzionalmente elimina uno o più file

Alcune opzioni:

-i chiede conferma prima di rimuovere ogni file

Esempi d'uso

Elimina i file `pippo` nella directory corrente e `/tmp/pippo.back`

```
rm pippo /tmp/pippo.back
```

Elimina la directory `nuovadir` e tutto il suo contenuto, previa conferma

```
rm -Ri nuovadir
```

chown (change owner)

`chown [options] [user][:[group]] file...`

cambia proprietario e/o gruppo primario per uno o più file.

Se dopo `:` non segue il nome del gruppo, viene attribuito il gruppo principale a cui appartiene *user*.

Se prima di `:group` non viene indicato il nome dell'utente, viene cambiato solo il gruppo primario.

Alcune opzioni:

- R** opera ricorsivamente su directory e subdirectory
- h** se *file* è un link simbolico, opera su di esso piuttosto che sul file da esso referenziato

Nota:

Alcuni sistemi operativi (es. Solaris) hanno una opzione di configurazione per permettere i cambi di proprietà solo all'utente root.

chmod (change mode)

```
chmod [options] mode file...
```

cambia la modalità dei permessi sui file indicati come argomento

Il parametro *mode* può essere espresso sia in forma ottale che simbolica. In quest'ultimo caso, la stringa dei beneficiari ((**u**)ser,(**g**)roup,(**o**)ther) è separata da quella dei permessi ((**r**)ead,(**w**)rite,e(**x**)ecute) tramite i segni + o -, a seconda che i permessi vadano aggiunti o sottratti; mentre stringhe relative a beneficiari/permessi diversi sono separate da virgole, come in: **u+rw**x , **go+r**x

Alcune opzioni:

-R opera ricorsivamente su directory e subdirectory

Esempi d'uso

Attribuisce al file pippo il modo di permessi **rwxr-**x**r-**x****

```
chmod 755 pippo
```

Elimina i permessi **x per il gruppo ed **rx** per other per pippo**

```
chmod g-x,o-rx pippo
```

touch

`touch [options] [date_time] file...`

cambia sia il tempo di ultimo accesso che di ultimo aggiornamento dei file.

Se non viene specificato il parametro *date_time*, viene utilizzato il valore ottenuto dall'orologio del sistema nel momento in cui viene eseguito il comando.

Se si specificano file che non esistono, questi vengono creati vuoti.

Il parametro *date_time* deve avere il formato **MMDDhhmm** [YY]

Alcune opzioni:

- a cambia solo il tempo di ultimo accesso
- c se *file* non esiste non viene creato
- m cambia solo il tempo di ultima modifica

file

```
file [options] file...
```

analizza i file indicati come argomento e cerca di determinarne il tipo

Esempi di esecuzione

```
gio$ file multiboot.pdf
multiboot.pdf:  PDF (Portable Document Format) Adobe  v1.2
gio$ file "SCSI vs. EIDE.sxw"
SCSI vs. EIDE.sxw:      archivio ZIP
gio$
```

find

find [*pathname...*] [*expression*]

discende ricorsivamente le directory specificate (**pathname...**), cercando tutti i file che rendono vera **expression**.

Molto flessibile:

- **ricerca** file di specificati **attributi** (nome, tipo, permessi, proprietario, gruppo, numero di link, dimensione, data di ultima modifica/accesso ...)
- **and**, **or**, **not** di attributi
- può **eseguire** automaticamente, o previa conferma, uno o più **comandi** sui file individuati
- le espressioni si ottengono combinando flag, parametri e gli operatori booleani;
- le espressioni costituite solo da un flag e da un parametro (opzionale) si dicono espressioni elementari;

find - alcune espressioni elementari

Espressione	Risulta verificata se
-atime [+ <i>-</i>] <i>n</i>	l'accesso al file è avvenuto [più di, meno di] n giorni prima
-ctime [+ <i>-</i>] <i>n</i>	il modo del file è stato cambiato [più di, meno di] n giorni prima
-exec <i>command</i>	il comando restituisce 0 come stato di uscita
-fstype <i>type</i>	il filesystem cui il file appartiene è i tipo <i>type</i>
-links [+ <i>-</i>] <i>n</i>	il file ha [più di, meno di] n link
-mtime [+ <i>-</i>] <i>n</i>	i dati del file sono stati modificati [più di, meno di] n giorni prima
-name <i>pattern</i>	<i>pattern</i> coincide col nome del file
-newer <i>file</i>	il file corrente è stato modificato più recentemente di <i>file</i>
-perm <i>onum</i>	i permessi del file (in notazione ottale) coincidono con <i>onum</i>
-size [+ <i>-</i>] <i>n</i> [<i>c</i>]	la dimensione del file è [più di, meno di] n blocchi (512 byte) [byte]
-type <i>t</i>	il file è di tipo <i>t</i> . <i>t</i> può assumere i valori b(lock), d(irectory), p(lain file), etc.
-user <i>uname</i>	il file è di proprietà dell'utente <i>uname</i>

find – esempi

Ricerca in /home/gio di file la cui dimensione è maggiore di 100 blocchi

```
find /home/gio -size +100
```

Come prima, ma adesso la dimensione dei file è espressa in byte

```
find /home/gio -size +100c
```

Ricerca dei file modificati meno di due giorni fa e che non sono di root

```
find / -mtime -2 -a ! -user root
```

Rimozione di file il cui ultimo accesso risale a più di 7 giorni

```
find /home/gio -atime +7 -exec rm {} \;
```

Rimozione di file dell'utente gio e che hanno estensione .old oppure .back

```
find . \ ( '*.old' -o '*.back' \ ) -user gio -ok rm {} \;
```

tar (tape archive)

```
tar c [options] [f tar_file] [-C directory] file..
```

```
tar x [options] [f tar_file] [file..]
```

crea archivi, vi aggiunge o vi estrae file.

Nella prima forma crea un archivio dei file passati come argomento sul dispositivo di default, oppure – se è presente l'opzione **f** – nel file *tar_file*. Se *file* è una directory, viene archiviato tutto il suo contenuto. Se è presente l'opzione -C, il comando effettua dapprima il cambio di directory in *directory*, eppoi archivia *file*. Se *file* è uguale a “.” viene archiviato tutto il contenuto della directory in cui opera tar.

Nella seconda forma, estrae i file passati come argomento dall'archivio di default, oppure - se è presente l'opzione **f** – dal file *tar_file*. Se *file* è omesso, viene estratto tutto il contenuto dell'archivio.

Esistono altre forme del comando, rispettivamente per sostituire file in un archivio (**tar r**), aggiornare un archivio (**tar u**) e mostrare il (o parte del) contenuto di un archivio (**tar t**).

Alcune opzioni:

-v elenca i file che vengono elaborati

tar - esempi

Esempi d'uso

Archivia il contenuto della directory corrente sul dispositivo /dev/rmt/0

```
tar cf /dev/rmt/0 .
```

Archivia nel file docs.tar il contenuto della directory /home/gio/docs

```
tar cf docs.tar -C /home/gio/docs .
```

Visiona il contenuto dell'archivio docs.tar

```
tar tf docs.tar
```

Estrae il file y2004/jan.pdf dall'archivio docs.tar

```
tar xf docs.tar y2004/jan.pdf
```

gzip, gunzip

```
gzip [options] [file...]
```

```
gunzip [options] [file...]
```

vengono usati , rispettivamente, per comprimere e decomprimere i file indicati negli argomenti.

gunzip può attualmente decomprimere file compressi creati con **zip**, **compress** e **pack**, oltre naturalmente a quelli generati da **gzip**.

Alcune opzioni:

- h** mostra una breve documentazione sul comando
- l** per ogni file compresso, mostra informazioni circa il fattore di compressione, la dimensione del file compresso e non compresso, il nome del file non compresso, ecc.
- r** se *file* è una directory opera ricorsivamente su ogni file e subdirectory in esso contenuti

Comandi di utilità

cat	Concatena file
wc	Conta caratteri, parole e linee in un file
cut	Taglia delle colonne opportune da un file di testo
paste	Compone più file di testo
sort	Ordina le linee di un file
diff	Mostra le differenze di contenuto tra due file
cmp	Confronta due file byte per byte
grep	Cerca espressioni regolari all'interno di un file
passwd	Cambia o visualizza la password di un utente
printenv	Mostra il contenuto delle variabili d'ambiente

cat (concatenate)

```
cat [options] [file...]
```

concatena i file indicati come argomento, visualizzandoli attraverso lo standard output

Alcune opzioni:

- n fa precedere ogni linea di output dal numero progressivo che identifica la posizione della linea nel file concatenato
- b come l'opzione precedente, ma omette la numerazione delle linee bianche
- v mostra anche i caratteri non stampabili, ad eccezione dei caratteri di tabulazione, nuova linea e ritorno a capo

wc (word count)

```
wc [options] [file...]
```

fornisce il numero dei codici di interruzione di riga (in pratica il numero delle righe), delle parole o dei caratteri contenuti in *file*. Senza opzioni fornisce, nell'ordine suddetto, ciascuna delle precedenti informazioni.

Alcune opzioni:

- c emette solo il numero complessivo di caratteri di *file*.
- w emette solo il numero complessivo di parole in *file*.
- l emette solo il numero di righe in *file*.

Esempi di esecuzione

```
gio$ wc which_manpage
132    239    2083  which_manpage
gio$ wc -c which_manpage
2083  which_manpage
gio$
```

cut

```
cut [options] [file...]
```

estrae delle colonne specifiche dalle linee di testo che compongono *file*.

Alcune opzioni:

- c *char_list*** definisce gli intervalli da estrarre espressi in caratteri.
- f *field_list*** definisce gli intervalli da estrarre espressi in campi. I campi sono distinti in base a un certo carattere usato come delimitatore. Quello predefinito è il carattere di tabulazione.
- d *delimiter*** definisce un delimitatore alternativo al carattere di tabulazione.

Esempi d'uso

Estrae la prima colonna del file `/etc/passwd`

```
cut -d: -f1 /etc/passwd
```

Estrae i primi dieci caratteri da ogni riga del file `/etc/passwd`

```
cut -c1-10 /etc/passwd
```

paste

`paste [options] file...`

concatena le righe corrispondenti dei file inseriti come argomento, inserendo tra di esse un delimitatore. Il delimitatore di default è il carattere di tabulazione.

Alcune opzioni:

`-d delimiter` definisce un delimitatore alternativo al carattere di tabulazione.

Esempi d'uso

Concatena ciascuna riga di `pippo` con la corrispondente di `prova`

```
paste pippo prova
```

Come prima, ma in luogo di `<TAB>` è inserito “:” come delimitatore

```
paste -d: pippo prova
```

sort

`sort [options] [file...]`

permette di (ri)ordinare o fondere insieme il contenuto dei file passati come parametri, oppure di (ri)ordinare le linee passategli in input. L'ordinamento è la modalita predefinita, ed è effettuato tenendo conto delle opzioni inserite dall'utente. Il comando tratta ogni linea come un insieme ordinato di campi, separati da opportuni caratteri (di default i caratteri di separazione sono la tabulazione e lo spazio). In assenza di opzioni che definiscano diversi criteri di ordinamento, quest'ultimo avviene in base al primo campo ed è alfabetico.

Alcune opzioni:

- `-f` ignora le differenze tra lettere minuscole e maiuscole
- `-n` considera numerica anzichè testuale la chiave di ordinamento
- `-r` ordina in senso decrescente anzichè crescente
- `-o fileout` invia l'output a fileout anzichè sull'output standard
- `-t s` usa s come separatore di campo
- `-k s1, s2` usa i campi da s1 a s2-1 come chiavi di ordinamento

sort - esempi

Esempi d'uso

Ordina le linee del file /etc/passwd in base al valore del terzo campo (UID)

```
sort -t: -k3,4 /etc/passwd
```

Come prima, solo che ora l'ordinamento è numerico anzichè alfabetico

```
sort -t: -n -k3,4 /etc/passwd
```

Come prima, ma seguendo l'ordinamento inverso (prima l'UID maggiore)

```
sort -t: -n -k3,4 -r /etc/passwd
```

Come prima, ma ora l'output è memorizzato in passwd_reordered

```
sort -t: -n -k3,4 -r /etc/passwd -o passwd_reordered
```

diff (differences)

```
diff [options] file_1 file_2
```

può funzionare con diverse modalità, stabilite in base alle opzioni, per determinare semplicemente i due file passati come parametri sono identici o meno, oppure per indicare le differenze che ci sono tra i due, con maggiore o minore dettaglio di informazioni al riguardo. Il risultato del confronto dei file viene emesso attraverso lo standard output, e di default mostra la lista di cambiamenti da apportare a *file_1* per renderlo uguale a *file_2*.

Alcune opzioni:

- l produce l'output in formato esteso
- s segnala se i due file sono identici (per default non è emesso output in tal caso)
- b ignora i “caratteri bianchi”, quali TAB e spazi, e considera stringhe di tali caratteri come equivalenti
- i considera equivalenti le lettere maiuscole e minuscole
- r opera ricorsivamente su directory e subdirectory

cmp (compare)

```
cmp [options] file_1 file_2 [skip_1] [skip_2]
```

confronta i due file passati come parametri, scrivendo sullo standard output solo qualora i due file presentano delle differenze. A meno che le opzioni specifichino diversamente, scrive sullo stdout il byte ed il numero di linea in cui occorre la prima differenza. I byte e le linee sono numerati a partire da 1. I parametri *skip_1* e *skip_2* sono byte offset iniziali relativi, rispettivamente, a *file_1* e *file_2*. possono essere espressi sia in forma decimale che ottale, ed in quest'ultimo caso va utilizzato "0" come prima cifra.

Alcune opzioni:

- l scrive il numero di linea (in decimale) ed il byte differente (in ottale) per ogni differenza riscontrata;
- s non scrive nulla nel caso di file differenti, restituendo solo lo stato di uscita.

grep (get regular expression)

```
grep [options] modello [file...]
```

```
grep [options] -e modello [file...]
```

```
grep [options] -f file_modello [file...]
```

esegue una ricerca all'interno dei file indicati come argomento oppure, nel caso che non sia specificato alcuno di tali file, all'interno dello standard input.

Il modello di ricerca può essere semplicemente il primo degli argomenti che seguono le opzioni (prima versione del comando), oppure può essere indicato precisamente come argomento dell'opzione `-e`, oppure ancora può essere contenuto in un file che viene indicato attraverso l'opzione `-f`.

Le opzioni *options* sono relative a modalità di visualizzazione dell'output e degli eventuali messaggi di errore.

Alcune opzioni

`-v` restituisce le linee che non contengono *modello*

`-w` restituisce solo le linee che hanno *modello* come parola completa

`-x` restituisce solo le linee che coincidono con *modello*

grep (get regular expression)

Metacarattere	Significato
^	Inizio della linea
\$	Fine della linea
\<	Inizio di una parola
\>	Fine di una parola
.	Un singolo carattere qualsiasi
[<i>stringa</i>]	Un qualunque carattere in stringa
[~ <i>stringa</i>]	Un qualunque carattere non in stringa
[<i>c0-c1</i>]	Un qualunque carattere compreso tra il carattere <i>c0</i> ed il carattere <i>c1</i>
\	Inibisce l'interpretazione del metacarattere seguente

Nota:

Per evitare interpretazioni improprie da parte della shell è opportuno racchiudere l'espressione che definisce il modello di ricerca tra gli apici ' ' qualora esso contenga uno dei metacaratteri mostrati in tabella.

grep – esempi

Esempi d'uso

Fornisce le linee del file `/etc/passwd` che contengono la parola `gio`

```
grep -w gio /etc/passwd
```

Come prima, ma ora sono restituite tutte le linee che hanno `gio` come (sub) stringa

```
grep gio /etc/passwd
```

Restituisce le linee che finiscono con la lettera `f`

```
grep /home/gio/pippo f$
```

Restituisce tutti i file nella directory corrente che non sono directory e che hanno il permesso in esecuzione per il proprietario

```
ls -al | grep ^-..x
```

passwd (password)

passwd [*options*] [*name*]

modifica o mostra la password od i suoi attributi per l'utente corrente, oppure per l'utente il cui nome di login è *name*.

Solo un utente con privilegi speciali (ad es. **root**) può cambiare la password relativa ad utenti diversi e gli attributi di questa.

Le regole relative agli attributi di una password (lunghezza, periodo di validità, ecc.) sono stabilite in base ad opportuni file di configurazione, e sono sotto l'esclusivo controllo di utenti muniti di credenziali opportune (es. **root**).

La scelta di una nuova password non conforme alle suddette regole provoca il rigetto della password e la richiesta da parte del programma di una password adeguata.

Una volta lanciato, il programma richiede l'inserimento della vecchia password (se definita) ed in caso tale inserimento sia corretto, l'inserimento per due volte della nuova password.

printenv (print environment)

```
printenv [env_var]
```

emette attraverso lo standard output il contenuto della variabile d'ambiente indicata come argomento. Se viene usato senza argomento, emette il contenuto di tutte le variabili d'ambiente

Esempi d'uso

Mostra il valore della variabile d'ambiente indicante la shell di login

```
printenv SHELL
```

Mostra il valore di tutte le variabili d'ambiente

```
printenv
```

Sviluppo software

`cc` (`gcc`)

Consente di compilare ed effettuare il linking per un codice scritto in C

`make` (`gmake`)

Automatizza la manutenzione e l'aggiornamento di programmi

Creazione di un eseguibile in C

Creazione di un file sorgente:

con un editor di testo (ad es. vi), viene creato un file ascii contenente le istruzioni in linguaggio C e per il pre-processore C.

Pre-elaborazione:

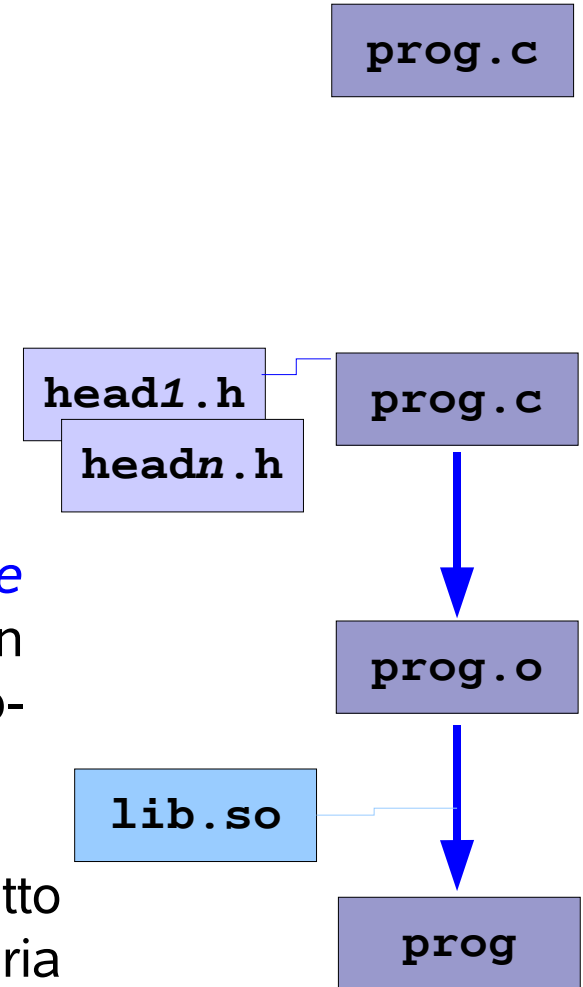
il pre-processore C esegue le istruzioni che gli competono scritte nel file sorgente.

Compilazione del programma:

il compilatore crea un file binario *rilocabile* (*codice oggetto*), in cui gli indirizzi di memoria sono definiti in termini relativi, per differenza rispetto ad un indirizzo-base (*offset*).

Creazione dell'eseguibile:

il linker esegue il collegamento del codice oggetto ottenuto alla fase precedente con le funzioni di libreria richieste, creando un codice binario con indirizzi di memoria assoluti (*codice assoluto*).



Creazione di un eseguibile in C

Creazione di un file sorgente:

E' importante utilizzare un **editor di testo** e non un word processor. Esempi di editor di testo sotto Unix sono **vi** (e la sua variante **vim**), **emacs**, **gedit**, ...

Esempio:

```
vi prog.c
```

Pre-elaborazione:

Le istruzioni del pre-processore cominciano con “#”; servono ad esempio ad includere gli **header**.

Esempio:

```
#include<head1.c>
```

Compilazione del programma:

Il **compilatore C** è spesso richiamato col comando **cc** seguito dall'opzione “-c”. Lo stesso comando permette di lanciare il **linker**. Se si omette “-c” il comando esegue entrambe le fasi.

Esempio:

```
cc -c prog.c
```

Creazione dell'eseguibile:

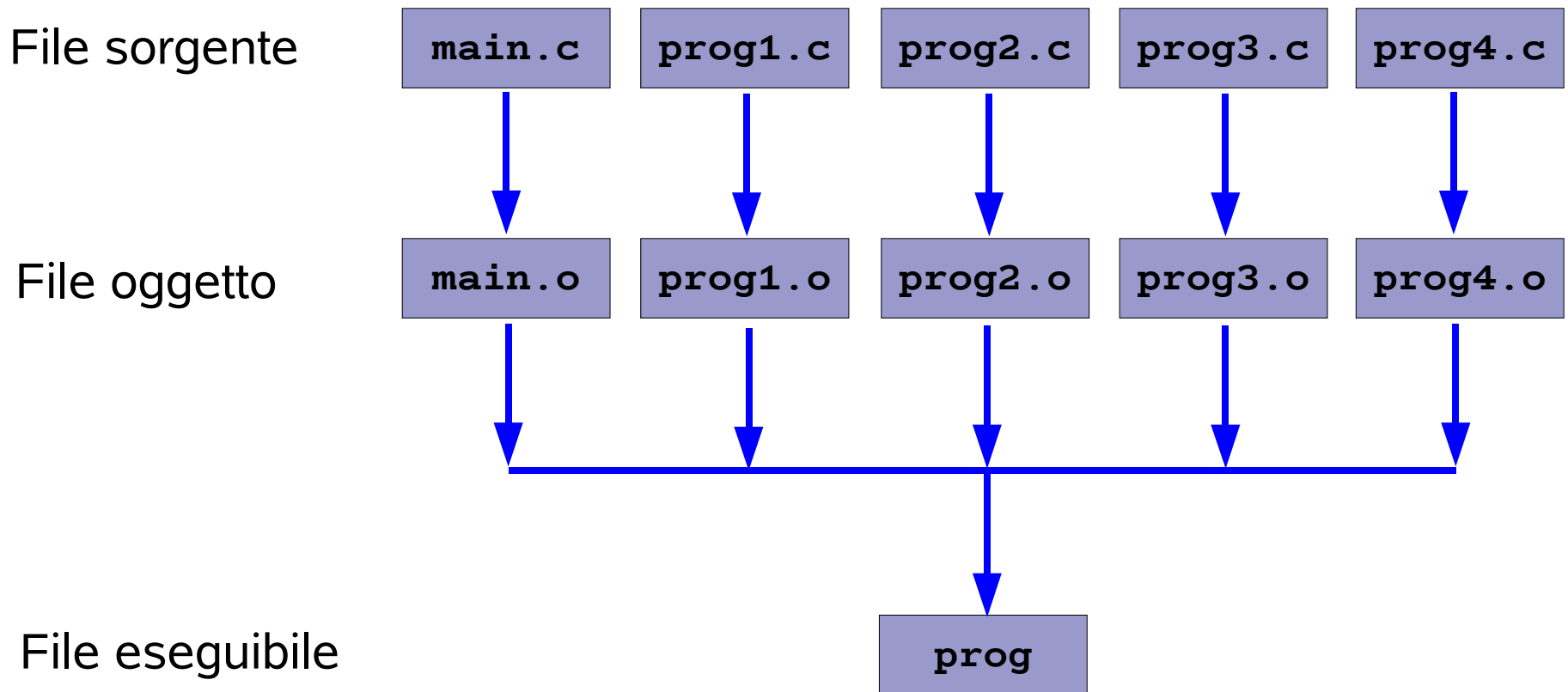
L'opzione “-o” serve a definire per il codice assoluto un nome diverso da quello di default, che è **a.out**.

Esempio:

```
cc -o prog prog.o
```

Creazione di un eseguibile in C

La creazione di un eseguibile in genere comporta la compilazione ed il linking di diversi file sorgente:



Creazione di un eseguibile in C

Supponiamo che:

- `main.o` non esista;
- `prog1.o` e `prog4.o` siano antecedenti all'ultima versione di `prog1.c` e `prog4.c`;
- `prog2.o` e `prog3.o` siano relativi all'ultima versione di `prog2.c` e `prog3.c`;

Le operazioni di aggiornamento necessarie sono le seguenti:

```
cc -c main.c
```

```
cc -c prog1.c
```

```
cc -c prog4.c
```

```
cc -o prog main.o prog1.o prog2.o prog3.o prog4.o
```

Creazione di un eseguibile in C

Se i sorgenti sono frequentemente modificati è necessario ripetutamente:

- controllare se qualche file sorgente è stato modificato;
- ricompilare gli eventuali file modificati;
- ricostruire l'eseguibile.

Il comando **make** consente la manutenzione e l'aggiornamento di programmi, automatizzando le precedenti operazioni

Per svolgere le sue funzioni il comando **make** ha bisogno di un file ausiliario, il **makefile**

make

```
make [-f makefile] [options] target....
```

Usa le istruzioni contenute in un file di testo detto *makefile* - che di default è il file di nome `{M|m}akefile` contenuto nella directory corrente - e le informazioni relative alla data di ultima modifica dei file, per aggiornare *target*. Se *target* è omesso viene previsto l'aggiornamento di tutti i target definiti nel *makefile*.

Alcune opzioni

- `-n` visualizza i comandi che verrebbero eseguiti in base alle istruzioni nel *makefile*, ma non li esegue
- `-p` visualizza il contenuto del *makefile* e procede all'esecuzione delle istruzioni ivi contenute

Esempi d'uso:

*Usa il *makefile* di nome {M|m}akefile nella directory corrente:*

```
make
```

*Usa come *makefile* il file /home/gio/mio_make per aggiornare programma*

```
make -f /home/gio/mio_make programma
```

Il makefile

Un makefile si compone di:

- linee di dipendenza;
- linee di comando.

Una **linea di dipendenza** ha una sintassi del tipo:

```
target: file1 file2 file3... filen
```

essa definisce da quali file dipende un altro file, detto **target**, che costituisce l'obiettivo che si deve aggiornare.

Una **linea di comando** o regola, definisce le operazioni che **make** deve effettuare per passare dai **file di dipendenza** al **target**.

Il makefile – esempio 1

Esempio di makefile relativo alla creazione del programma prog:

```
#makefile per la compilazione di prog - vers.1

prog:    main.o prog1.o prog2.o prog3.o prog4.o
        cc -o prog main.o prog1.o prog2.o prog3.o prog4.o

main.o:  main.c
        cc -c main.c

prog1.o: prog1.c
        cc -c prog1.c

prog2.o: prog2.c
        cc -c prog2.c

prog3.o: prog3.c
        cc -c prog3.c

prog4.o: prog4.c
        cc -c prog4.c
```

Il makefile

Lanciando **make** si ottiene la potenziale esecuzione delle linee di comando del **makefile** associate a tutti i target in esso definiti, oppure al target che è stato dato in input come parametro a **make**.

Le suddette linee di comando sono eseguite solo qualora il **target** sia stato modificato più recentemente di almeno uno dei **file dipendenti** associati

Se i **file di dipendenza** sono da aggiornare oppure non esistono, vengono eseguite anche le linee di comando necessarie al loro aggiornamento o alla loro creazione.

Non è necessario specificare tutte le dipendenze e le regole, perché **make** è in grado di riconoscere le relazioni esistenti tra alcuni file e di applicare ad essi regole predefinite.

Il makefile – esempio 2

Ad esempio, **make** riconosce i file con il suffisso **.c** come sorgenti in linguaggio C e fa eseguire, se necessario, la loro compilazione

Esempio di makefile relativo alla creazione del programma prog:

```
#makefile per la compilazione di prog - vers.2

prog:    main.o prog1.o prog2.o prog3.o prog4.o
        cc -o prog main.o prog1.o prog2.o prog3.o prog4.o

main.o:  main.c

prog1.o: prog1.c

prog2.o: prog2.c

prog3.o: prog3.c

prog4.o: prog4.c
```

Il makefile – esempio 3

Ad esempio, se uno dei file oggetto specificati e` da aggiornare, **make** ricerca nella **directory corrente** un file sorgente con lo stesso nome e fa eseguire la compilazione

Esempio di makefile relativo alla creazione del programma prog:

```
#makefile per la compilazione di prog - vers.3

prog:    main.o prog1.o prog2.o prog3.o prog4.o
         cc -o prog main.o prog1.o prog2.o prog3.o prog4.o
```

Il makefile – esempio 4

E` possibile rendere parametrici i makefile utilizzando delle **macro**.

Le macro si usano per definire elenchi di file, opzioni dei compilatori, librerie, comandi, etc...

Esempio di makefile relativo alla creazione del programma prog:

```
#makefile per la compilazione di prog - vers.4

OGGETTI: main.o prog1.o prog2.o prog3.o prog4.o

prog: $(OGGETTI)
    cc -o prog $(OGGETTI)
```

Riferimenti

Sun Microsystem – *Solaris 10 Reference Manual Collection*

<http://docs.sun.com/app/docs/coll/40.10>

D. Giacomini – *Appunti di Informatica Libera*

<http://a2.swlibero.org/a2.html>