

Corso di Linux – Parte I

Alessio Pennasilico

<alessio.pennasilico@alba-networks.it>

Sabato 20 Marzo 2004



Introduzione

- Finalità del Corso
- Tempi e modi del corso
- Presentazione dei partecipanti

Finalità del corso

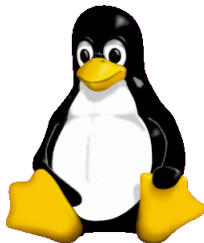
- Avere una prima infarinatura sulla natura di Linux ed ottenere le prime nozioni per un corretto approccio all'utilizzo di questo sistema.
- Non si vuole fare del semplice nozionismo inserendo un lungo elenco di comandi e di loro opzioni.
- Per poi ottenere maggiori informazioni vi invito a digitare “man <comando>” che fornisce una guida esaustiva a tutti gli usi ed a tutte le opzioni disponibili per quel comando.

Tempi e modi del Corso

- Il corso inizia alle 9:00 e termina indicativamente alle 12:00
- Ci saranno 2 pause di 10 minuti

Legenda

- Parleremo non solo di Linux, ma anche di OpenBSD: per questa ragione useremo dei simboli quando la sintassi del comando descritto si riferira' ad uno soltanto di questi due sistemi.



Linu

x



OpenBSD

Sommario

- Cosa e' Unix.
- Come nascono Linux ed OpenBSD.
- La shell ed i suoi principali comandi.
 - Gestire i file.
 - Gestire i processi.
- Il file system.

Cosa e' Unix

- Unix e' un sistema operativo nato negli U.S.A. negli anni '70 nei laboratori Bell.
- E' un sistema operativo multi-utente, multitasking.
- E' stato pensato per fare lavorare molti utenti da dumb-terminal su un grosso (per l'epoca ;P) mainframe.

SystemV vs. BSD

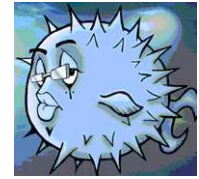
- Sono I due `unix flavours` piu' diffusi.
- SystemV e' la variante allo unix dei Bell Laboratories fatta da AT&T per i loro sistemi.
- BSD e' la variante allo unix di Bell fatta dall'Universita' di Berkley per essere ridistribuita tra i ricercatori.
- A questo link si puo' trovare uno schema completo, esaustivo, frequentemente aggiornato della storia dei diversi Unix <http://www.levenez.com/unix/>

SystemV

- E' il piu' commerciale, il piu' diffuso.
- Solaris e Linux Red Hat sono esempi di SystemV.

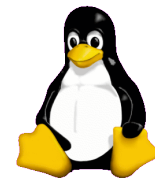


BSD



- E' sempre stato lo Unix piu' innovativo `by design`.
- OpenBSD e Linux Slackware sono esempi di OS che hanno adottato la filosofia BSD.
- La prima differenza che notiamo rispetto ai SysV e' il numero ed il tipo di file di inizializzazione del sistema.

Le distribuzioni



- Linux e' uno solo, ed e' quello che comunemente chiamiamo kernel.
- Esistono diverse aziende od associazioni che, a partire da quel kernel, hanno associato applicazioni, procedure di gestione e di installazione.
- Sempre di linux si tratta, ma diversi saranno gli strumenti a disposizione e la filosofia su cui tutto il sistema e' basato (workstation, server, etc).

Accedere al sistema

- L'accesso al sistema puo' avvenire in due modi: da locale o da remoto.
- Da locale si utilizzerà uno dei tty.
- Da remoto e' consigliato usare ssh.
- Dovremo scegliere con quale utente accedere.

I tty

- I tty sono 12, e si puo' passare da uno all'altro premendo ALT-Fx, dove x e' il numero del TTY.
- Non tutti I tty potrebbero essere attivi.

ssh

- ssh permette di accedere da remoto, dalla lan o tramite Internet, a seconda di come sono impostate le politiche di sicurezza.
- E' paragonabile al telnet, tuttavia la password e tutte le videate passano criptate, quindi protette.

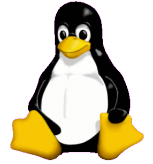
Utenti

- Quando accediamo possiamo scegliere se accedere come amministratore (root) o come semplice utente.
- E' bene lavorare normalmente come utente, ed utilizzare i privilegi amministrativi solo quando necessario (es. un erroneo comando di cancellazione impartito da un utente non puo' fare gravi danni, a differenza dello stesso comando dato da root).
- Questa e' una regola di carattere generale che non vale solo per i sistemi unix.

La shell

- La shell e' paragonabile al command.com di ms-dos.
- E' il nostro interprete dei comandi, la nostra interfaccia verso il sistema.
- Ne esistono di diversi tipi e con diverse caratteristiche.

bash



- La Bourne Again Shell (BASH) e' la shell tradizionalmente piu' usata sulle macchine Linux.
- E' una versione avanzata della Bourne Shell (sh).
- Si puo' comunque utilizzare su tutti gli unix.

ksh



- La Korn Shell (KSH) e' una versine avanzata di sh, tuttavia ha delle caratteristiche diverse da bash.
- Ci si accorge di questo solo utilizzando funzionalita' avanzate.
- OpenBSD utilizza di default csh, che spesso risulta molto scomoda (soprattutto per chi e' abilitato alle caratteristiche della bash): per questa ragione consiglio l'utilizzo di ksh su questo sistema operativo.

Gestione dei file

- Vediamo ora come gestire da command line tutti i nostri file e le nostre directory.
- Molti comandi purtroppo vanno ricordati a memoria, ma con la pratica si ricordano senza fatica.

Visualizzare il contenuto di una directory

- Sotto unix l'equivalente del comando dir e' ls. Le sue opzioni piu' utilizzate sono l ed a.
- ls visualizza i file.
- ls -a visualizza anche i file che iniziano per . (file o directory nascosti, di solito contengono impostazioni).
- ls -l visualizza il contenuto della cartella in 'verbose mode', mostrandoci tutti i dettagli relativi ai file.

ls

```
mayhem@coniglio:~$ ls
```

```
documents/      ipv6tunnel.pdf
```

```
mayhem@coniglio:~$ ls -a
```

```
.bash_history  .ssh/  
documents/    ipv6tunnel.pdf
```

```
mayhem@coniglio:~$ ls -l
```

```
drwxr-xr-x  20 mayhem  users           4096 Oct 13 02:10 documents/  
-rw-----   1 mayhem  users        209532 Oct 28 14:08 ipv6tunnel.pdf
```

I permessi

- Cosa significa:

```
drwxr-xr-x  mayhem  users  documents/
```

- mayhem e' il proprietario del file, users e' il gruppo a cui il file appartiene.
- L'altra stringa identifica i permessi sul file.

Proprieta' di un file/directory

- La stringa di 10 caratteri che definisce le proprieta' va cosi' scomposta:
- 1: - se si tratta di file, d se si tratta di directory, l se si tratta di un link
- 2,3,4: si riferisce al proprietario
- 5,6,7: si riferisce al gruppo
- 8,9,10: si riferisce a tutto il resto del mondo

rwX

- Ogni gruppo di 3 definisce il tipo di accesso all'oggetto da parte del soggetto definito:
- r: lettura
- w: scrittura
- x: esecuzione

Tirando le somme

- `-rw-r--r--`: e' un file, il proprietario (user) lo puo' leggere o scrivere, i membri del gruppo (group) lo possono leggere come tutti gli altri utenti (other).
- `drwxr-xr--`: user puo' entrare nella directory e scrivere, group puo' solo accedere in lettura, other non puo' nemmeno entrare nella directory.

chown

- Il comando `chown` ci permette di modificare il proprietario del file.

```
root@coniglio:~# ls -l
-rw----- 1 mayhem users 209532 Oct 28 14:08 ipv6tunnel.pdf
root@coniglio:~# chown alessio ipv6tunnel.pdf
root@coniglio:~# ls -l
-rw----- 1 alessio users 209532 Oct 28 14:08 ipv6tunnel.pdf
```

chgrp

- Il comando `chgrp` ci permette di modificare il gruppo di appartenenza del file.

```
root@coniglio:~# ls -l
-rw----- 1 alessio users 209532 Oct 28 14:08 ipv6tunnel.pdf
root@coniglio:~# chgrp staff ipv6tunnel.pdf
root@coniglio:~# ls -l
-rw----- 1 alessio staff 209532 Oct 28 14:08 ipv6tunnel.pdf
```

chmod

- Con chmod e' possibile cambiare i permessi ai file o alle directory.
- `chmod [u|g|o|a] +/- r|w|x`
- u=user, g=group, o=other, a=all

chmod: esempi

- `chmod u +w`: aggiunge il diritto di scrittura per l'utente.
- `chmod a +x`: aggiunge per tutti il diritto all'esecuzione.
- `chmod g -w`: rimuove il diritto di scrittura per il gruppo.

cp

- Copia un file da una posizione ad un'altra.

```
mayhem@coniglio:~$ cp ipv6tunnel.pdf ./documents
```

mv

- mv sposta un file da una posizione ad un'altra.

```
mayhem@coniglio:~$ mv ipv6tunnel.pdf ../shared
```

rm

- rm cancella un file.

```
mayhem@coniglio:~$ rm ../shared/ipv6tunnel.pdf
```

- NB: rm cancella solo i file, tuttavia con `rm -rf` (ricorsivo e forzato) puo' cancellare interi alberi di directory. Bisogna sempre prestare la massima attenzione nel suo utilizzo: perdere il disco intero e' facile, e spesso succede per colpa di uno spazio in piu' sulla riga di comando.

In

- In crea un link verso un file/cartella.

```
mayhem@coniglio:~/test$ ln -s ../ipv6tunnel.pdf ipv6.pdf
```

```
mayhem@coniglio:~/test$ ls -l
```

```
lrwxrwxrwx 1 mayhem users ipv6.pdf -> ../ipv6tunnel.pdf
```

Hard link

- In `ln -s` crea un “soft link”. Senza specificare `-s`, se si tratta di un file sullo stesso disco e' possibile creare un hard link.

```
mayhem@coniglio:~/test$ ln ../ipv6tunnel.pdf ipv6_cisco.pdf
mayhem@coniglio:~/test$ ls -l
lrwxrwxrwx  2  mayhem  users      17 Nov 15 01:18  ipv6_cisco.pdf
```

- Questo e' fisicamente lo stesso file.

File eseguibili

- Utilizzando unix non sono le estensioni che determinano se un file e' eseguibile o meno: e' l'attributo "x".
- Se creo uno script, mi bastera' un "chmod a +x file" per renderlo eseguibile.
- Se digito "chmod a -x file" rendo un programma binario, eseguibile, non piu' eseguibile per il sistema.
- Solo l'attributo "x" fa capire al sistema se un file e' eseguibile o meno.

Struttura del file system

■ mayhem@coniglio:~\$ ls -al /

total 148

```
drwxr-xr-x 17 root  root    4096 Oct 15 18:16 .
drwxr-xr-x 17 root  root    4096 Oct 15 18:16 ..
drwxr-xr-x  2 root  bin     4096 Sep  8 03:36 bin
drwxr-xr-x 14 root  root   40960 Nov 13 09:22 dev
drwxr-xr-x 32 root  root    4096 Nov 14 19:11 etc
drwxr-xr-x  6 root  root    4096 Sep  8 21:46 home
dr-xr-xr-x 117 root  root      0 Nov 13 10:21 proc
drwx--x--- 19 root  root    4096 Nov  8 15:58 root
drwxr-xr-x  2 root  bin     4096 Sep 11 03:58 sbin
drwxrwxrwt 34 root  root   40960 Nov 14 20:36 tmp
drwxr-xr-x 20 root  root    4096 Mar 10 2002 usr
drwxr-xr-x 18 root  root    4096 Mar 10 2002 var
```

/bin, /sbin

- Entrambe contengono file eseguibili.
- /bin contiene i comandi di sistema disponibili per tutti gli utenti (es. /bin/l`s`).
- /sbin contiene tutti i comandi di sistema a disposizione dell'amministratore (es. /sbin/r`oute`).

/dev

- I file contenuti nella /dev corrispondono fisicamente alle periferiche di sistema.

- Es. /dev/audio e' la sound blaster:

Dare il comando :

```
mayhem@coniglio:~$ cat song.au > /dev/audio
```

produce suoni (nella fattispecie la riproduzione di song.au) sulle nostre casse.

/dev/null

- Questo device corrisponde al nulla. Tutto ciò che vi viene inviato sparisce per sempre. Spesso viene utilizzato per mandare output dei processi o come link per far credere ai processi di scrivere regolarmente

/etc

- Contiene molti dei files di configurazione della nostra macchina.
- Dentro /etc troviamo diverse cartelle, per i diversi servizi che la nostra macchina offre (es. /etc/sshd).

/home, /root

- /home contiene tutte le home directory degli utenti del sistema.
- Es. /home/mayhem e' la "home directory" dell'utente mayhem del sistema locale.
- Root ha la sua home in /root, non in /home/root

/proc

- /proc non e' una directory che contiene file "reali", ma file che vengono creati dal sistema operativo e contengono informazioni circa lo stato e la configurazione della macchina.
- Cambiare un file nella /proc, spesso, corrisponde a cambiare la configurazione della macchina.
- Quasi tutti i comandi che forniscono statistiche di utilizzo si basano sulla /proc.

/tmp

- In questa directory sono contenuti i file temporanei generati dai processi di sistema e dei diversi utenti.

/usr

- Questa directory contiene quasi una copia esatta di /
- Sono i file che caratterizzano il sistema utilizzato (Linux, piuttosto che OpenBSD, Red Hat piuttosto che Slackware).
- Il senso delle diverse cartelle e' riconducibile a quanto gia' spiegato.

/usr/local/

- Di nuovo una copia di /
- Questa volta il contenuto e' relativo al nostro sistema.
- Nella /usr/local/bin troviamo infatti i file che l'amministratore della macchina ha installato in locale.

/var

- /var contiene tutti i file di sistema che cambiano “continuamente”.
- Es. La directory con i file di posta (/var/spool/mail), la directory con i log (/var/log).
- In /var/run, ad esempio, troviamo diversi file .pid, (es. syslog.pid) che hanno come contenuto il pid del processo relativo (es. syslogd).

Gestione dei processi

- Unix e' multitasking e multiutente: sullo stesso sistema possono girare contemporaneamente piu' servizi/ applicazioni, con proprieta' e diritti molto diversi tra loro.

ps

- ps elenca i processi della shell in uso.
- ps -aux mostra tutti i processi attivi del sistema e tutte le proprietà dei diversi processi.

```
mayhem@coniglio:~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	444	64	?	S	Nov13	0:05	init
root	2	0.0	0.0	0	0	?	SW	Nov13	0:01	[keventd]
root	44	0.0	0.1	1424	440	?	S	Nov13	0:00	/sbin/cardmgr
root	61	0.0	0.1	1296	436	?	S	Nov13	0:00	/usr/sbin/syslogd -r
postfix	165	0.0	0.3	2624	880	?	S	Nov13	0:01	qmgr -l -t fifo -u
mayhem	262	0.0	2.1	17924	5612	?	S	Nov13	0:14	evolution

kill

- kill permette di terminare istantaneamente una applicazione bloccata.
- Esistono diversi modi per killare un programma .
- Es. “kill -HUP pid”: chiude e riapre il processo.
- Es. “kill -9 pid”: termina il processo immediatamente.
- Di solito solo root o il proprietario possono killare un processo.

Processi interattivi

- Quando lanciamo un processo esso e' di default in foreground, in modo interattivo.
- Potremmo avere l'esigenza di lanciare processi molto lunghi e laboriosi che non richiedono pero' intervento umano, ad esempio creare un enorme archivio compresso di file.

Processi in background

- Abbiamo due modi per portare in background una applicazione:

- Sin dall'inizio (&):

```
mayhem@coniglio:~$ gunzip big_arch.tar.gz &  
[1] 11398  
mayhem@coniglio:~$
```

- Durante la sua esecuzione (^Z):

```
mayhem@coniglio:~$ gunzip big_arch.tar.gz  
[1]+  Stopped                  gunzip big_arch.tar.gz  
mayhem@coniglio:~$ bg  
[1]+ gunzip big_arch.tar.gz &  
mayhem@coniglio:~$
```

Elencare i processi

- Il comando `jobs` ci elenca quali processi abbiamo messo in background.

```
mayhem@coniglio:~$ jobs
```

```
[1]-  Running
```

```
yes >/dev/null &
```

```
[2]+  Running
```

```
while true; do
```

bg, fg, ^Z, ^Y

- Con fg %n possiamo riportare in modalita' interattiva un processo.
- Con ^Z sospendiamo un processo.
- Con ^Y sospendiamo un processo nel momento in cui richiede input dall'utente.
- Con bg %n possiamo riportare in esecuzione un processo sospeso.

Riavviare i processi

- Vorrei sottolineare che “kill -HUP” non e' il modo migliore, per quanto molto usato per riavviare un processo.
- Alcuni servizi non lasciano scelta, ad esempio syslogd.
- Di norma e' preferibile utilizzare i comandi proprii dell'applicazione:

```
root@coniglio:~# /usr/local/sbin/squid -k reconfigure
```

SU

- A volte l'utente con cui siamo collegati non ha i privilegi necessari per svolgere un compito.
- Per questa ragione e' possibile utilizzare su per "diventare" un altro utente, a patto di conoscerne la password.

```
mayhem@coniglio:~$ su - alessio
password:
alessio@coniglio:~$ su -
password:
root@coniglio:~# exit
alessio@coniglio:~$
```

Accensione e spegnimento del Sistema

- Solo root puo' spegnere o riavviare il sistema.

- Per spegnere il sistema:

```
root@coniglio:~# halt
```

- Per riavviare il sistema:

```
root@coniglio:~# reboot
```

pipe

- Pipe, il carattere “|” serve a dare in pasto l’output di un programma ad un altro.
- grep serve a filtrare sulla base di una stringa.

```
mayhem@coniglio:~$ ps aux | grep syslogd
root      61  0.0  0.1  1296  436 ?      S   Nov13   0:00 /usr/sbin/syslogd -r
mayhem 29299 0.0  0.1  1408  500 pts/7  S   02:00   0:00 grep syslogd
```

Redirezione dell'output

- “>” reindirige l'output di un comando.

```
mayhem@coniglio:~$ ps aux | grep syslogd > process_list.txt
```

```
mayhem@coniglio:~$ cat process_list.txt
```

```
root          61 0.0 0.1  1296  436 ?      S   Nov13   0:00 /usr/sbin/syslogd -r
mayhem 29299 0.0 0.1  1408  500 pts/7 S   02:00   0:00 grep syslogd
```

Le emissioni standard

- I processi conoscono 3 modi per comunicare con la shell:
- stdin
- stdout (1&)
- stderr (2&)

stdout e stderr

- Tutto l'output dei processi viene inviato a stdout, mentre le eccezioni vengono inviate a stderr.
- Es. `cp -vR /tmp /mnt/backup`
#tutto quel che il programma ha da dire viene mandato sul tty, che corrisponde sia allo stdout che allo stderr)

stdout e stderr

- `cp -vR /tmp /mnt/backup > log`

dentro log verra' inviato tutto lo stdout, mentre lo stderr verra' inviato al tty

- `cp -vR /tmp /mnt/backup 2>1& > log`

lo stderr viene reindirizzato dentro stdout, che a sua volta viene indirizzato nel file log

- `cp -vR /tmp /mnt/backup 2> err 1& > log`

tutto lo stderr viene scritto dentro il file err mentre lo stdout viene scritto nel file log

Variabili

Per le variabili si utilizza un nome maiuscolo; per richiamarle si premette un \$ al nome.

Il comando export mentre si creano le rende durature rispetto al sistema e non rispetto al processo (es. Un comando batch)

```
mayhem@coniglio:~$ export VAR0=ciao  
mayhem@coniglio:~$ echo $VAR0  
ciao
```

Espansione delle variabili

- Usare in uno script i doppi apici "", o i singoli apici ` , piuttosto che ' e ' diverso.

```
mayhem@coniglio:~$ VAR="date"
mayhem@coniglio:~$ echo $VAR
date
mayhem@coniglio:~$ VAR=`date`
mayhem@coniglio:~$ echo $VAR
Thu Nov 21 16:48:16 CET 2002
```

Accodare comandi

- Per immettere una serie di comandi da eseguire in sequenza si utilizza ;

```
mayhem@coniglio:~$ VAR=`date`; echo $VAR  
Thu Nov 21 16:51:10 CET 2002
```

cat e nomi case-sensitive

- cat e' l'equivalente del type di ms-dos.
- Da notare che per unix le lettere maiuscole o minuscole sono diverse:

```
mayhem@coniglio:~$ cat process_list.txt
root      61 0.0 0.1  1296  436 ?      S   Nov13   0:00 /usr/sbin/syslogd -r
mayhem 29299 0.0 0.1  1408  500 pts/7 S   02:00   0:00 grep syslogd
mayhem@coniglio:~$ cat Process_List.txt
cat: Process_List.txt: No such file or directory
```

Riferimenti

- <http://www.linuxdoc.org>
(Tutta la documentazione che serve per lavorare con Linux)
- <http://www.pluto.linux.it>
(How-to tradotti in italiano)
- <http://www.openbsd.org>
- \$ man :)

Disclaimer

- Queste slides sono realizzate da Alessio Pennasilico e sono soggette alla licenza GPL, sempre nella sua corrente versione, possono pertanto essere distribuite liberamente ed altrettanto liberamente modificate, a patto che se ne citi l'autore e la provenienza.
- Sarò lieto di ricevere domande, suggerimenti, correzioni al mio indirizzo di e-mail, cso@alba-networks.it