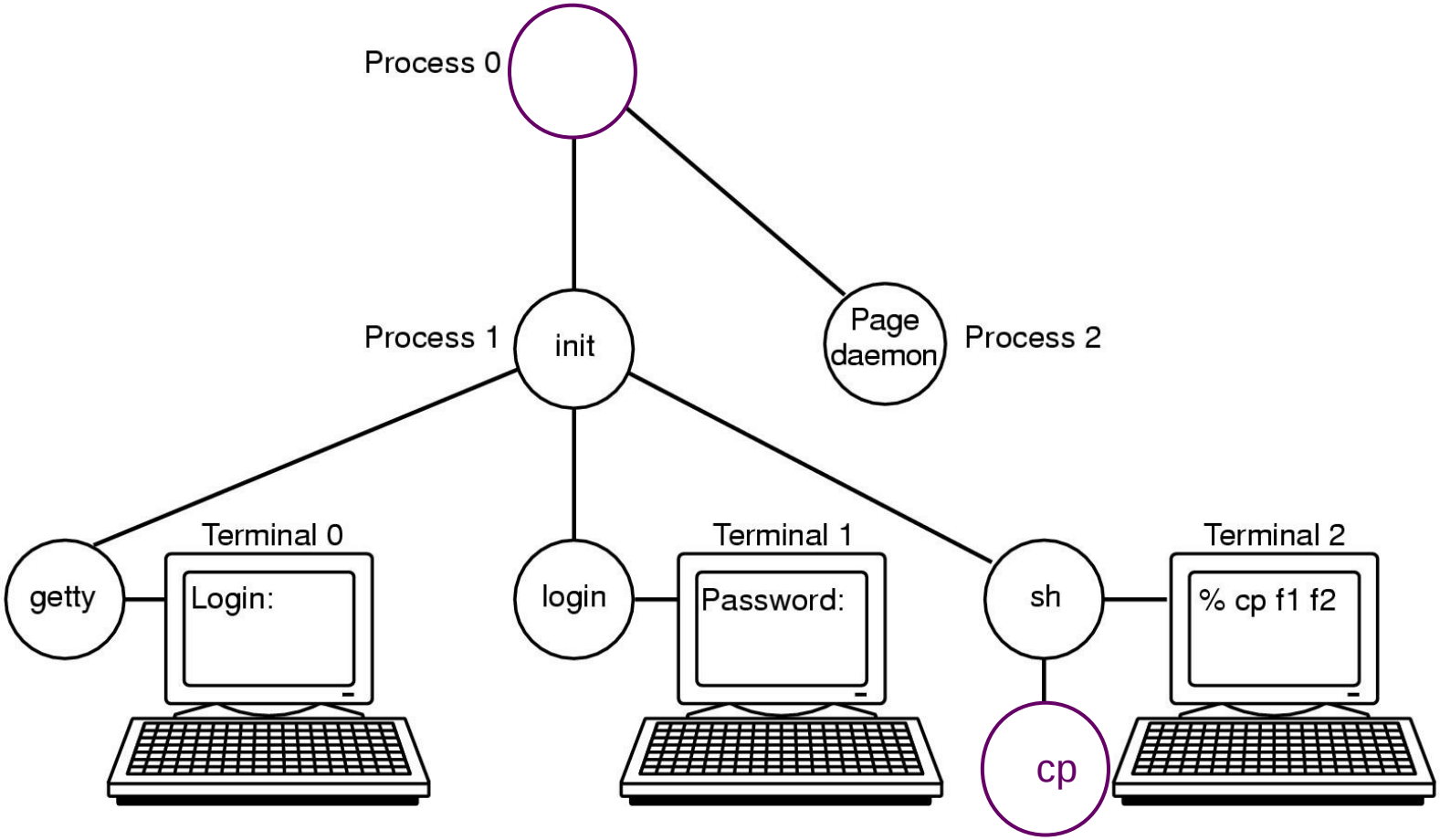
Avvio di un sistema Linux

- Quando si accende il calcolatore, si verifica la seguente sequenza di eventi:
  - L'hardware esegue tests di diagnostica.
  - Il kernel di Linux viene caricato dal device di root.
    - BIOS cerca il master boot record (MBR) nell'hard disk.
    - L'indirizzo di LILO (LInux LOader) è nel MBR, e permette di scegliere e caricare il sistema operativo. La maggior parte di LILO sta in /boot/boot.b e la tabella con l'indirizzo del sistema operativo sta in /boot/map. La configurazione di LILO sta in /etc/lilo.conf.
  - Il kernel comincia a girare e inizializza le risorse.
  - Il kernel fa partire init, il processo da cui discendono tutti i processi utenti. I processi padri e figli sono disposti secondo una gerarchia, nella quale init è il processo padre di tutti i processi.

Avvio di un sistema Linux

- Il processo init esegue alcune azioni:
  - Controlla il file system, esegue i programmi di routing per la rete, rimuove i files temporanei …
  - Quindi genera un processo chiamato getty, che è responsabile della gestione dei login dei vari utenti.
  - Il processo getty scrive “login: ” sullo schermo e sostituisce se stesso con il processo login se si accorge che qualcuno sta cercano di fare log in.
  - login accetta uno user id e una password, li controlla con quelli contenuti nel file /etc/passwd (o /etc/shadow), e sostituisce se stesso col programma di avvio dell’utente se la password è corretta.
  - Il programma di avvio è di solito una shell (bash).
  - Eventualmente compare il messaggio del giorno scritto dall’amministratore di sistema (è contenuto in /etc/motd).

Avvio di un sistema Linux



La sequenza dei processi durante l'avvio.

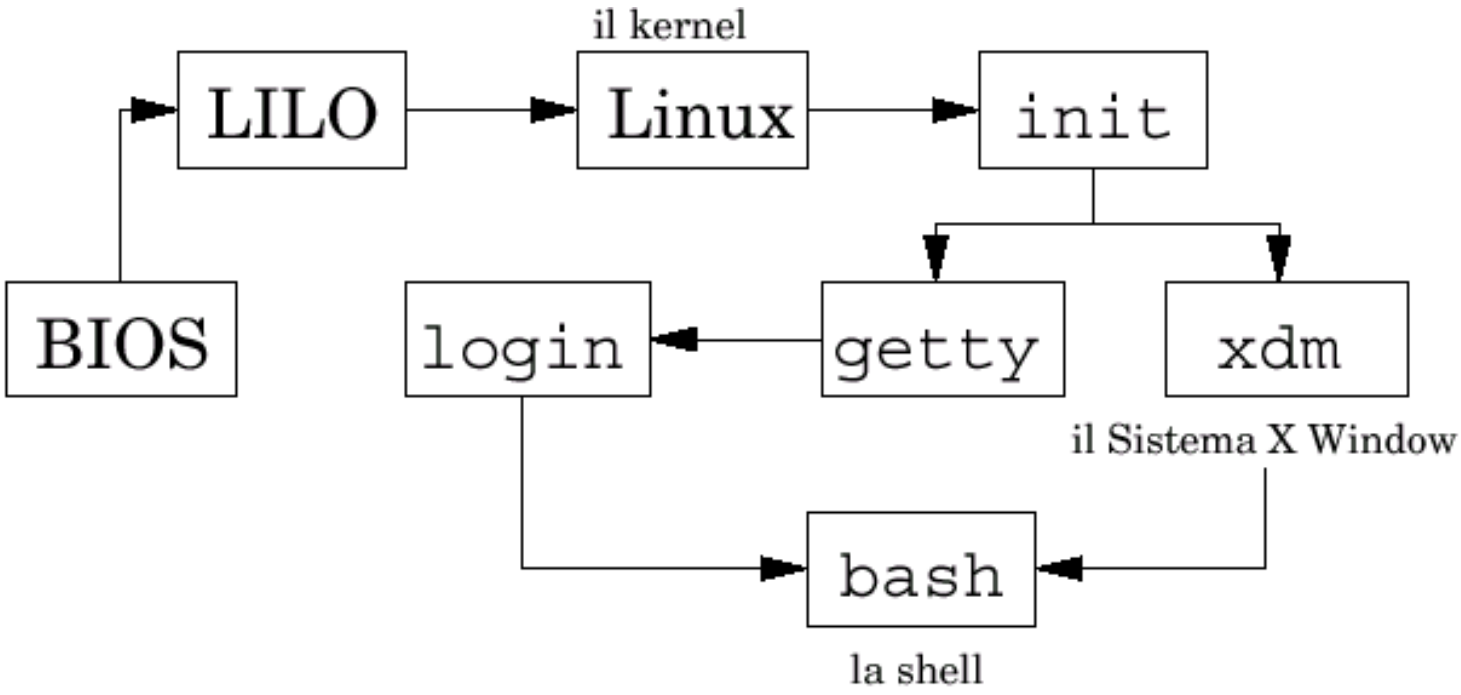
Avvio di un sistema Linux

```
login: lferrari
Password:
Last login: Fri Mar 21 17:34:49 from
  paperino.capitano.unisi.it
[lferrari@homelinux lferrari]$
```

- Per uscire dal sistema, digitare una delle seguenti sequenze di tasti:
  - `[lferrari@homelinux lferrari]$ logout`
  - `[lferrari@homelinux lferrari]$ ^D`

Avvio di un sistema Linux

- Il processo init può avviare il sistema X Window (utilizzando KDE o GNOME) invece di un'interfaccia di testo. In questo caso il processo init crea il processo xdm invece del processo getty, ma poi la procedura per l'utente è molto simile alla precedente.



Spegnimento di un sistema Linux

- Ogni RAM non utilizzata dal kernel o da applicazioni è utilizzata automaticamente come **cache di disco**. Questo velocizza l'accesso al disco finchè c'è memoria disponibile. La cache di disco e il disco vengono sincronizzati ogni 30 secondi circa.
- Ciò significa che dobbiamo utilizzare una procedura che termini il processo della cache di disco prima di spegnere il PC.
- Quindi, non si dovrebbe uscire da Linux semplicemente premendo il bottone POWER sul cabinet del PC. Ciò può danneggiare seriamente il file system. Utilizzare sempre i comandi halt o shutdown.
  - **`shutdown –h 20:00`**
  - **`shutdown –r now`**

Spegnimento di un sistema Linux

- Per fare ciò, occorre accedere con un account speciale, chiamato **root**. L'account root è l'account dell'amministratore di sistema e può accedere ad ogni file del sistema. Se si deve spegnere il calcolatore, occorre utilizzare la password di root. Effettuare il login come root:

```
login: root
Password:
[root@homelinux root]$ shutdown –h now
```

o

```
[lferrari@homelinux lferrari]$ su
Password:
[root@homelinux root]$ shutdown –h now
```

La shell

- Il prompt $ che si vede al primo login è visualizzato utilizzando una shell (di solito bash è la shell di default di Linux).
- La Shell esegue programmi.
  - L'utente digita i comandi
  - La shell legge i comandi e li traduce al sistema operativo.
- `[lferrari@homelinux lferrari]$ date`

```
dom mar 16 18:41:44 CET 2003
```

- La scritta `[username@server current_directory] $` significa che questa console è attualmente in uso da parte dell'utente `username`, e l'host-name è `server`. La seconda parola è l'attuale directory in cui si sta lavorando, in questo caso /home/lferrari, che è la 'home' directory di default dell'utente `lferrari` nel sistema Linux in oggetto. Da qui in avanti il prompt sarà '$ _'.

Comandi

- Formato generale della sintassi dei comandi:

  $ command -options arguments

- Esempio:
  - `$ clear`
  - `$ cd /usr/src/linux`
  - `$ wc -w file1` (numero di parole in file1)
  - `$ wc -c file1` (numero di caratteri in file1)
  - `$ wc -l file1` (numero di linee in file1)
  - `$ cat file1 file2 file3`
  - `$ ls –al`

Wild-cards

- `$ ls –l *.c`
- `$ ls [abc]*`
- `$ ls ?a*`

- ***** significa “fai la ricerca di stringhe con un numero qualsiasi di caratteri”.
  - Ad esempio, chap* trova: *chap01, chapa, chap_end*, ed anche *chap*.
  - Se si richiede soltanto * (nient’altro), trova tutti i files.
- ? significa “fai la ricerca di stringhe di lunghezza 1, cioè contenenti un solo, arbitrario carattere”.
  - Ad esempio, chap? trova: *chapa* e *chap1*, ma non *chap01* e *chap*.

Wild-cards

- [..] significa “fai la ricerca di stringhe contenenti uno qualsiasi dei caratteri indicati tra parentesi”. Un intervallo di caratteri può essere specificato separando una coppia di caratteri con un trattino (dash).
  - Ad esempio, chap[abc] trova: *chapa* e *chapc*, ma non *chap1* e *chapab*.
  - [A-Za-z]* trova tutte le parole il cui primo elemento è un carattere.
  - [!abc] trova tutti i caratteri diversi da a,b e c.

Problemi più comuni

- Se un comando provoca un messaggio di errore del tipo:
  - `-bash: <command>: Command not found`

  allora le prime cose da verificare sono le seguenti:

1. Hai digitato il comando correttamente?
   - RICORDA: LINUX è CASE SENSITIVE. Maiuscole e minuscole vengono valutate in maniera diversa.
- Esempi di errori relativi al comando `cat`:
  - `$ ct ; $ Cat ; $ CAT`

2. Controllare se il file eseguibile del comando è stato installato usando il comando `whereis`.
   - A titolo di esempio, applichiamo whereis per il comando useradd:

```
$ whereis useradd
bash: useradd: /usr/sbin/useradd
/usr/share/man/man8/useradd.8.bz2
```

   - Dunque il comando c'è, ma non nella directory giusta. Si può eseguire tale comando digitando:

```
$ /usr/sbin/useradd
```

Problemi più comuni

3. Se il precedente passo 2 (cioè, `$ /usr/sbin/useradd`) non funziona, allora forse il comando potrebbe richiedere l'esecuzione da root.
4. Se il passo non dà risultati,

   ```
   $ latex
   -bash: latex: command not found
   $ whereis latex
   latex:
   ```

   allora occorre installare il pacchetto. Inserire il CD di Linux, e installare il pacchetto col comando **`rpm.`**

   - Ci sono parecchi strumenti grafici per l'installazione dei pacchetti (rpmdrake, Kpackage, Gnome RPM, …).

Filtri

- ```
  $ cat
  Help! I'm stuck in a Linux program!
  Help! I'm stuck in a Linux program!
  ^D
  ```

(Il testo in rosso indica quello che digita l'utente).

- ```
  $ sort
  ugo
  pino
  mario
  angelo
  luca ^D
  angelo
  luca
  mario
  pino
  ugo
  ```

Caratteri di controllo

- Interrupt
  - **^C** interrompe. Esce dal programma e fa tornare al prompt.
  - **^Z** sospende. Ferma il programma e lo mette in pausa. Digitare *fg* per riprendere l'esecuzione del programma.
  - **^D** fine del file. Se digitato nel prompt, fa uscire dalla shell e porta l'utente dove era prima di iniziare a lavorare con la shell.

- Stop/Start Scrolling: se ciò che si desidera vedere va a finire troppo in basso, uscendo così dallo schermo, è possibile evitare ciò mandando un segnale di "stop" dello scorrimento (**^S**). Mandare un segnale di "start" per riprendere lo scorrimento (**^Q**).
  - **^S** fermare lo scorrimento
  - **^Q** riprendere lo scorrimento

Manuale di aiuto on-line di Linux

- Il comando man mostra il manuale di aiuto per il comando selezionato:

man command_name

- `$ man man` o `$ man –help` aiuto per il comando man stesso.

Manuale di aiuto di Linux

- Sezioni del manuale
  1. Comandi Utente.
  2. Chiamate di Sistema.
  3. Librerie dei linguaggi.
  4. Device e Interfacce di rete.
  5. Formati dei File.
  6. Giochi e Demos.
  7. Tabelle di ambiente e Macros.
  8. Manutenzione del Sistema.

- Esempio:
  - man kill (= man 1 kill)
  - man 2 kill
  - man 3f kill

- Ogni capitolo contiene alcune pagine introduttive:
  - man intro (*introduzione al manuale*)
  - man 1 intro (*introduzione al capitolo 1*)
  - man 3f intro

Esempio: il comando cat

[lferrari@homelinux lferrari]$ man cat

CAT(1) User Commands CAT(1)

NAME
cat - concatenate files and print on the standard output

SYNOPSIS
cat [OPTION] [FILE]...

DESCRIPTION
Concatenate FILE(s), or standard input, to standard output.

-A, --show-all
equivalent to -vET

-b, --number-nonblank
number nonblank output lines

-e equivalent to -vE

-E, --show-ends
display $ at end of each line

-n, --number
number all output lines

-s, --squeeze-blank
never more than one single blank line

-t equivalent to -vT

-T, --show-tabs
display TAB characters as ^I

-u (ignored)

-v, --show-nonprinting
use ^ and M- notation, except for LFD and TAB

--help display this help and exit

--version
output version information and exit

With no FILE, or when FILE is -, read standard input.

AUTHOR

Written by Torbjorn Granlund and Richard M. Stallman.

REPORTING BUGS

Report bugs to <bug-textutils@gnu.org>.

COPYRIGHT

Copyright (C) 2002 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

The full documentation for cat is maintained as a Texinfo manual. If the info and cat programs are properly installed at your site, the command *info cat* should give you access to the complete manual.

Opzioni

- `$ man –f` *`command-name`* dà solo una breve descrizione del comando.
  - man –f cat
- `$ whatis` *`command_name`* analogo.
  - whatis cat

  `man –f` e `whatis` mostrano tutte le occorrenze del comando cercato nei diversi capitoli.
- `$ man -k` *`keyword`* mostra una lista di comandi nella cui descrizione c'è la parola "keyword".
  - man -k cat
  - man -k manual

Manuale di Linux

- info è un programma per leggere la documentazione sulle utilities GNU.
  - `$ info info` (mostra un'introduzione a info).
  - `$ info emacs` (descrive come editare testi con Emacs).
  - `$ info bash` (fornisce una breve descrizione della shell Bash).
  - `$ info uname` (il comando uname stampa informazioni sulla macchina e sul sistema operativo che gira su di essa).
    - `$ uname –a`

```
Linux homelinux 2.4.19-16mdk #1 Fri
Sep 20 18:15:05 CEST 2002 i686
unknown unknown GNU/Linux
```

Working directory

- Ogni processo è localizzato in una directory, chiamata la sua working directory.
- Quando si accede a un sistema Linux, la shell si avvia in una particolare directory, detta home directory (es. /home/lferrari/).
- Per visualizzare la propria working directory usare
  - `$ pwd`
- Per tornare alla home usare
  - `$ cd`
- Ciascun file o directory ha un nome non ambiguo specificato dal suo pathname relativo a ***/***, la directory root. Un pathname relativo alla root viene spesso chiamato un pathname assoluto (es. /usr/src/linux/CREDITS).
  - `$ cd /usr/src/linux/CREDITS`

Pathname assoluti e relativi

- Un file può essere specificato in modo non ambiguo usando un pathname relativo alla sua working directory.
  - Esempio: se /usr/src/ è la working directory, /linux/CREDITS è il pathname relativo di CREDITS.
- Il file system fornisce i seguenti campi speciali che possono essere usati all'interno di un path relativo:
  - `.` - current directory
  - `..` - parent directory
  - `~` - home directory
  - Esempio:
    - `cd .`
    - `cd ..`
    - `cd pippo`
    - `cd ./pippo`

Elencare i contenuti di una directory

- `$ ls` elenca i contenuti della working directory.
- `$ ls dir_name` elenca i contenuti di `dir_name`.
- ls: opzioni
  - -a tutti i files, compresi quelli nascosti
    -F aggiunge / per le directories, * per gli eseguibili, @ per i link simbolici
  - -l formato completo – dettagli per i files
  - -m elenca i files come se fossero un testo scritto, separandoli con la virgola
  - -r inverte l'ordine alfabetico
  - -R ricorsivo; comprende le sottodirectories
  - -s dimensioni dei files in blocchi
  - -t elenca secondo la data dell'ultima modifica
  - -u elenca secondo la data dell'ultimo accesso
  - -i inode di ciascun file

Creare e cancellare directories

- `$ mkdir dir_name` (crea una directory).
  - `$ mkdir appunti ; ls`
  - `$ mkdir {appunti,lucidi};  ls`
- `$ rmdir dir_name` (cancella directories vuote. Attenzione: no warning!).
  - `$ rmdir appunti`
- Se la directory non è vuota, per cancellarla si deve usare `$ rm –r dir_name`
  - `$ rm –r appunti`

Creare e cancellare files

- `$ cat > file_name` (scrive l'input proveniente dalla tastiera nel file `file_name`).
  - `$ cat > prova`

```
Help! I'm stuck in a Linux program!
^D
```

(Il testo in rosso indica ciò che scrive l'utente).

- `$ rm file_name` (cancella il file. Attenzione: no warning!).
  - `$ rm pippo` (Cancella definitivamente il file `pippo`. No warning!).
  - rm options:
    - -r (*recursive)*: rimuove i contenuti delle directories uno dopo l'altro
    - -i (*interactive*): avvisa prima di cancellare ciascun file
    - -f (*force*): forza rm a cancellare i files ignorando errori o avvertimenti

Copiare

- `$ cp options file1 file2` (per copiare il file file1 nel file file2).
  - `$ cp /etc/passwd pass`
  - `$ cp problemi/*  ~/backup`
- Se il file file2 non esiste, allora cp lo crea; altrimenti cp lo sovrascrive.
- Se file2 è una directory, cp fa una copia del file1 nella directory file2.
  - `$ cp pippo /articoli`
  - `$ cp /etc/passwd .`
- cp: opzioni
  - -i avvisa prima di sovrascrivere su un file esistente
  - -p conserva i permessi
  - -r copia i files e le sottodirectories uno dopo l'altro

Spostare

- `$ mv olddirectory newdirectory` (rinomina la directory olddirectory in newdirectory).
  - Se `newdirectory` esiste già, mv sposta `olddirectory` dentro quella nuova.
- `$ mv oldname newname` (rinomina il file oldname in newname).
  - Se `newname` esiste già, mv scrive `oldname` su `newname`.
- mv: opzioni
  - -i avvisa prima di sovrascrivere su un file esistente
  - -f forza mv ad agire indipendentemente dai permessi (non sempre…)
- `$ mv file path` (sposta il file `file` dalla current directory alla nuova directory, indicata in `path`).
  - `$ mv chap[1,3,7] book` (sposta i files chap1, chap3, e chap7 nella directory book).
  - `$ mv chap[1-5] book` (sposta i files da chap1 a chap5 nella directory book).

Mostrare i contenuti di un file

- `$ cat filename` (mostra i contenuti di filename).
- `$ more filename` (mostra la prima schermata di filename; usare la barra spaziatrice per proseguire nelle schermate successive; si chiude automaticamente dopo l'ultima schermata).
- `$ less filename` (mostra la prima schermata di filename; usare la barra spaziatrice per proseguire nelle schermate successive; usare le frecce per spostarsi su e giù di una linea per volta; è necessario digitare `q` o `Q` per uscire dal comando less).
- `$ head -n filename` (mostra le prime `n` linee di filename. Se n non è specificato, di default di solito è 10).
- `$ tail -n filename` (mostra le ultime `n` linee di filename. Se n non è specificato, di default di solito è 10).

Il comando more

`more [-cs] [+startline] [+/pattern] [filename]`

- ✓ `startline:` numero di linea da cui si vuole iniziare a visualizzare;
- ✓ `pattern:` pattern iniziale da cercare;

- `more dati: mostra “dati” una pagina alla volta;`

`Comandi fondamentali:`

- h: informazioni aggiuntive;
- <space>: mostra la schermata successiva;
- q: esce dal programma;
- <return>: va avanti di una riga;
- n <return>: va avanti di *n* righe;
- v: apre il file con l’editor vi;
- . : ripete l’ultimo comando.

`Opzioni:`

- -c: mostra le schermate successive dall’alto della pagina;
- -s: sostituisce linee vuote consecutive con un’unica linea vuota
- +/pattern: mostra il file a partire dalla prima occorrenza di “pattern”;

Standard Input e Standard Output

- Ogni programma eseguito dalla shell apre tre files:
  - standard input ←0
  - standard output ←1
  - standard error ←2
- I files forniscono i principali mezzi di comunicazione tra i programmi, e rimangono in vita per tutta la durata del processo.
- Il file standard input fornisce un modo per inviare dati a un processo. Di default, lo standard input viene letto dalla *tastiera del terminale*.
- Lo standard output fornisce al programma un mezzo per rendere disponibili i dati. Di default, lo standard output viene visualizzato sullo *schermo del terminale*.
- Lo standard error è dove il programma registra ogni eventuale errore incontrato durante l'esecuzione. Di default, anche lo standard error viene indirizzato sullo *schermo del terminale.*

Standard Input e Standard Output

- 1>pippo invia lo standard output al file pippo.
- 2>pippo invia lo standard error al file pippo.
- Es. $ ls 1> pippo scrive l'output di ls in pippo,
  come ls > pippo
  $ list 2> pippo scrive l'errore
  "-bash: list: command not found"
  nel file pippo.

Redirezionare Input e Output

- È possibile dire a un programma:
  - dove cercare l'input
  - dove inviare l'output,

  usando la tecnica di redirezione dell'input/output. In UNIX si usano i caratteri speciali **<** e **>** per significare redirezionamento di input e output, rispettivamente.
- Redirezionare l'input: usando **<** con il nome di un file (as es., < file1) in un comando di shell, si comunica alla shell di leggere l'input da un file chiamato "file1" invece che dalla tastiera.
  - `$ more < /etc/passwd`
- Redirezionare l'output: usando **>** con il nome di un file (per es., > file 2), si impone alla shell di memorizzare l'output del comando in un file chiamato "file2" invece che sullo schermo. Se il file "file2" esiste già, la versione vecchia verrà sovrascritta.
  - `$ ls /tmp > ~/ls.out`
  - `$ sort pippo > pippo.ordinato`

Redirezionare Input e Output

- L'uso di **>>** per completare un file esistente (per es., >> file2) impone alla shell di accodare l'output del comando alla fine di un file chiamato "file2".
  - Se il file "file2" non esiste già, verrà creato.
- Esempio

1. `$ ls /bin  > ~/bin;  wc –l ~/bin`

```
$ ls /usr/sbin > ~/bin ; wc -l ~/bin
```

2. `$ ls /bin  > ~/bin;`

```
$ ls /usr/sbin >> ~/bin; wc –l ~/bin
```

Redirezionare l’errore

- L’uso di **>&** con il nome di un file (ad es., >& file1) impone alla shell di inserire lo standard error e lo standard output del comando in un file detto "file1".
  - Se il file "file1“ esiste già, la versione vecchia verrà sovrascritta.
- Esempio
  - `$ ls abcdef`
  - `$ ls abcdef >& lserror`
  - `cat lserror`
  - `$ abcdef >& command`
  - `cat command`
  - `$ mkdir /bin/miei >& ~/miei; cat ~/miei`
  - `$ rm /bin/perl >& ~/errperl; cat ~/errperl`

Pipes (tubi)

- UNIX offre la possibilità di connettere processi, permettendo allo standard output di un processo di venire usato come standard input di un altro processo. Questo meccanismo viene detto una **pipe (|)**.
- `$ command1 | command2` fa sì che lo standard output di `command1` venga utilizzato come standard input di `command2`.
- Una sequenza di comandi concatenati in questo modo viene detto una pipeline. Connettere processi semplici in una pipeline permette di eseguire compiti complessi senza scrivere programmi troppo complessi.
  - `$ cat /etc/passwd | sort > ~/pass_ord`
  - `$ sort < pippo | lpr`

Esercizi

1. Determinare il numero di files nella directory /bin la cui prima lettera è “c”.
2. Creare un file contenente i nomi dei primi 7 files della directory /etc.
3. Determinare il numero dei files della current directory nel cui nome compare la stringa *“string”.*
4. Creare un file contenente una lista col nome di 10 comandi di /bin ordinati secondo il momento dell’ultimo accesso.
5. Creare un file contenente i nomi dei primi 7 files e gli ultimi 6 files (in ordine alfabetico) della directory /etc.
6. Creare un file contenente una lista coi nomi di 8 files in /usr/sbin ordinati secondo il momento dell’ultima modifica.

Sequenze e Valori di Uscita

- Una serie di comandi semplici o pipelines separati da punto e virgola (;) verrà eseguita in sequenza da sinistra a destra.
  - `pwd ; ls; echo the date of today is ; date`
- Ogni processo di Linux temina con un valore di uscita:
  - 0 : il processo si è completato con successo,
  - valore di uscita diverso da zero : indica fallimento.
- La variabile di shell $? contiene il valore del codice di uscita del comando precedente.
  - `$ date`

```
dom mar 16 18:41:44 CET 2003
$ echo $?
0
```

  - `$ mkdir /ect/prova`

```
mkdir: impossibile …
$ echo $?
1
```

Sequenze condizionali

- Specificando una serie di comandi separati da **&&**, il comando successivo viene eseguito solo se il precedente ha 0 come codice di uscita.

  - `date && ls`

```
date && ls && abcdef && date
```

- Specificando una serie di comandi separati da **||**, il comando successivo viene eseguito solo se il precedente ha 1 come codice di uscita.

  - `abcdef || date`

```
abcdef || rm /bin || date || ls
```

Ricerca dei files

- L'utility find è un comando molto versatile, senza corrispondente in DOS. Si può usare find per cercare files usando una grande varietà di condizioni di ricerca, quindi eseguire svariate azioni coi risultati della ricerca. La sintassi è:
  - `$ find <path> <search-condition(s)> <action>`
- L'utility find:
  - Discende ricorsivamente attraverso il path,
  - Applica le condizioni di ricerca a ogni file.
- Condizioni di ricerca:
  - -atime n files visitati n giorni fa
  - -mtime n files modificati n giorni fa
  - -size n[bckw] files di dimensione esattamente n (ove n può essere data in blocchi da 512 byte [b], caratteri da 1 byte [c], Kilobytes [k] o parole da due bytes [w])
  - -type c tipo di file (ad es., f=file, d=directory, l=link)
  - -name "name" trova tutti i files di nome "name" (es., "*.c")

Ricerca dei files

- Come troviamo un file chiamato kernel in /usr/src?

```
$ find /usr/src –name “kernel”
```

- Come troviamo i files con estensione pdf?

```
$ find / -name “*.pdf”
```

- Come facciamo a trovare quali files sono stati modificati nella nostra home negli ultimi due giorni?

```
$ find ~ -mtime –2
```

- Come facciamo a trovare i files della nostra home visitati nelle ultime due ore?

```
$ find ~ -amin -120
```

- Come troviamo files di dimensione superiore a 1 MB, ma inferiore a 2 MB?

```
$ find / -size +1024 –size –2048
```

- Come troviamo i files nella current directory più recenti del file “test”?

```
$ find . -newer test
```

Ricerca dei files (<action>)

- <action>:
  - -exec *command [options]* { } \;

    esegue il comando *command* usando come input il file trovato; { } rappresenta il percorso del file trovato, mentre \; termina la linea di comando.
  - *-ok command [options]* { } \;

    come –exec ma richiede conferma.
  - -print

    mostra i files trovati sullo schermo; non è necessario se non dopo un –exec, oppure se si vuole redirezionare l'output.

- Come facciamo a trovare i files con estensione tmp e poi cancellarli con richiesta di conferma?

```
$ find / -name "*.tmp" -ok rm {} \;
```

- Come facciamo a cancellare tutti i files più vecchi di un mese (modificati più di un mese fa) nella directory /tmp?

```
$ find /tmp -mtime +30 -exec rm {} \;
```

- Come facciamo a trovare i files dell'utente lferrari (che hanno cioè l'utente lferrari come owner)?

```
$ find / -user lferrari
```

- Come facciamo a trovare tutti i files che cominciano con "pr" nella current directory la cui dimensione è minore di 10Kbyte, quindi registrare l'output nel file "risultato"?

```
$ find . -name "pr*" –maxdepth 1 –size –10k
–print > risultato
```

Filtrare i files

- grep sta per get regular expression. Si può usare grep quando cerchiamo files contenenti un motivo specifico. grep è stato ulteriormente esteso con comandi come egrep e fgrep. La sintassi è:
  - `$ grep <options> <search-pattern> <file(s)>`
- Grep mostra le linee
  - che contengono il motivo `<search-pattern>`
  - in ciascuno dei files `<file(s)>`.
- Esempio:

```
$ grep Italy /usr/src/linux/CREDITS
$ grep pippo ~/prova.txt
```

Opzioni

- -n visualizza anche il numero di riga

```
$ grep –n italy /usr/src/linux/CREDITS
```

- -c mostra il numero di linee in cui la sequenza viene trovata, ma non le linee stesse

```
$ grep –c Italy /usr/src/linux/CREDITS
```

- -i non distingue tra maiuscole e minuscole

```
$ grep –i Italy /usr/src/linux/CREDITS
```

- -w trova solo parole intere

```
$ grep –w Italy /usr/src/linux/CREDITS
```

- -q dà 0, se il testo è stato trovato, e 1 altrimenti

```
$ grep –q Italy /usr/src/linux/CREDITS
```

- -l visualizza solo i nomi dei file che contengono le righe corrispondenti all'espressione regolare, ma non le righe stesse

```
$ grep –l Italy /usr/src/linux/CREDITS
```

Espressioni regolari

- abc — la sequenza “abc”
- [abc] — uno dei caratteri a, b o c
- [~abc] — nessuno dei caratteri a, b or c (tutti gli altri vanno bene)
- [a-z] — un carattere da a a z
- . — un carattere arbitrario
- * — il carattere può comparire 0,1 o n volte
- + — il carattere può comparire 1 o n volte
- {n} — il carattere deve comparire n volte
- {, n} — il carattere deve comparire al più n volte
- {n, } — il carattere deve comparire almeno n volte
- {n,m} — il carattere deve comparire almeno n volte e al più m volte

Alcune espressioni regolari

- .at ogni occorrenza di una lettera, seguita da at, come cat, rat, mat, bat, fat, hat
- xy*z ogni occorrenza di una x, seguita da zero o più y, seguita da una z
- ^cat cat all'inizio della linea
- cat$ cat alla fine della linea
- * ogni occorrenza di un asterisco
- \(. * \) ogni sequenza (non vuota) tra parentesi
- [cC]at cat o Cat
- [a-zA-Z] ogni occorrenza di un carattere alfabetico
- [0-9]$ ogni linea che termina con un numero
- [A-Z][A-Z]* una o più lettere maiuscole
- [A-Z]* zero o più lettere maiuscole

Filtrare i files

```
$ grep .it /usr/src/linux/CREDITS
$ grep “\.it” /usr/src/linux/CREDITS
$ grep “\.it$” /usr/src/linux/CREDITS
$ grep –q “dario.ballabio” /usr/src/linux/CREDITS
```

Find + Grep

- Il comando grep può essere invocato all'interno di una find
  - `$ find –name '*tex' –exec grep emacs {} \;`

- Trovare la sequenza emacs in tutti i files .tex
- `$ find –name '*tex' –exec grep –q emacs {} \;`
- `$ find –name '*tex' –exec grep –q emacs {} \; -print`
- `$ find –name '*tex' –exec grep –q emacs {} \; -print > pippo`

Esercizi

1. Trovare i files contenenti la lettera “c” nella tua home e metterli in un nuovo file.
2. Creare un file contenente la lista dei nomi ed e-mail dei manutentori italiani di Linux.
3. Creare un file contenente la lista dei files della directory /etc contenenti la stringa “utente”.
4. Trovare tutti i files che cominciano con la lettera “p” e copiarli nella tua home; inoltre, creare nella tua home un file (di nome “errore”), nel quale registrare tutti i risultati del find effettuato (compresi dunque tutti i messaggi di errore).

L'editor *vi*: lista dei comandi I

INSERT MODE

ESC passa a command mode
^H cancella l'ultima lettera
^W cancella l'ultima parola
^U cancella l'ultima linea

COMMAND MODE

i passa a insert mode
x cancella un carattere in command mode
dd cancella una singola linea
dw cancella una singola parola
D cancella fino alla fine della linea
:nn sposta il cursore alla nn-esima linea
:nn d cancella la linea d
p incolla l'ultimo oggetto cancellato

CARATTERI SPECIALI

. linea corrente
$ ultima linea del file

L’editor *vi*: lista dei comandi II

: <range> y	copia le linee definite da range (senza cancellarle)
: pu	incolla le linee copiate dopo la linea corrente
: nn pu	incolla le linee copiate dopo la linea nn
: <range> m nn	sposta le linee definite da range dopo la linea nn
/ *pattern*	cerca *pattern* sotto la posizione corrente del cursore
/	cerca tutte le occorrenze del pattern
? pattern	cerca *pattern* sopra la posizione corrente del cursore
?	cerca tutte le occorrenze del pattern
:<range> s / *pattern* / *replace* /	sostituisce la prima occorrenza di *pattern* con *replace* in range
:<range> s / *pattern* / *replace* / g	sostituisce tutte le occorrenze di *pattern* con *replace* in range
:s / *pattern* / *replace* / c	sostituisce l’occorrenza di *pattern* con *replace* nella linea corrente con richiesta di conferma
u	annulla l’ultimo comando
:nn r *! command*	esegue il comando di shell *command* a partire dalla linea nn

L’editor *vi*: lista dei comandi III

:w	salva il file
:w *file*	salva il file col nome *file*
:w *>>file*	accoda ciò che si è scritto al file *file*
:q!	esce senza salvare

L’editor *vi*: esempi

:1,3 d cancella le linee 1,2,3
:1,$ d cancella tutto il file
:1,. d cancella dalla linea 1 alla linea corrente
: . , .+5 d cancella la linea corrente e le cinque linee successive

: 1,3 y copia le linee 1,2,3
: 4 pu inserisce le tre linee dopo la linea 4
: 1,3 m 4 sposta le linee 1,2,3 sotto la linea 4

:1,$ s/UNIX/unix/g sostituisce UNIX con unix in tutto il file
:3,. s/UNIX/unix/c
sostituisce UNIX con unix nelle linee dalla 3 alla corrente chiedendo conferma

:4 !ls aggiunge l’output di ls dopo la linea 4
:$!date aggiunge la data alla fine del file
:1,5 !sort ordina le linee dalla 1 alla 5

Stampare i files: *lpr*

- lpr [-# num] [-j name] [-p printer] [-t title] [file ...]
  - num: numero di copie
  - name: nome del job
  - printer: nome della stampante (nessun nome = quella di default)
  - title: titolo all'estremità superiore delle pagine stampate
  - file ... : lista dei files da stampare
- lpq [job ...] [-p printer] (controllo lo stato della stampa)
  - job: numero del job da controllare
  - printer: stampante su cui il job è in esecuzione
- lprm [job ...] [-p printer] (elimina una stampa)

  - `$ lpq –p hplaserjet1200`

```
lp is ready and printing
Rank    owner      job  files      total size
Active  frosini    773  pippo.txt  1200 bytes
```