

Il sistema operativo UNIX/LINUX

Fabio Maino <maino@polito.it>
Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

**Politecnico di Torino
Dip. Automatica e Informatica**

Organizzazione

- **Assistenti:**
 - Giorgio Di Natale
 - dinatale@polito.it (7084)
 - Stefano Di Carlo
 - dicarlo@polito.it (7084)
- **Esercitazioni:**
 - in aula (lun. 12.30-14.30):
 - al CCLINF (gio. 10.30-14.30):

Programma (I modulo)

- Il sistema operativo Linux: introduzione
- Comandi principali
- vi: un editor di testo
- Shell: l'interprete dei comandi
- Tools e Comandi avanzati
- Uso avanzato di shell e shell script
- Amministrazione del sistema

Il sistema operativo LINUX Introduzione

Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

**Politecnico di Torino
Dip. Automatica e Informatica**

Indice

- Introduzione
- Il file system
- Manipolazione di file e directory
- I comandi principali

Indice

- Introduzione
- Il file system
- Manipolazione di file e directory
- I comandi principali

Il sistema operativo LINUX

La shell

Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

Politecnico di Torino
Dip. Automatica e Informatica

La shell

- E' lo strato più esterno del S.O. e fornisce l'interfaccia utente
- Offre una via di comunicazione con il S.O.
 - dialogo interattivo
 - comandi memorizzati in un file di script
- In Unix la shell non è parte del kernel
 - è un normale processo utente

Esecuzione della shell

- Una shell può essere attivata:
 - automaticamente al login (secondo la specifica in /etc/passwd)
 - in modo annidato dentro un'altra shell (si ritorna alla shell iniziale quando termina quella interna)
- Per terminare una shell:
 - exit
 - il carattere di EOF (tipicamente ^d)

Caratteri speciali

- / separa i nomi dei direttori in un path
- ? un carattere qualunque
- * una sequenza di caratteri qualunque
- ~ il direttorio di login
- ~utente il direttorio di login dell'utente
- [] un carattere tra quelli in parentesi
- { } una parola tra quelle in parentesi (separate da virgola)
- '...' non espande le espressioni regolari

Le shell disponibili

- In Unix sono disponibili molte shell:
 - Bourne shell (sh): la shell originaria, molto usata nella programmazione sistemistica
 - C-shell (csh): la shell di Berkeley, ottima per l'uso interattivo e per gli script non di sistema
 - Korn shell (ksh): la Bourne shell riscritta dall'AT&T per assomigliare di più alla C-shell
 - Tahoe C-shell (tcsh): dal progetto Tahoe, una C-shell migliorata
 - Bourne again shell (bash)

File di configurazione della shell

- All'avviamento ogni shell cerca nel direttorio di login i propri file di configurazione:
 - **.login** (csh, tcsh): comandi eseguiti al login
 - **.cshrc** (csh, tcsh): comandi eseguiti all'avviamento
 - **.profile** (sh, ksh): comandi eseguiti al login
- csh e tcsh usano anche il file **.logout** per eseguire comandi al termine della sessione

Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

Completion

- Espansione dei nomi di file con il carattere memorizzato nella variabile filec (spesso TAB o ESC)
 - per i nomi di file eseguibili la shell cerca nei direttori del path
 - per file generici, la shell espande nomi di file nel direttorio corrente

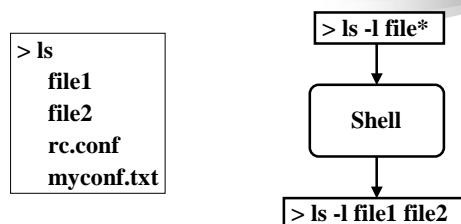
Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

Espressioni regolari

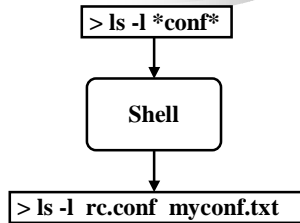
- La shell espande automaticamente le espressioni regolari
- Le espressioni regolari vengono sostituite con la lista dei nomi di file che soddisfano il pattern

Espressioni regolari (cont)



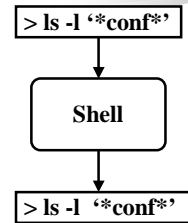
Espressioni regolari (cont)

```
> ls
file1
file2
rc.conf
myconf.txt
```



Espressioni regolari (cont)

```
> ls
file1
file2
rc.conf
myconf.txt
```

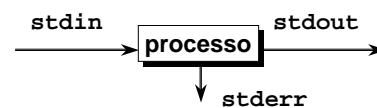


Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

Redirezione dell'I/O

- Ogni processo ha tre canali di I/O standard:



- Ogni canale può essere ridiretto:
 - su file
 - su un altro canale tramite pipe

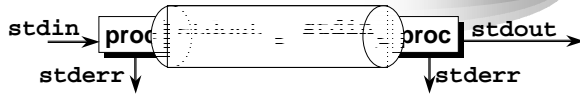
Ridirezione dell'I/O da/su file

- **comando < file** stdin da file
- **comando > file** stdout in file (cancellato se esiste)
- **comando >> file** stdout accodato a file
- **comando <<HERE** stdin da “here document”
text
HERE
- **comando >& file** (csh) stderr+stdout in file
- **comando 2> file** (ksh) stderr in file (&1 è stdout)

Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

Definizione della pipe



- Il collegamento stdout-stdin si chiama *pipe* e crea in memoria un canale diretto tra i due processi

Redirezione dell'I/O tramite pipe

- `comando1 | comando2`
 - pipe tra i due comandi
- **Esempi:**
 - `ls -la | more`

Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

History

- `↑↓`: mostra i comandi eseguiti precedentemente
- `!n`: esegue il comando numero n nel buffer
- `!-n`: esegue l'n-ultimo comando
- `!$`: l'ultimo parametro del comando precedente
- `!*:` tutti i parametri del comando precedente
- `!stringa`: l'ultimo comando che inizia con stringa
- `^stringa1^stringa2`: rimpiazza le occorrenze di stringa1 nell'ultimo comando con stringa2

C-shell: esempio di history

```
25% cc -g prog.c
26% vi iop.c
27% cc prog.c iop.c
28% a.out lettera
```

- `rm !$` `rm lettera`
- `!-1` `a.out lettera`
- `!c` `cc prog.c iop.c`
- `!25` `cc -g prog.c`
- `rm !*` `esegue rm a.out lettera`

Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

Aliasing

- E' possibile definire dei comandi con nuovi nomi (*alias*), tipicamente più semplici
- **alias**
Elenca gli alias definiti
- **alias nome valore** (C-shell)
Definisce un alias (racchiudere *valore* tra apici se contiene degli spazi)
- **unalias nome**
Cancella un alias

Esempi

- **alias dir ls**
- **alias tgz tar czvf**

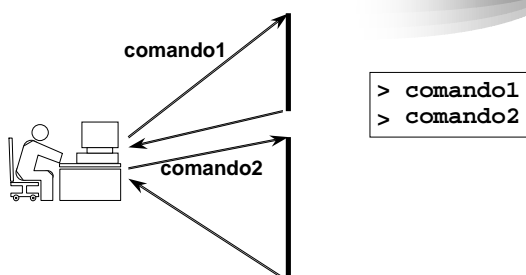
Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

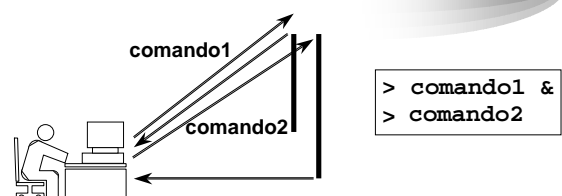
I processi

- Linux è un sistema operativo multitasking
- E' possibile eseguire contemporaneamente più processi
- Dalla shell è possibile eseguire i comandi in due modalità:
 - batch: l'utente può eseguire un nuovo comando solo dopo la terminazione del primo processo
 - concorrente: l'utente può eseguire un nuovo comando anche se il primo non è concluso

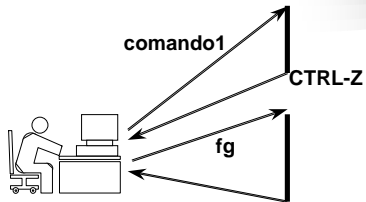
Esecuzione batch



Esecuzione concorrente



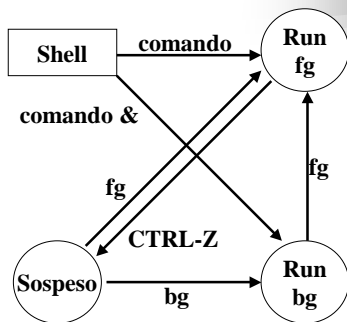
Interruzione di processi



Stato dei processi

- **Esecuzione in foreground**
 - hanno i tre canali standard connessi al terminale
- **Esecuzione in background**
 - sono privati di stdin
- **Sospeso**

Stato dei processi



Comandi per gestire i processi

- **jobs** elenca i job
- **bg %job-id** porta il job in background
- **fg %job-id** porta il job in foreground

I processi

- **A ogni processo sono associati:**
 - pid process id
 - uid user id di chi ha eseguito il processo
 - stime istante in cui il processo è partito
 - ...
- **Il comando ps mostra l'elenco dei processi**

Il comando ps

- **Il comando ps permette di elencare i processi ed il loro stato**
 - -e elenca tutti i processi
 - -f elenco in formato pieno
 - -l elenco in formato lungo

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675  110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```

Il comando ps

- Il comando ps permette di elencare i processi ed il loro stato
- Stato:
 - R in esecuzione
 - T bloccato
 - S sleeping
 - Z zombie

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```

Il comando ps

- Il comando ps permette di elencare i processi ed il loro stato
- UID dell'utente che ha eseguito il comando

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```

Il comando ps

- Il comando ps permette di elencare i processi ed il loro stato
- Process ID

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```

Il comando ps

- Il comando ps permette di elencare i processi ed il loro stato
- Parent PID

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```

Il comando ps

- Il comando ps permette di elencare i processi ed il loro stato
- Console

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```

Il comando ps

- Il comando ps permette di elencare i processi ed il loro stato
- Tempo complessivo dedicato al processo

```
% ps -l
S  UID  PID  PPID  TTY  TIME  CMD
R 2103 1728 1676  ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675  ttys0 0:00 -csh
```


Il comando ps

- Il comando **ps** permette di elencare i processi ed il loro stato

- **-e** elenca tutti i processi
- **-f** elenco in formato full
- **-l** elenco in formato long

Comando

```
% ps -l
S  UID  PID  PPID  TTY  TIME CMD
R 2103 1728 1676 ttys0 0:00 ps
S   0 1675 110  ttys0 0:00 telnetd
S 2103 1676 1675 ttys0 0:00 -csh
```

Terminazione di un processo

- E' possibile terminare forzatamente un processo con i comandi:

- **kill -9 pid**
- **kill -9 %job-id**

Gestione temporizzata dei processi

- **at time filename**
 - esegue il programma all'ora specificata
- **at -l**
 - elenca i job sottomessi
- **at -r [jobname]**
 - rimuove dalla coda il job

Caratteristiche della shell

- **Completion**
- **Gestione di espressioni regolari**
- **Redirezione dell'I/O**
- **Pipeline**
- **History**
- **Aliasing**
- **Gestione dei processi**
- **Scripting**
- **Variabili**

File di comandi (script)

- E' possibile memorizzare in un file una serie di comandi, eseguibili richiamando il file stesso
- **Esecuzione indiretta:**
 - **source <scriptname> <args>**
- **Esecuzione diretta eseguendo lo script**
 - è necessario che abbia il permesso di esecuzione
 - la prima riga del file inizia con **#!** seguita dal nome (con il path assoluto) della shell con cui si devono eseguire i comandi

Script di shell

```
#!/bin/csh
date
who
```

Caratteristiche della shell

- Completion
- Gestione di espressioni regolari
- Redirezione dell'I/O
- Pipeline
- History
- Aliasing
- Gestione dei processi
- Scripting
- Variabili

C-shell: variabili

- La shell mantiene un insieme di variabili per la personalizzazione dell'ambiente
- **set variabile = valore**
 - assegna un valore alla variabile
- **set**
 - visualizza il valore di tutte le variabili
- **echo \$variabile**
 - visualizza il valore della variabile indicata

Variabili di shell

- **Variabili più utili:**
 - home = direttorio di login
 - path = direttori in cui cercare i comandi (in parentesi, separati da spazi)
 - prompt = il prompt dei comandi
 - cwd = il direttorio corrente
 - status = il risultato dell'ultimo comando
- Usare rehash dopo aver cambiato il path

C-Shell: ambiente

- L'ambiente (*environment*) di un processo è costituito da una lista di coppie:
(variabile , valore)
dove *valore* è una stringa di caratteri
- **setenv variabile [=valore]**
assegna un valore alla variabile
- **printenv [variabile]**
stampa il valore di una o tutte le variabili d'ambiente
- **env**
stampa il valore di tutte le variabili d'ambiente

Variabili di ambiente

- **HISTFILE**
 - nome del file utilizzato per l'history
- **HOME**
 - home directory
- **LOGNAME**
 - username
- **PATH**
- **SHELL**
 - shell utilizzata

Il sistema operativo LINUX

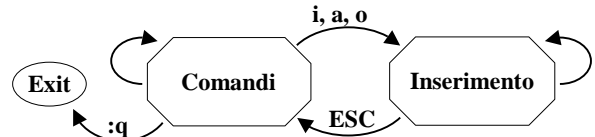
L'editor vi

Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

Politecnico di Torino
Dip. Automatica e Informatica

Caratteristiche dell'editor vi

- Si trova su tutti gli UNIX e funziona con qualunque terminale
- Due modi base di funzionamento:
 - modo comandi
 - modo inserimento



Esecuzione di vi

- **vi file**
 - edita il file (lo crea se non esiste)
- **view file**
 - edita un file ma non permette di modificarlo
- **vi -r file**
 - edita il file cercando di recuperare (per quanto possibile) le modifiche fatte (e non salvate) prima di una brusca interruzione

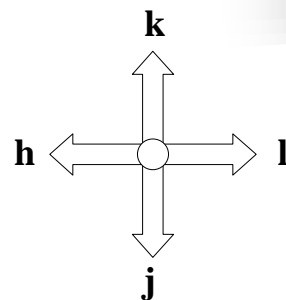
Indice

- **Comandi di base**
- **Altri comandi**
- **Ricerche e sostituzioni**

Indice

- **Comandi di base**
- **Altri comandi**
- **Ricerche e sostituzioni**

Movimento del cursore



Movimento del cursore

- **w** alla prossima parola
- **0** all'inizio della riga
- **\$** alla fine della riga
- **CTRL-f** avanti di una schermata
- **CTRL-b** indietro di una schermata
- **1G** all'inizio del file
- **G** alla fine del file
- **nG** alla riga n

Ripetizione dei comandi

- L'effetto di molti comandi può essere ripetuto digitando, prima del comando stesso, il numero di ripetizioni:
 - **w** si sposta di 1 parola
 - **5w** si sposta di 5 parole
 - **10CTRL-f** avanti di 10 schermate
 - **20h** si sposta di 20 caratteri a sinistra

Inserimento di testo

- i testo** nella posizione del cursore
- a testo** dopo il cursore
- I testo** a inizio riga
- A testo** a fine riga
- O testo** in una nuova linea sopra l'attuale
- o testo** in una nuova linea sotto l'attuale
- :r file** tutto il contenuto di *file* viene inserito sotto l'attuale riga

Inserimento di testo

- i testo** nella posizione del cursore
 - a testo** dopo il cursore
 - I testo** a inizio riga
 - A testo** a fine riga
 - O testo** in una nuova linea sopra l'attuale
 - o testo** in una nuova linea sotto l'attuale
 - :r file** tutto il contenuto di *file* viene inserito sotto l'attuale riga
- Dopo questi comandi, l'editor passa in modalità inserimento.

Inserimento di testo

- i testo** nella posizione del cursore
 - a testo** dopo il cursore
 - I testo** a inizio riga
 - A testo** a fine riga
 - O testo** in una nuova linea sopra l'attuale
 - o testo** in una nuova linea sotto l'attuale
 - :r file** tutto il contenuto di *file* viene inserito sotto l'attuale riga
- Dopo questo comando, l'editor resta in modalità comando.

I/O verso file

- :w [file]** salva il file
- :e file** edita il file indicato
- :e!** ri-edita il file corrente scartando tutte le modifiche già fatte
- :x :wq** salva il file e termina
- :q** termina
- :q!** termina senza salvare

Cancellazione

- **[n]x** cancella i prossimi n caratteri
- **[n]X** cancella i precedenti n caratteri
- **[n]dw** cancella le prossime n parole
- **[n]dd** cancella n linee
- **D** cancella fino alla fine della linea

Cancellazione avanzata

- **:[range]d**
- **range:**
 - riga inizio, riga fine
- **Caratteri speciali:**
 - **.** riga corrente
 - **\$** ultima riga
 - **-\$-1** penultima riga

Esempi

- **:1,\$d** cancella tutto il contenuto del file
- **:3,4d** cancella le righe 3 e 4
- **:.+1,\$-5d** cancella dalla riga seguente alla quintultima riga del file

Undo

- **u** annulla l'ultima modifica
- **U** annulla tutte le modifiche alla linea corrente
- **CTRL-I** ridisegna lo schermo

Help on-line

- **Dalla modalità comandi:**
 - **:help [topic]**

Indice

- **Comandi di base**
- **Altri comandi**
- **Ricerche e sostituzioni**

Comandi vari

- **J** unisce due linee
- **r char** sostituisce char al carattere corrente
- **~** converte il carattere corrente da maiuscolo a minuscolo e viceversa
- **.** ripete l'ultima modifica
- **:f** informazioni relative al file corrente

Comandi vari

- **s** elimina il carattere sotto il cursore e entra in modo inserimento
- **C** elimina il testo fino a fine riga e entra in modo inserimento
- **cw** cancella una parola e entra in modo inserimento

Cut, Copy e Paste

- Per spostare o copiare parte del testo, occorre metterlo in un buffer
- **Copia:**
 - [n]yy ricopia n righe
- **Taglia:**
 - [n]dd taglia n righe
 - D taglia dal cursore a fine riga
- **Incolla:**
 - P inserisce prima del cursore
 - p inserisce dopo il cursore

Cut & Paste Avanzato

- Occorre servirsi, implicitamente o esplicitamente di uno dei seguenti buffer:
 - buffer denominato (nome di una lettera a-z)
 - buffer numerato [1-9]
 - DTB, Deleted Text Buffer (default)
- Un comando di cut&paste ha la forma:
" **bufname azione**
 - bufname specifica il nome del buffer
 - l'azione può essere: y, yG, dd, p, P

Esempi

- **"a5dd**
 - sposta 5 linee nel buffer a
- **"byG**
 - copia dalla posizione corrente alla fine del file nel buffer b
- **"ap**
 - estrae il contenuto del buffer a inserendolo nella riga dopo il cursore
- **"bP**
 - estrae il contenuto del buffer b inserendolo nella riga prima del cursore

Azioni speciali

- **!comando**
 - esegue comando in UNIX
- **!}comando**
 - esegue il comando ed inserisce il suo output al posto del paragrafo attuale (es. !}fint)
- **:map key comandi-vi**
 - assegna alla chiave la sequenza di comandi indicata. Esempio:
 - :map CTRL-W :wq

Editing di più file

- **vi file1 file2 ...**
 - per editare una serie di file in sequenza
- **Quando si finisce con ciascun file:**
 - :w per salvare il file
 - :n per passare al prossimo file (:n! per non salvare quello attuale)
 - :e file per editare un nuovo file (:e! per non salvare quello attuale)

vi - aiuti alla programmazione

- % posiziona il cursore sulla parentesi che bilancia quella su cui è il cursore
- >> indenta di una posizione la riga corrente
- << de-indenta di una posizione la riga corrente
- Ctrl-D elimina un'indentazione automatica durante l'inserimento di testo

Indice

- Comandi di base
- Altri comandi
- Ricerche e sostituzioni

Espressioni regolari in vi

- . un carattere qualunque
- ^ inizio riga
- \$ fine riga
- * ripetizione (zero o più volte)
- + ripetizione (una o più volte)
- [] un carattere tra quelli in parentesi
- [^] un carattere esclusi quelli in parentesi
- [x-y] un carattere nel range specificato

Espressioni regolari in vi (cont)

- \ carattere per formare le sequenze di escape
- \t tab
- \\$ carattere \$
- < inizio parola
- > fine parola

Ricerche in vi

- **Ricerche:**
 - /pattern ricerca in avanti una stringa corrispondente al pattern
 - ?pattern ricerca all'indietro
 - n cerca la prossima occorrenza
 - N cerca l'occorrenza precedente
- **Il pattern può essere una normale stringa oppure una espressione regolare ossia avere caratteri con significato speciale**

Esempi

- `/estate` ricerca la parola estate
- `^[<[A-Z]` ricerca una qualunque parola che comincia con una lettera maiuscola
- `/^estate` ricerca la parola estate ad inizio riga
- `/are\>` ricerca tutte le parole con il suffisso are
- `?##*` ricerca indietro una ripetizione di uno o più simboli #

Sostituzioni

- Per ogni linea in `[range]`, sostituisce la prima occorrenza di `{pattern}` con `{string}`:
`:[range]s/{pattern}/{string}/[c][g][i]`
- Opzioni:
 - `[c]` Chiede conferma prima di ogni sostituzione
 - `[i]` Ignora il case per il pattern
 - `[g]` Effettua la sostituzione per tutti gli elementi della riga

Sostituzioni

- `:s/marzo/aprile/`
 - sostituisce nella linea corrente la prima occorrenza di marzo con aprile
- `:s/marzo/aprile/g`
 - sostituisce nella linea corrente tutte le occorrenze di marzo con aprile

Sostituzioni

- `:12,23s/marzo/ottobre/i`
 - sostituisce nelle righe da 12 a 23 le prime occorrenze di marzo con ottobre (case insensitive)
- `:1,$s/marzo/aprile/g`
 - sostituisce ovunque marzo con aprile

Collegamento col sistema

- Da un calcolatore dotato di rete TCP/IP:
 - telnet <hostname>
 - ssh [-l <username>] <hostname>

Una sessione di lavoro

- Inizio di una sessione:
 - login:
 - Password:
- Fine di una sessione:
 - CTRL-d exit logout

ATTENZIONE !
 Unix è case sensitive: i caratteri
 maiuscoli sono considerati diversi da
 quelli minuscoli



Caratteristiche principali

- Multiutente: a ogni user sono associati:
 - username
 - uid
 - gid
 - password
- Multiprocesso

Esiste l'utente
 privilegiato
 "root"

I comandi in UNIX

- La sintassi di un generico comando UNIX è:
 comando [-opzioni] [argomenti]
- I comandi troppo lunghi possono essere
 continuati sulla riga successiva battendo “\”
 come ultimo carattere della riga
- Si possono dare più comandi sulla stessa riga
 separandoli con “;”
 comando1 ; comando2 ; ...
 Essi saranno eseguiti in sequenza

Il manuale in linea

- Tutti i comandi di Unix sono documentati in
 linea:
 - man <comando>
 - apropos <termine>
 - whatis <comando>
- Alcuni sistemi sono dotati di una presentazione
 grafica (xman) o ipertestuale

Comunicazione tra gli utenti

- mail <username>
 manda un messaggio di posta elettronica (al
 login si riceve notifica dei messaggi non ancora
 letti)
- talk <username>
 permette di dialogare interattivamente con
 l'utente specificato

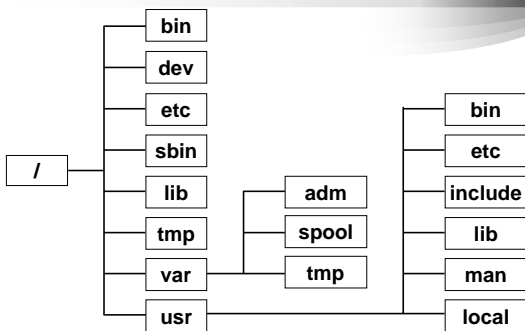
Indice

- Introduzione
- Il file system
- Manipolazione di file e directory
- I comandi principali

Il file system UNIX

- **Caratteristiche:**
 - gerarchico (relazione padre-figlio)
 - organizzazione a directory
 - uniformità di notazione (dischi, directory, file)
 - non esistono estensione e versione
 - *hard link* (stesso file con nomi diversi)
 - *soft link* (un file che punta ad un altro)
 - protezione

La gerarchia del file system



La gerarchia del file system

- L'elenco dei file di un direttorio e dei direttori suoi figli è chiamato " ." (dot)
- Il padre di un direttorio è puntato dal file " ." (dot dot)

I nomi dei file

- Il nome di un file può essere una sequenza di caratteri qualunque
- Non esistono estensione e versione
- Esistono comunque desinenze o nomi molto usati:
 - .c .f .p .o .a .so a.out core
- Se il nome di un file inizia col carattere punto " ." è detto *file nascosto* perchè normalmente non viene elencato

I nomi dei file (cont.)

- Si sconsiglia di utilizzare i seguenti caratteri nei nomi dei file:
 - / \ " ' * ; ? [] () ~
 - ! \$ { } < > # @ & |
- Si può usare il carattere spazio
- Per indicare un file non nel direttorio corrente si deve specificarne il path:
 - path assoluto: /dir1/dir2/file
 - path relativo: subdir1/subdir2/file

I file: classificazione

- **Un solo tipo fisico di file:**
 - byte stream (sequenza di byte)
- **Quattro tipi logici di file:**
 - file ordinario: ad esempio, i file di testo hanno le righe separate da LF (ASCII 10)
 - directory: contiene nomi ed indirizzi di altri file
 - special file: un entry point per un dispositivo di I/O
 - link: un puntatore ad un altro file

I link

- **Due tipi di link:**
 - hard link: un nome (in una directory) che punta ad un i-node puntato anche da altri nomi
 - soft link (symbolic link): un file che come unico blocco dati ha il nome di un altro file
- **Particolarità:**
 - no hard link ad una directory
 - no hard link a file su un altro file system
 - un file è fisicamente rimosso solo quando tutti i suoi hard link sono stati rimossi

Indice

- **Introduzione**
- **Il file system**
- **Manipolazione di file e directory**
- **I comandi principali**

Manipolazione di file

- **cp [-fir] src1 src2 ... dest**
 - copia uno o più file
- **rm [-fir] file1 file2 ...**
 - cancella i file elencati
- **mv [-fi] file1 file2 ... dest**
 - sposta (rinomina) uno o più file

Manipolazione di file (cont.)

- **Opzioni:**
 - -f non chiede mai conferma
 - -i chiede conferma per ciascun file
 - -r opera recursivamente su tutti i file contenuti nei sottodirettori

Manipolazione di direttori

- **cd <dir>**
 - cambia il direttorio a quello indicato
- **pwd**
 - mostra il nome del direttorio corrente
- **mkdir <dir>**
 - crea il direttorio indicato
- **rmdir <dir>**
 - cancella il direttorio indicato (deve essere vuoto)

I link

- `ln source alias`
 - crea un hard link
- `ln -s source alias`
 - crea un soft link

Le protezioni dei file

- Anche dette modi o permessi di accesso
- Tre controlli di base:
 - read (r) : permesso di lettura
 - write (w) : permesso di scrittura
 - execute (x) : permesso di esecuzione
- Tre partizioni degli utenti:
 - user (u) : il proprietario
 - group (g) : il gruppo
 - others (o) : gli altri

Modi speciali

- Tre modi speciali (per i file eseguibili):
 - setuid (s) : viene eseguito come se fosse root ad eseguirlo
 - setgid (S) : viene eseguito con lo stesso GID di root
 - sticky (t) : l'inaffondibile

Le protezioni dei direttori

- Le protezioni di una directory hanno un significato particolare:
 - x: attraversamento della directory
 - r: elenco dei file
 - w: creazione e/o cancellazione di file

Cambiamento di proprietario

- Per cambiare il gruppo dei file:


```
chgrp [-R] gruppo file
```
- Per cambiare il proprietario (ed eventualmente gruppo) dei file:


```
chown [-R] utente[.gruppo] file
```

```
chown [-R] uid[.gid] file
```
- In entrambi i casi, con l'opzione -R si opera recursivamente su tutti i file dei sottodirettori

Cambiamento di protezione

- Per cambiare le protezioni ai file:


```
chmod [-R] protezioni file
```
- Protezioni specificate in modo assoluto
 - un numero ottale di quattro cifre

sys	utente	gruppo	altri
4 2 1	4 2 1	4 2 1	4 2 1
s s t	r w x	r w x	r w x

Cambiamento di protezione (cont.)

- **Protezioni specificate in modo simbolico**
 - una stringa di tre caratteri
u(ser), **g**(roup), **o**(ther), **a**(ll)
 - +, -, =
 - s, S, t, r, w, x

Cambiamento di protezione (cont.)

- **Esempi:**
 - `chmod 777 xx`
 - ...

Protezioni standard

- **umask**
 - mostra (in forma assoluta) i permessi che sono negati quando si crea un file (la maschera delle protezioni).
- **umask maschera**
 - per definire la maschera delle protezioni

Indice

- **Introduzione**
- **Il file system**
- **Manipolazione di file e directory**
- **I comandi principali**

Il comando *ls*

- **Visualizza l'elenco dei file con le loro caratteristiche**
- **ls [-opzioni] [file ...]**
- **Opzioni:**
 - a: elenca anche i file che iniziano con .
 - l: output in formato esteso
 - g: include l'indicazione del gruppo
 - r: ordine inverso (alfabetico/temporale)
 - t: elenca i file in ordine temporale
 - R: elenca anche i file nei sottodirettori

ls - un esempio

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff   512 Sep  1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep  6 09:06 ..
-rw-r--r--  1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x---  2 maino staff   512 May 22 14:08 examples
-rw-----  1 maino staff 2416 Jun 30 15:24 gendata.c
-rw-----  1 maino staff  332 Jun 18 15:29 local.c
drwxr-xr-x  2 maino staff   512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw-----  1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff   70 Jun  2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May  6 14:20 random.c
-rw-r----- 1 maino staff   62 May  6 14:21 random.h
drwx----- 2 maino staff   512 May 25 14:36 testprof
```

ls - un esempio

Tipo di file

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

Protezioni

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

N° di link

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

Owner

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

Group

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

Dimensione

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

Data di creazione

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

ls - un esempio

Nome del file

```
# ls -alg ~/tmp
total 84
drwx----- 6 maino staff 512 Sep 1 16:14 .
drwxr-xr-x 19 maino staff 1024 Sep 6 09:06 ..
-rw-r--r-- 1 maino staff 1240 Jan 21 1992 AA.readme
drwxr-x-- 2 maino staff 512 May 22 14:08 examples
-rw----- 1 maino staff 2416 Jun 30 15:24 gendata.c
-rw----- 1 maino staff 332 Jun 18 15:29 local.c
drwxr-xr-x 2 maino staff 512 May 22 14:08 man
-rw-r----- 1 maino staff 27930 Mar 12 23:19 new.tex
-rw----- 1 maino staff 28077 Mar 12 22:52 numer.tex
-rw-r----- 1 maino staff 70 Jun 2 18:00 prova.tex
-rw-r----- 1 maino staff 1364 May 6 14:20 random.c
-rw-r----- 1 maino staff 62 May 6 14:21 random.h
drwx----- 2 maino staff 512 May 25 14:36 testprof
```

Visualizzazione di file testo

- Mediante un editor (es. vi o emacs)
- **cat file1 file2 ...**
 - concatena i file in output
- **head [-n] file ...**
 - visualizza le prime n righe

Visualizzazione di file testo (cont.)

- **tail [-n] [+n] [-rf] file ...**
 - Visualizza:
 - n: le ultime n righe
 - +n: tutto il file tranne le prime n righe
 - r: visualizza le righe in ordine inverso
 - f: rilegge continuamente il file

Visualizzazione una pagina per volta

- **pg file ...**
- **more file ...**
- **less file ...**

Visualizzazione una pagina per volta (cont.)

- **Comandi durante la visualizzazione:**
 - spazio prossima pagina
 - CR prossima riga
 - b pagina precedente
 - /pattern prossima pagina con pattern
 - ?pattern pagina precedente con pattern
 - q termina la visualizzazione

Occupazione di spazio su disco

- **df [-k] [disco ...]**
 - per controllare l'occupazione dei dischi
 - -k: occupazione in KB
- **du [-aks] direttorio ...**
 - per vedere lo spazio occupato da un direttorio e tutti i suoi sottodirettori
 - -a: occupazione di ciascun file
 - -s: solo il totale complessivo
 - -k: occupazione in KB

df: esempio

```
$ df
Filesys. 1024-bl.  Used      Av.   Cap. Mnt
/dev/hda3 199270   182354   6625  96%  /
/dev/hda1 61060    20967   36939  36%  /usr
/dev/hda4 199271   147953   41027  78%  /home/pc
```

Ricerca di un file

- **find direttorio espressione**
 - visita tutto l'albero sotto il direttorio specificato ed opera sui file che rendono vera l'espressione
- **esempio:**
 - find /users -name core

Elementi di ricerca

- **-name pattern**
 - attenzione: racchiudere il pattern tra apici se si usano espressioni regolari
- **-type tipo**
 - tipo del file = b(lock), c(haracter), d(irectory), l(ink), f(ile), s(ocket)
- **-user utente / -group gruppo**

Confronto di file

- **cmp file1 file2**
 - indica il byte e la riga in cui i file differiscono
- **comm [-123] file1 file2**
 - genera un output su tre colonne:
 - solo-in-file1 solo-in-file2 righe-comuni
 - le opzioni sopprimono la stampa delle colonne specificate
 - i file devono essere ordinati

Confronto di file

- **diff [-opzioni] file1 file2**
 - mostra le righe diverse, indicando quelle aggiunte (a), cancellate (d) e cambiate (c)
- **diff [-opzioni] dir1 dir2**
 - effettua il confronto tra tutti i file con lo stesso nome nei due direttori

Confronto di file

- **Opzioni:**
 - b: ignora gli spazi a fine riga, collassa gli altri
 - i: ignora la differenza maiuscolo/minuscolo
 - w: ignora completamente la spaziatura

Il sistema operativo LINUX Tools e comandi avanzati

Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

**Politecnico di Torino
Dip. Automatica e Informatica**

Indice

- **Alcuni comandi avanzati**
 - find
 - grep
 - tar, gzip, gunzip
- **I filtri**
 - sort
 - sed
 - awk

Indice

- **Alcuni comandi avanzati**
 - find
 - grep
 - tar, gzip, gunzip
- **I filtri**
 - sort
 - sed
 - awk

Elementi di ricerca

- **find <dir> [-opt]**
- **Alcune opzioni:**
 - -name pattern
Attenzione: racchiudere il pattern tra apici se si usano espressioni regolari
 - -type [b c d l]

Elementi di ricerca

- **find <dir> [-opt]**
- **Alcune opzioni:**
 - -name pattern
Attenzione: racchiudere il pattern tra apici se si usano espressioni regolari
 - -type [b c d l]

File a blocchi

Elementi di ricerca

- **find <dir> [-opt]**
- **Alcune opzioni:**
 - -name pattern
Attenzione: racchiudere il pattern tra apici se si usano espressioni regolari
 - -type [b c d l]

File a caratteri

Elementi di ricerca

- **find <dir> [-opt]**
- **Alcune opzioni:**
 - -name pattern
Attenzione: racchiudere il pattern tra apici se si usano espressioni regolari
 - -type [b c d l]

Directory

Elementi di ricerca

- **find <dir> [-opt]**
- **Alcune opzioni:**
 - -name pattern
Attenzione: racchiudere il pattern tra apici se si usano espressioni regolari
 - -type [b c d l]

Link

Uso avanzato di find

- Si può far eseguire un comando su tutti i file trovati specificando come ultimo elemento dell'espressione:
 - -exec comando \;
- In *comando*, usare \{\} per indicare il file corrente
- Esempio:
`find . -name core -exec rm \{\} \;`

Uso avanzato di find

- L'espressione di ricerca può contenere più condizioni usate per formare una condizione logica complessa
- Operazioni logiche:
 - AND: si elencano le condizioni una di seguito all'altra
 - OR: usare -o
 - NOT: usare !

Uso avanzato di find

- Usare le parentesi tonde per formare espressioni complesse
- Esempio:
`find . (-name core -o -size +2000)`

Indice

- **Alcuni comandi avanzati**
 - find
 - grep
 - tar, gzip, gunzip
- **I filtri**
 - sort
 - sed
 - awk

grep

- Per cercare se una stringa compare all'interno di un insieme di file, si può usare il comando:
grep [-opzioni] pattern files

grep

- **Opzioni:**

- -c conta le righe che contengono il pattern
- -i ignora la differenza maiuscolo/minuscolo
- -l elenca solo i nomi dei file contenenti il pattern (di default viene anche mandata in output la relativa riga)
- -n indica il numero d'ordine delle righe
- -v considera solo le righe che non contengono il pattern

Espressioni regolari in grep

- I pattern di ricerca in grep possono essere normali stringhe o espressioni regolari
- Alcuni caratteri hanno un significato speciale (a meno che siano preceduti da \)
 - . un carattere qualunque
 - ^ inizio riga
 - \$ fine riga

Espressioni regolari in grep (cont)

- * ripetizione (zero o più volte)
- + ripetizione (una o più volte)
- [] un carattere tra quelli in parentesi
- [^] un carattere esclusi quelli in parentesi
- \< inizio parola
- \> fine parola

Ricerca ricorsiva tra le directory

- Se si vogliono effettuare delle ricerche all'interno di un albero di directory si deve usare il comando:
- **find . -name "*" -exec grep 'pattern' \{\} \;**

Indice

- **Alcuni comandi avanzati**

- find
- grep
- tar, gzip, gunzip

- **I filtri**

- sort
- sed
- awk

tar

- **tar [option] files**
- **Accoda una lista di file in un unico tarfile ed effettua l'operazione inversa**

tar - Creazione

- **Opzioni per creare un tarfile:**
 - -c crea un nuovo tarfile
 - -f file specifica il nome del tarfile
 - -v Verbose
- **Esempi:**
 - tar -cvf /tmp/maino.tar /home/maino

tar - Estrazione

- **Opzioni per estrarre da un tarfile:**
 - -x Estrae i file dal tarfile
 - -t Testa il contenuto del tarfile
 - -f file Specifica il nome del file
 - -v Verbose
- **Esempi:**
 - tar -tvf /tmp/maino.tar
 - tar -xvf /tmp/maino.tar

gzip

- **gzip [opt] file**
- **Comprime un file**
- **Opzioni:**
 - -1 fastest
 - -9 max compression

gunzip

- **gunzip file**
- **Decomprime un file**

Indice

- **Alcuni comandi avanzati**
 - find
 - grep
 - tar, gzip, gunzip
- **I filtri**
 - sort
 - sed
 - awk

I filtri

- Un filtro è un programma che riceve i dati di ingresso da stdin e genera i suoi risultati su stdout
- I filtri sono molto utili in connessione con la ridirezione dell'I/O e con le pipe
- Esempi noti:
 - more, less
 - head, tail

Indice

- Alcuni comandi avanzati
 - find
 - grep
 - tar, gzip
- I filtri
 - sort
 - sed
 - awk

Ordinamento di dati

- **sort [-opzioni] [+inizio[-fine]] [file ...]**
- **Opzioni:**
 - -b ignora gli spazi iniziali
 - -n (modo numerico) considera le stringhe di cifre in modo numerico
 - -d (modo alfabetico) confronta solo lettere, cifre e spazi
 - -f ignora la differenza maiuscolo/minuscolo

Ordinamento di dati

- **sort [-opzioni] [+inizio[-fine]] [file ...]**
 - -o file scrive i dati ordinati in file (di default scrive su stout)
 - -r ordinamento inverso
 - -tcar separatore dei campi
 - inizio e fine indicano i campi da confrontare, nella forma n-campo[n-carattere]

Indice

- Alcuni comandi avanzati
 - find
 - grep
 - tar, gzip
- I filtri
 - sort
 - sed
 - awk

sed - Stream text EDitor

- E' un filtro in grado di modificare il contenuto di un file in base a certi criteri specificati in uno script
- **sed [-n] script [file ...]**
- **sed [-n][-e script][-f script_file][file ...]**
 - filtra stdin secondo quanto specificato in script
 - -n non ripete stdin su stdout

sed - Stream text Editor

- **Sintassi di script:**
 - [address[,address]] function [args]
 - address: numero di linea oppure una espressione regolare
 - function: comando che agisce sul pattern match
 - args: argomenti di function

sed functions

- **p** stampa la riga corrente
- **d** elimina la riga corrente
- **q** termina l'elaborazione
- **y/orig/subs/** trasforma i caratteri in orig con quelli in subs

sed functions

- **s/regexp/replace/flags**
sostituisce i pattern che soddisfano regexp con replace
- **flags:**
 - num sostituisce solo num occorrenze
 - g sostituisce tutte le occorrenze
 - p stampa la riga se ha sostituito

sed: esempi

- **sed '1,3 d' filename**
- **sed '3,\$ d' filename**
- **sed -n '/^pippo/ p' filename**
- **sed -f sedfile filename**

sed: esempi

- **sed '1,3 { s/^/Inizio:/ s/\$/ -- Fine/ }**
- **sed '3,\$ d' filename**
- **sed -n '/^pippo/ p' filename**
- **sed -f sedfile filename**

sed: esempi

- **sed '1,3 { s/^/Inizio:/ s/\$/ -- Fine/ }**
- **sed '3,\$ d' filename**
- **sed -n '/^pippo/ p' filename**
- **sed -f sedfile filename**

Uno Due Tre
A B C
/* Commento */

sed: esempi

- `sed '1,3 { s/^/Inizio:/ s/$/ -- Fine/ }`
- `sed '3,$ d'`
- `sed -n '/^appo/ p' filename`
- `sed -f sedfile filename`

Uno Due Tre
A B C
/* Commento */

Inizio: Uno Due Tre --Fine
A B C

Indice

- **Alcuni comandi avanzati**
 - find
 - grep
 - tar, gzip
- **I filtri**
 - sort
 - sed
 - awk

Origini

- awk è stato inventato nel 1977 da
 - A. V. Aho
 - P. J. Weinberger
 - B. W. Kernighan
- E' un linguaggio di elaborazione basato sulla ricerca di corrispondenze tra pattern

Funzionamento di base

- Per ciascuna linea del file aperto, si cercano dei record che corrispondono a un determinato pattern
- Quando viene trovata una corrispondenza, viene svolta una determinata azione
- La sintassi è simile a quella del C

Funzionamento di base

- Per ciascuna linea del file aperto, si cercano dei record che corrispondono a un determinato pattern
- Quando viene trovata una corrispondenza, viene svolta una determinata azione
- La sintassi è simile a quella del C

Non ci si deve preoccupare di aprire il file e di ciclare al suo interno

File di input

- Il file di input è organizzato in record (una riga del file)
- Un record è organizzato in campi (le varie parole della riga)

File di input (cont)

- **Variabili:**
 - RS: record separator
 - FS: field separator (lo spazio)
 - \$0: l'intero record
 - \$1: primo campo
 - \$n: n-esimo campo

Esecuzione

- E' possibile usare awk sia a linea di comando che tramite un file di script
- **Linea di comando:**
 - awk 'comando' < inputFile > outputFile
- **Script:**
 - awk -f scriptFile < inputFile > outputFile

Struttura dei comandi

- Ogni comando di awk è formato da un pattern e da un'azione

pattern {azione;}

Decide quando viene eseguita l'azione

Può essere formata da una o più istruzioni. E' eseguita solo se il pattern è vero

Struttura dei comandi (cont)

- Se il pattern non è inserito, viene eseguita sempre l'azione
- Se l'azione non è inserita, viene stampata la riga corrente

Esempi

- awk '\$1=="Ciao"' < inputFile
 - stampa tutte le righe in cui la prima parola è Ciao
- awk '{print \$1;}' < inputFile
 - stampa il primo campo di ogni riga del file

Formato dei pattern

- **I pattern possono essere:**
 - espressioni regolari
/ expr /
 - operatori di confronto
\$1 == "Ciao"
 - operatori relativi a intervalli
(cond1, cond2)

Espressioni regolari

- \ sequenza di escape
- ^ inizio della riga
- \$ fine della riga
- . un carattere
- [abc] uno dei caratteri
- [a-z] sequenza di caratteri
- [^abc] tutti i caratteri tranne a, b, c

Espressioni regolari (cont)

- uno|due soddisfatta da *uno* o *due*
- * 0 o più occorrenze del simbolo precedente
- + 1 o più occorrenze del simbolo precedente
- ? [AB]? È soddisfatta dalla stringa vuota o da A o da B

Espressioni regolari (cont)

- () combina espressioni regolari.
Cara(mella|bina) è soddisfatta da:
 - Caramella
 - Carabina

Espressioni regolari - Esempi

- /^(may)|(MAY)|(May))\$/ { print "Maggio"; }
- /^[Tt]itolo.* / { print "\nNuovo titolo."; }

Operatori di confronto

- == uguale
- < minore
- > maggiore
- <= minore o uguale
- >= maggiore o uguale
- != diverso
- ~ soddisfatto dall'espressione regolare
- !~ non soddisfatto dall'espressione regolare

Operatori di confronto (cont)

- && AND logico
- || OR logico
- ! NOT logico

Operatori di confronto - Esempi

- `$1 == "Bob" { print "Bob stuff"; }`
- `$1 !~ /[Mm]aggio/ { print "Non è maggio"; }`
- `($1 == "Bob") && ($2 ~ /[mM]*xy?RR$/) {
 print "Il primo campo è Bob. Il secondo
 è stato scritto da un ubriaco"; }`

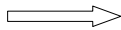
Operatori relativi a intervalli

- **cond1, cond2**
 - Il pattern diventa vero quando si verifica la prima condizione e rimane vero fino a quando si verifica la seconda condizione
 - Funziona solo con `nawk` o `gawk`

Operatori relativi a intervalli *(cont)*

- `$1=="1", $1=="CIAO" { print $2; }`

Prova ciao
1 numero uno
2 numero due
CIAO Finito
3 numero tre



numero
numero
Finito

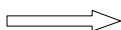
Pattern predefiniti

- **BEGIN:** eseguito prima dell'apertura del file di input
- **END:** eseguito dopo aver letto tutto il file di input

Esempio

- `BEGIN { FS=":"; }`
- `$1 ~ /[0-9]/ { print $3; }`

Prova:ciao
1:numero:uno
2:numero:due
CIAO:Finito
3:numero:tre



uno
due
tre

Azioni

- Uso delle variabili
- Stringhe
- Array
- Operatori
- Flusso condizionale
- Loop
- Input avanzato
- Funzioni
- Interazione con la shell

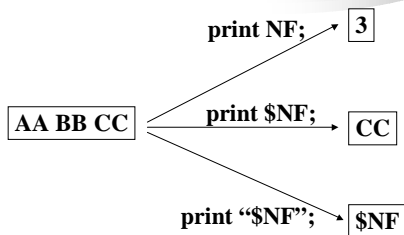
Variabili

- Possono essere utilizzate delle variabili
- Le variabili possono essere:
 - di campo, precedute dal \$ (\$0, \$n)
 - predefinite
 - definite dall'utente (non devono essere dichiarate)

Variabili predefinite

- ENVIRON array simbolico delle variabili d'ambiente. Esempio:
ENVIRON["PATH"]
- FS separatore del campo di input
- IGNORECASE 0 se si fa distinzione, 1
- NF numero di campi nel record
- NR numero di record già letti

Esempi



Operatori per le stringhe

- Concatenazione:
 - basta mettere una dietro l'altra le stringhe:
x="Ciao";
y="da me";
print x " " y;

Funzioni per le stringhe

- sub (reg, string, target), gsub (reg, string, target)
 - sostituisce nella stringa *target* la prima occorrenza (tutte se si usa *gsub*) della sottostringa che soddisfa l'espressione regolare *reg* con *string*
- length (s)
 - restituisce la lunghezza della stringa *s*
- match (string, reg)
 - restituisce la posizione in *string* della prima sottostringa che soddisfa l'espressione regolare *reg*

Funzioni per le stringhe

- printf (s, ...), sprintf (s, ...)
 - come in C
- split (string, vec, delim)
 - suddivide *string* negli elementi dell'array *vec* in base al delimitatore *delim*. Restituisce il numero di elementi in *vec*.
- substr (string, position, len)
- tolower (s), toupper (s)

Array

- **Con indice numerico:**
 - v[5]
 - i=3; v[i]="CIAO"
- **Con indice simbolico:**
 - giorni["gennaio"] = 31;
 - giorni["febbraio"] = 28;

Array (cont)

- **Esempio:**
 - giorni["gennaio"] = 31;
 - giorni["febbraio"] = 28;
- **Funzioni per vettori con indice simbolico:**
 - "gennaio" in giorni ==> restituisce true
 - delete giorni["marzo"]

Array multidimensionali

- **Con indice numerico:**
 - V[5,3]
- **Con indice simbolico:**
 - V["abc", "def"]
- **Con indice misto:**
 - V[1999, "gennaio"]

Operatori

- $x+y$, $x-y$, $x*y$, x/y
- x^y , $x\%y$
- $++x$, $x++$, $--x$, $x--$
- $\sin(x)$, $\cos(x)$
- $\text{rand}()$ (numero casuale tra 0 e 1)
- $\text{sysime}()$ (numero di secondi dal 1° gennaio 1970)

Flusso condizionale

- **if (cond) {**
 istruzione vera
} else {
 istruzione falsa
}
- **condizione ? istruzioneVera : istruzioneFalsa;**

Loop

- **do {**
 istruzione
} while (condizione)
- **for (init; cond; op) {**
 istruzione
}
- **for (i in array) {**
 istruzione
}

Loop (cont)

- **while (condizione) {**
 istruzione
 }

Input avanzato

- **exit**
 - emula la fine del file (se è presente il pattern END, viene eseguito)
- **getline**
 - legge una nuova linea e la copia in \$0.
 - Restituisce 1 se la lettura ha avuto successo.

Funzioni

- **Funzionano con nawk o gawk**
- **function myFunc (parametri)**
 {
 ...
 return x;
 }

Interazione con la shell

- **systyem (shellCommand);**
 - avvia una shell ed esegue il comando specificato
- **utilizzando la redirectione**
 - print “Ciao” > “nomeFile.txt”;
 - print myVar | “more”;

Esempio

- **Bookmark file:**

+|categoria
url1|descrizione
url2|descrizione
...

+|Università
<http://www.polito.it>|Politecnico di Torino
<http://www.unito.it>|Università di Torino
+|Musica
<http://www.mp3.com>|Sito per mp3

Esempio

```
<HTML><HEAD><TITLE>My bookmarks</TITLE></HEAD>
<BODY>
<H1>Università</H1><BR><UL>
<LI>  <A HREF="http://www.polito.it">
    Politecnico di Torino</A></LI>
<LI>  <A HREF="http://www.unito.it">
    Università di Torino</A></LI>
</UL>
<H1>Musica</H1><BR><UL>
<LI>  <A HREF="http://www.mp3.com">
    Sito per mp3</A></LI>
</UL>
</BODY>
</HTML>
```

Intestazione

```
BEGIN {  
    FS = "|";  
    firstTime = 1;  
    printf("<HTML><HEAD><TITLE>");  
    printf("My bookmarks</TITLE></HEAD>");  
    printf("\n<BODY>\n");  
}
```

Titolo

```
$1 == "+" {  
    if (firstTime == 0) {  
        print "</UL>";  
    } else {  
        firstTime = 0;  
    }  
    print "<H1>" $2 "</H1><BR><UL>";  
}
```

URL e fine

```
$1 != "+" {  
    print "<LI><A HREF=\"\" $1 \"\" >\" $2 \"</A></LI>";  
}  
END {  
    print "</UL></BODY></HTML>"  
}
```

Il sistema operativo LINUX *Script di Shell*

Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

Politecnico di Torino
Dip. Automatica e Informatica

File di comandi (script)

- **E' possibile memorizzare in un file una serie di comandi, eseguibili richiamando il file stesso:**
 - Esecuzione indiretta:
`source <scriptname> <args>`
 - Esecuzione diretta eseguendo lo script
 - E' necessario che abbia il permesso di esecuzione
 - La prima riga del file inizia con #! seguita dal nome (con il path assoluto) della shell con cui si devono eseguire i comandi

Indice

- Variabili
- Caratteri particolari
- Condizioni
- if, for, while, case
- Istruzioni particolari
- Segnali

Indice

- Variabili
- Caratteri particolari
- Condizioni
- if, for, while, case
- Istruzioni particolari
- Segnali

Variabili

- | | |
|---------------------------|---------------------|
| • TCSH | • BASH |
| – set myVar = ciao | – myVar=ciao |
| – set myVar = "Una prova" | – myVar="Una prova" |
| – set myVar = \$otherVar | – myVar=\$otherVar |

Variabili predefinite

- **Parametri della linea di comando**
 - \$1 primo parametro
 - \$# numero di parametri
 - \$0 nome dello script di shell
 - \$* stringa con tutti i parametri
 - \$?var 1 se var è stato definito
 - \$#var il numero di elementi in var (per array)
 - \$? Exit code dell'ultimo programma eseguito
 - \$\$ PID del processo

Variabili vettoriali

- **Sono previste variabili vettori:**
 - definizione enumerando i valori tra parentesi tonde
 - accesso con la notazione del C (ossia parentesi quadre); è possibile specificare degli intervalli

Variabili vettoriali - Esempio

```
set v = (a b c)
echo $?v      1
echo $#v      3
echo $v[2]    b
echo $v[1-2]  a b
unset v
echo $#v      Undefined variable: v
```

Indice

- Variabili
- Caratteri particolari
- Condizioni
- if, for, while, case
- Istruzioni particolari
- Segnali

Caratteri particolari (“)

- “...” identifica una stringa. All'interno vengono espanso le variabili
 - myVar = “Ciao”
 - echo “Contenuto di myVar: \$myVar”
- ⇒ Contenuto di myVar: Ciao

Caratteri particolari (‘)

- ‘...’ identifica una stringa. All'interno non vengono espanso le variabili
 - myVar = “Ciao”
 - echo ‘Contenuto di myVar: \$myVar’
- ⇒ Contenuto di myVar: \$myVar

Caratteri particolari (\)

- \ identifica il carattere di escape
 - myVar = \\$Ciao
 - echo “Contenuto di myVar: \$myVar”
- ⇒ Contenuto di myVar: \$Ciao

Caratteri particolari (`)

- `...` **identifica la sostituzione di comando**
 - myVar = `echo \$LOGIN_NAME`
 - echo "Contenuto di myVar: \$myVar"
- ⇒ Contenuto di myVar: dinatale

Indice

- Variabili
- Caratteri particolari
- Condizioni
- if, for, while, case
- Istruzioni particolari
- Segnali

Condizioni

- | | |
|--|--|
| <ul style="list-style-type: none"> • TCSH <ul style="list-style-type: none"> – (cond) <pre> str1 = "abc" str2 = "abd" if (str1 == str2) then echo "str1 uguale a str2" else echo "str1 diversa da str2" endif </pre> | <ul style="list-style-type: none"> • BASH <ul style="list-style-type: none"> – test expr – [expr] <pre> str1 = "abc" str2 = "abd" if test str1 = str2 then echo "str1 uguale a str2" else echo "str1 diversa da str2" fi </pre> |
|--|--|

Condizioni

- | | |
|--|---|
| <ul style="list-style-type: none"> • TCSH <ul style="list-style-type: none"> – (cond) <pre> str1 = "abc" str2 = "abd" if (str1 == str2) then echo "str1 uguale a str2" else echo "str1 diversa da str2" endif </pre> | <ul style="list-style-type: none"> • BASH <ul style="list-style-type: none"> – test expr – [expr] <pre> str1 = "abc" str2 = "abd" if [str1 = str2] then echo "str1 uguale a str2" else echo "str1 diversa da str2" fi </pre> |
|--|---|

Confronto tra stringhe

- | | |
|--|---|
| <ul style="list-style-type: none"> • TCSH <ul style="list-style-type: none"> – == uguale – != diverso | <ul style="list-style-type: none"> • BASH <ul style="list-style-type: none"> – = uguale – != diverso – -n lunghezza > di 0 – -z lunghezza = 0 |
|--|---|

Confronto tra numeri

- | | |
|---|--|
| <ul style="list-style-type: none"> • TCSH <ul style="list-style-type: none"> – == uguali – >= maggiore o uguale – <= minore o uguale – != diversi – > maggiore – < minore <pre> if (n1 >= n2) then ... endif </pre> | <ul style="list-style-type: none"> • BASH <ul style="list-style-type: none"> – -eq uguali – -ge maggiore o uguale – -le minore o uguale – -ne diversi – -gt maggiore – -lt minore <pre> if [n1 -ge n2] then ... fi </pre> |
|---|--|

Operatori per file

- | | |
|------------------------|------------------------|
| • TCSH | • BASH |
| – -d directory | – -d directory |
| – -f file normale | – -f file normale |
| – -r permesso di read | – -r permesso di read |
| – -w permesso di write | – -w permesso di write |
| – -x permesso di exec | – -x permesso di exec |
| – -z dim(file) > 0 | – -s dim(file) > 0 |
| – -e il file esiste | |
| – if (-d myVar) then | – if [-d myVar] |
| ... | then |
| endif | ... |
| | fi |

Operatori logici

- | | |
|---------------|---------------|
| • TCSH | • BASH |
| – ! NOT | – ! NOT |
| – && AND | – -a AND |
| – OR | – -o OR |

Indice

- Variabili
- Caratteri particolari
- Condizioni
- if, for, while, case
- Istruzioni particolari
- Segnali

L'istruzione if

- | | |
|----------------------------|----------------------|
| • TCSH | • BASH |
| – if (espressione) then | – if [espressione] |
| istruzioni | then |
| else if (espressione) then | istruzioni |
| istruzioni | elif [espressione] |
| else | then |
| istruzioni | istruzioni |
| endif | else |
| | istruzioni |
| | fi |

L'istruzione for

- | | |
|--------------------------|-----------------------|
| • TCSH | • BASH |
| – foreach myVar (elenco) | – for myVar in elenco |
| istruzioni | do |
| end | istruzioni |
| | done |

L'istruzione for - Esempi

- | | |
|-----------------------|--------------------|
| • TCSH | • BASH |
| – foreach x (1 2 3 4) | – for x in 1 2 3 4 |
| echo \$x | do |
| end | echo \$x |
| – foreach x (`ls`) | done |
| mv \$x ../backup | – for x in `ls` |
| end | do |
| | mv \$x ../backup |
| | done |

L'istruzione while

- **TCSH**
 - while (espressione)
 istruzioni
end
- **BASH**
 - while [espressione]
 do
 istruzioni
done

L'istruzione while - Esempio

- **TCSH**
 - set ind = 0
 while (ind <= 15)
 touch "xxx\$ind"
 set ind = `expr \$ind + 1`
end
- **BASH**
 - ind = 0;
 while [ind -le 15]
 do
 touch "xxx\$ind"
 ind = `expr \$ind + 1`
done

L'istruzione case

- **TCSH**
 - switch (str)
 case str1 | str2:
 istruzioni
 breaksw
 default:
 istruzioni
 breaksw
endsw
- **BASH**
 - case str in
 str1)
 istruzioni ;;
 str2 | str3)
 istruzioni ;;
 *)
 istruzioni ;;
esac

L'istruzione case - Esempio

- **TCSH**
 - switch (\$1)
 case 01 | 1:
 echo "Gennaio"
 breaksw
 ...
 default:
 echo "Non valido"
 breaksw
endsw
- **BASH**
 - case \$1 in
 01 | 1) echo "Gennaio";;
 02 | 2) echo "Febbraio";;
 ...
 12) echo "Dicembre";;
 *) echo "Non valido";;
esac

Indice

- Variabili
- Caratteri particolari
- Condizioni
- if, for, while, case
- Istruzioni particolari
- Segnali

Input interattivo

- read var1 var2 ... varn
- `#!/bin/bash`
 - echo "Scrivi qualcosa: "
read one two other
echo "La prima parola è: \$one"
echo "La seconda parola è: \$two"
echo "Il resto della linea è: \$other"

Operazioni aritmetiche

- Solo per bash
- `$(expr)`
- `num1 = 5`
- `num2 = $$(num1*3+1)$`

Redirezione I/O con bash

- `>` stdout su file
- `2>` stderr su file
- `&>` stdout+stderr su file
- `>>` stdout appeso a file
- `<` stdin da file

Modificatori di variabili per tcsh

- `:r`
il nome di un file senza estensione
- `:e`
l'estensione di un file
- `:h`
la "testa" di un nome di file (il path)
- `:t`
la "coda" di un nome di file (no path)

Esempio

- `foreach i ($*)`
 `echo "Estensione: "$i:e`
 `echo "Nome senza estensione: "$i:r`
 `echo "Path: "$i:h`
 `echo "Nome senza path: "$i:t`
 `end`
 `exit 0`

Indice

- Variabili
- Caratteri particolari
- Condizioni
- `if`, `for`, `while`, `case`
- Istruzioni particolari
- Segnali

Segnali

- I programmi possono ricevere dei segnali
 - Es: quando si preme CTRL-C il programma viene interrotto. In realtà, quando il sistema operativo riceve dalla tastiera l'informazione della pressione di CTRL-C, invia un segnale al processo
- Per inviare un segnale, si usa il comando `kill`
 - `kill -sig pid`

Segnali possibili

Signal	Value	Notes
interrupt	2	generato quando si preme CTRL-C
kill	9	terminazione forzata del processo
alarm	14	generato alla fine della system call alarm()

Comportamento dei processi

- Quando un processo riceve un segnale, può:
 - ignorare il segnale (tranne che per il segnale kill)
 - accettare l'azione di default del segnale (comportamento di default)
 - eseguire del codice particolare per gestire quel segnale

Gestione dei segnali negli script

- Per gestire i segnali, si usa il comando trap:
- trap "command" signal_list
 - esegue l'azione espressa in *command*, quando si genera uno dei segnali nella lista
- trap signal_list
 - ripristina il comportamento di default
- trap "" signal_list
 - ignora il segnale

Esempio

- In una porzione dello script di seguito viene utilizzato un file temporaneo:
- trap "rm -f /var/tmp/file.tmp; exit 0" 2
touch /var/tmp/file.tmp
...
trap 2
parte finale dello script

Il sistema operativo Linux Amministrazione del sistema

Giorgio Di Natale <dinatale@polito.it>
Stefano Di Carlo <dicarlo@polito.it>

**Politecnico di Torino
Dip. Automatica e Informatica**

Indice

- **Gestione degli utenti**
 - Cambiare password
 - Aggiungere un utente
- **Installare linux**
 - Gestione del disco (partizioni, fdisk, lilo)
 - Installazione del sistema
 - Ricompilazione del kernel

Indice

- **Gestione degli utenti**
 - Cambiare password
 - Aggiungere un utente
- **Installare linux**
 - Gestione del disco (partizioni, fdisk, lilo)
 - Installazione del sistema
 - Ricompilazione del kernel

passwd

- **Modifica della password di un utente**
 - passwd [login_name]
- **utenti attivi:**
 - who

```
root  tty1  Jul 3 19:25
scott ttym0 Jul 3 20:06
(remote.domain.com)
```

Indice

- **Gestione degli utenti**
 - Cambiare password
 - Aggiungere un utente
- **Installare linux**
 - Gestione del disco (partizioni, fdisk, lilo)
 - Installazione del sistema
 - Ricompilazione del kernel

Creare nuovi utenti

- **Passi da eseguire per creare un nuovo utente:**
 - Aggiungere un record in /etc/passwd
 - Creare la home directory
 - Assegnare la home all'utente
 - Assegnare i permessi di accesso
 - Definire i gruppi di appartenenza in /etc/group

/etc/passwd

- Il file **/etc/passwd** contiene la lista degli utenti del sistema, con i relativi dati:
 - login:pass:uid:gid:user's name:home_dir:shell
- **Esempi:**
 - root:P5q015:0:3::/root:/bin/sh
 - dinatale:xy:10:20:Giorgio:/home/dinatale:/bin/tcsh

/etc/passwd

- Il file **/etc/passwd** contiene la lista degli utenti del sistema, con i relativi dati:
 - login:pass:uid:gid:user's name:home_dir:shell
- **Esempi:**
 - root:P5q015:0:3::/root:/bin/sh
 - dinatale:xy:10:20:Giorgio:/home/dinatale:/bin/tcsh

ATTENZIONE: il campo *uid* deve essere univoco

Home directory

- **mkdir /home/dinatale**
- **chown dinatale. /home/dinatale**
- **chmod 700 /home/dinatale**

Home directory

- **mkdir /home/dinatale**
- **chown dinatale. /home/dinatale**
- **chmod 700 /home/dinatale**

Il punto significa che **chown** modifica anche il gruppo della directory, utilizzando quello definito in **/etc/passwd**

/etc/group

- **group:pwd:gid:user_list**
- **Esempi:**
 - root::0:root
 - user::20:root,dinatale
 - webadmin::100:

Indice

- **Gestione degli utenti**
 - Cambiare password
 - Aggiungere un utente
- **Installare linux**
 - Gestione del disco (partizioni, fdisk, lilo)
 - Installazione del sistema
 - Ricompilazione del kernel

Partizioni

- Linux richiede almeno due partizioni
 - sistema
 - swap
- Linux è in grado di gestire più sistemi operativi

Scenario

/dev/hda1	Windows	}	Partizione Windows
/dev/hda2	Linux root		
/dev/hda3	Linux /home	}	Linux (utenti)
/dev/hda4	swap		
			Linux (swap), 64MB

Strumenti per il partizionamento

- fdisk (dos)
- fdisk (linux)
- fips
- Partition Magic

Formattazione

- mkswap -c /dev/hda4 33280
- swapon /dev/hda4
- mke2fs -c /dev/hda2 1200000
- mke2fs -c /dev/hda3 1600000

Il file system logico e quelli fisici

- Il file system è in realtà *un insieme* di file system
- I file system possono essere "collegati" in modo logico. Questo collegamento è detto `mount`
- Il `mount` permette di vedere file system diversi (anche su dischi fisici diversi) come un unico albero di direttori (unica root)
- Il `mount` di un file system può essere cancellato (`umount`).

Mount: sintassi

- `mount -t <type> <device> <directory>`
- Esempio:
 - `mount -t msdos /dev/fd0 /mnt/dos`

Linux Loader (LiLo)

- Consente il bootstrap con Linux o Windows
- Configurazione in /etc/lilo.conf

```
# Global
boot = /dev/hda; delay = 50
# DOS section
other = /dev/hda1; label = DOS; table = /dev/hda
# LINUX
image = /vmlinuz; root = /dev/hda3
label = linux; read-only
```

Indice

- **Gestione degli utenti**
 - Cambiare password
 - Aggiungere un utente
- **Installare linux**
 - Gestione del disco (partizioni, fdisk, lilo)
 - Installazione del sistema
 - Ricompilazione del kernel

Installare Linux

- Per il bootstrap sono necessari due dischetti:
 - Il boot disk contiene il kernel del sistema operativo
 - Il root disk contiene l'immagine della partizione di root necessaria per l'installazione

Installare Linux (cont)

- È importante scegliere le immagini giuste per il proprio hardware
- Le distribuzioni di Linux forniscono programmi (MSDOS) per creare boot e root disk (ad esempio rawrite.exe)

Installare Linux (cont)

- Si esegue il bootstrap del sistema usando il boot disk ed il root disk
- Si ottiene una copia minimale di Linux che usa un root filesystem interamente in RAM
- Utilizzando l'utente root si può partizionare l'hard disk
- Si procede quindi al setup del sistema

Indice

- **Gestione degli utenti**
 - Cambiare password
 - Aggiungere un utente
- **Installare linux**
 - Gestione del disco (partizioni, fdisk, lilo)
 - Installazione del sistema
 - Ricompilazione del kernel

Modifica dei sorgenti

- E' possibile modificare il sorgente del kernel di Linux
- E' un'operazione critica e richiede molta esperienza

Configurazione del sistema

- In questa fase si dice quali componenti devono essere inclusi nel sistema (es: scheda audio, tipo di scheda video, ...)
- Questa operazione consente di settare alcune #define nei sorgenti
- Quando il sistema sarà compilato, verranno inseriti solamente i driver richiesti

Configurazione del sistema (cont)

```
cd /usr/src/linux
make menuconfig
[ make xconfig ]
```

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Crea i vari makefile
nelle sottodirectory

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Vengono cancellati tutti
i file binari rimasti da
precedenti compilazioni

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Compila il sistema operativo

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Il formato di questo file è più compresso rispetto al precedente

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Compila i moduli

Compilazione

```
cd /usr/src/linux
make dep
make clean
make zImage
[ make bzImage ]
make modules
make modules-install
```

Installa i moduli

Risultato della compilazione

- Se non si sono verificati errori, il file binario è creato nella directory:

```
/usr/src/linux/arch/i386/boot/zImage
```

Installazione

- Per rendere attivo il nuovo kernel, si deve usare lilo
- Si deve modificare il file di configurazione lilo.conf

```
# LILO Configuration file
# Global section
boot = /dev/hda
delay = 50
# New Linux Kernel
image = /vmlinuz
root = /dev/hda3
label = linux
read-only
# Old Linux Kernel
image = /vmlinuz.old
root = /dev/hda3
label = linuxold
read-only
```

Lilo.conf

Attivazione del nuovo sistema

- `mv /vmlinuz /vmlinuz.old`
- `cp /usr/src/.../zImage /vmlinuz`
- `lilo`