Corso Base su Linux

Basato su Fedora 7
Lezione 3

Multiutenza

Linux è un sistema operativo multiutente

- Utenti diversi possono avere accesso al sistema e alle risorse ad esso connesse simultaneamente
- Accesso diretto: tramite console
- Accesso remoto: via rete.
- Ad ogni utente viene assegnato un identificativo (nome utente) ed una password.
- I dati, i programmi e le impostazioni sono completamente separati tra utenti diversi.
- La sicurezza è garantita da meccanismi di protezione basati su ACL (Access Control List) che permettono di limitare l'accesso ai files.

Login e Logout

Login

E' l'operazione di autenticazione, tramite nome utente e password, che permette all'utente l'accesso alle risorse del sistema.

L'accesso può avvenire in modalità testuale (shell) oppure in modalità grafica

Il login può essere eseguito da locale o da remoto.

Lo scopo è di verificare che l'utente abbia i requisiti per accedere al sistema o ad un suo servizio e metterlo in condizione di interagire con la macchina.

Logout

E' il processo opposto del login: chiude la shell (o l'ambiente grafico) aperta con il precedente login e tutti i programmi lanciati dall'utente.

Fa in modo che l'utente debba ripetere il login per accedere nuovamente al sistema.

La Shell

La shell è l'interfaccia testuale tramite la quale l'utente può operare sul sistema.
Viene lanciata dal processo di login dopo che il processo di autenticazione è andato a buon fine.

La shell è un programma che gestisce la comunicazione fra utente e sistema operativo interpretando ed eseguendo i comandi dell'utente.

Può avere diversi utilizzi:

- Uso interattivo: il sistema attende i comandi digitati dall'utente, che possono redirezionare input ed output;
- Configurazione: definire variabili e parametri che vengono utilizzati in ogni interazione dell'utente con la macchina;
- Programmazione: utilizzando comandi di sistema e funzionalità della shell è possibile realizzare piccoli programmi (script shell) in grado di automatizzare operazioni e reagire ad eventi.

Quale Shell ?

Esistono molteplici shell, ognuna delle quali presenta caratteristiche e peculiarità proprie, di seguito sono riportate le più utilizzate in ambiente Unix.

sh	Bourne shell, è disponibile su qualsiasi ambiente UNIX, quindi è la più utilizzata per creare script shell compatibili e cross-platform.
csh	C shell, prende il nome dal linguaggio di programmazione, ovviamente le funzionalità di tale shell derivano in modo diretto dal C.
bash	Bourne Again Shell. Una delle ultime nate, offre le stesse capacità della C shell, con l'aggiunta di alcune funzionalità come l'history dei comandi e la TAB Completion
ksh	Korn shell. Largamente diffusa è compatibile con la sh sulla parte di scripting ed ha tutte le funzionalità di interazione della csh. http://www.kornshell.com/
tcsh	E' un'evoluzione della csh, con cui mantiene piena compatibilità e introduce feature come command line editing e name completion.

Sulla gran parte delle distribuzioni Linux è preimpostata di default la shell bash.
La sintassi dei comandi presenti in questo corso è basata sulla bash.

BASH

Bash acronimo di Bourne Again SHell, è la shell di gran lunga più utilizzata in ambiente Linux.

Alcune sue caratteristiche (presenti anche in altre shell):
- Possibilità di editare la command line
- Autocompletion dei comandi usando il tasto TAB
- Possibilità di definire alias per i comandi
- History infinita (o quasi) dei comandi inseriti
- Funzionalità di scripting e funzioni condizionali e di ciclo.
- Possibilità di definire funzioni e alias
- Possibilità di gestire array indicizzati di dimensioni infinite
- Gestione e controllo dei job
- Espressioni aritmetiche
- Caratteri jolly (metacaratteri) nella gestione dei nomi di file

Nel seguito utilizzeremo i principali comandi della shell, ma l'unico vero modo per conoscere la bash è **usarla**.

Sui sistemi Linux viene lanciata automaticamente dopo il login, alternativamente basta scrivere **bash** (trovandosi in un'altra shell) per eseguirla.

BASH - Prompt

Il prompt dei comandi si presenta, in genere, con la seguente forma:

[username@hostname ~]$ (prompt utente)

Nel caso dell'utente root il carattere finale **$** è sostituito con un **#** per ricordare che si ha il controllo completo sul sistema ed un comando errato può creare problemi rilevanti.

[username@hostname ~]# (prompt root)

Il carattere *tilde* (~) denota la ***home directory***, cioè la directory di lavoro dell'utente, nella quale ci si trova dopo aver eseguito il login.

Quando si esegue un cambio directory la *tilde* viene sostituita dal nome della directory stessa.

[username@hostname Desktop]$

BASH – Personalizzazione del prompt

Cambiando il valore della variabile d'ambiente PS1, si può personalizzare il prompt della shell.

Per modificare il contenuto di una variabile, basta digitarne il nome seguito dal segno uguale (=) e dalla stringa che deve sostituire l'attuale contenuto.

$ PS1='Mio prompt \W $:' [INVIO] => Mio prompt ~ $:

Esistono alcuni caratteri speciali che permettono di ottenere un particolare output:

Carattere speciale	Output di testo
\d	Data corrente
\h	Hostname
\w	Path corrente
\W	Directory attuale
\u	Username
\t	Ora corrente nel formato 24 ore
\@	Ora corrente nel formato 12 ore

Caratteri speciali

Stringhe

Alcuni caratteri sono riservati e hanno un significato particolare per la shell, per poter trasferire uno di questi caratteri ad un comando come argomento è necessario racchiuderlo tra apici (').

Una qualunque sequenza di caratteri racchiusa tra apici è una **STRINGA.**

'mia stringa'

Qualora la stringa contenga un apice al suo interno è possibile racchiuderla tra due caratteri **doppio apice** (**“**).

“l'aurora”

Per passare una stringa contenente caratteri speciali ad un comando è necessario farla precedere dal carattere **$**

Sequenze di Escape

Alcuni caratteri speciali devono essere scritti come sequenze di codici Escape.

In tabella sono riportati i principali:

Sequenza di Escape	Descrizione
\a	Avviso (fa suonare il campanello di sistema)
\b	Backspace
\e	Escape
\f	Avanzamento foglio
\n	Nuova riga
\r	Ritorno a capo
\\	Barra rovesciata
\NNN	Carattere il cui codice ASCII è NNN (in base 8)

```
$ echo $'Prima riga\nSeconda riga\n'
```

Si ricordi di anteporre il carattere $

Cronologia e Line editing

La shell bash ha la proprietà di conservare la *cronologia* dei comandi impartiti in un elenco *sequenziale*.

Vengono memorizzati i comandi della sessione corrente e delle precedenti.

Per impostazione predefinita la bash conserva 500 eventi, ma è un parametro che pùò essere variato.

La cronologia è memorizzata in un file di testo all'interno della **home** chiamato **.bash_history**

Per ripetere il comando precedente è sufficiente premere il tasto “freccia su” (↑).

Usando i tasti cursore su (↑) e giù (↓) ci si muove tra i comandi precedentemente impartiti.

Quando viene evidenziato il comando cercato, si può editare spostandosi all'interno della riga con i tasti cursore sinistra (←) e destra (→) .

Cronologia uso avanzato

Il comando **history** visualizza l'intero elenco della cronologia dei comandi.
E' di aiuto quando l'elenco è lungo e l'uso dei tasti cursore richiederebbe molto tempo per individuare un evento dato diversi comandi fa.

```
$ history [INVIO]
1 who
2 ls
3 ps
4 history
$
```

Se l'elenco supera la lunghezza di una pagina è possibile usare il comando:

```
$ history | more
```
(è necessario premere un tasto per avanzare)

e per cercare un particolare comando si può usare:

```
$ history | grep ls
```
(filtra le righe che contengono la parola “ls”)

```
2 ls
$
```

Cronologia uso avanzato

Conoscendo il numero dell'evento è possibile eseguirlo digitando il punto esclamativo (!) seguito dal numero dell'evento:

$!<numero evento> [INVIO]

Premendo i tasti <CTRL>+r è possibile eseguire una ricerca inversa all'interno della cronologia:

$ <CTRL>+r
(reverse-i-search)'': grep

Una volta impostata la stringa di ricerca per trovare gli altri elementi contenuti nella cronologia bisogna ripetere la pressione dei tasti <CTRL>+r.

Ridirigere Input ed Output

La shell sposta il testo in “stream”, ovvero in flussi.

Sono presenti tre flussi:

Standard output:	costituito dallo schermo del terminale
Standard input:	di default è la tastiera.
Standard error:	è un flusso specifico per i messaggi di errore normalmente indirizzato anche esso allo schermo.

Ognuno di questi flussi può essere rediretto ad un file o anche ad un altro comando.

Reindirizzamento dei flussi ad un file

Standard input: si usa l'operatore minore **'<'**.

```
$ apropos < my_keywords [INVIO]
```

Standard output: si usa l'operatore maggiore **'>'**.

```
$ ls > my_dir [INVIO]
```

Standard error: si usa l'operatore **'2>'**.

```
$ gcc -c my_program.c  2> my_errors  [INVIO]
```


Ridirigere Input ed Output

Ridirigere l'output all'input di un altro comando

Tale operazione è detta ***piping***, e si verifica quando si connette lo **standard output** allo **standard input** di un altro comando.

Questa operazione viene eseguita specificando i due comandi in sequenza, separati dal carattere barra verticale **"|"** (detto ***pipe***).

I comandi così costituiti prendono il nome di *pipeline*

esempio:

```
$ ls -l | more
```

In questo modo lo *standard output* del comando **ls** viene ridiretto allo *standard input* del comando **more**, che si occupa di bloccare lo scrolling ed attendere la pressione di un tasto per continuare la visualizzazione.

Ridirezione nella BASH

La BASH identifica con un numero i flussi:

0 = Standard input - 1 = Standard output - 2 = Standard error

&n è la sintassi con la quale si ridirige un flusso ad un file già aperto.

Ad esempio 2>&1 ridirige 2 (standard error) a 1 (standard output).
Se 1 è stato ridiretto ad un file anche 2 verrà rediretto allo stesso file.

NOTA: In altre shell i caratteri usati per effettuare il Redirectory possono variare

Carattere	Azione
>	Ridirige lo ***standard output***
2>	Ridirige lo ***standard error***
2>&1	Ridirige lo ***standard error*** sullo ***standard output***
<	Ridirige lo ***standard input***
|	Esegue il ***pipe*** dello ***standard output*** su un altro programma
>>	Ridirige lo ***standard output*** accodando i dati
2>&1 |	Esegue il pipe dello ***standard error*** e ***standard output***

Gestione dei JOB

I processi eseguiti in una shell costituiscono i JOB dell'utente.

Più *job* possono essere eseguiti all'interno di una shell, ma solo un *job* può essere attivo sul terminale, quello che legge lo standard input e scrive lo standard output.

Questo processo prende il nome di *job* in ***foreground*** (primo piano), mentre l'esecuzione degli altri *job* viene definita in ***background*** (sfondo).

La shell assegna a ciascun processo un numero di *job* univoco che può essere utilizzato come argomento per i comandi che operano sui job.

Per passare un numero di job ad un programma bisogna farlo precedere dal carattere **%**

Visualizzazione dei job in esecuzione

Si utilizza il comando ***jobs***:

```
$ jobs [INVIO]
[1] – Stopped        vi
[2] + Stopped        apropos shell
```


Operazioni sui JOB

Arrestare un job

Digitando <CTRL>+c si interrompe il job in foreground prima del suo completamento uscendo dal programma.
Utilizzando il comando kill si può interrompere un job in background, specificando il numero di job come argomento.
Per interrompere il job numero 2, digitare:

```
$ kill %2 [INVIO]
```

Sospendere un *job*

Digitando <CTRL>+z, il job in foreground viene sospeso ed appare il seguente messaggio sul terminale:

```
[1] + Stopped       apropos shell
```

Se ci sono job sospesi al momento della disconnessione, la shell impedisce il logout e visualizza quanto segue:

```
$ logout [INVIO]
There are stopped jobs.
```

Operazioni sui JOB

Eseguire un job in background

I nuovi job vengono eseguiti in foreground a meno che non si specifichi diversamente.
Per eseguire un job in background si deve terminare la riga di comando con il carattere **“&”**

```
$ apropos shell > shell-commands & [INVIO]
[1] 6575
```

La shell visualizza il numero di job (in questo caso 1) ed il PID (Process IDentifier) del job (in questo caso 6575)

Al termine del job in background, la shell visualizza il numero di job, il comando e il testo “**Done**”, per indicare che il job è stato correttamente ultimato.

```
[1]+ Done      apropos shell > shell-commands
```


Operazioni sui JOB

Spostare un job in background

Per spostare un job dall'esecuzione in foreground a quella in background bisogna:

- Sospenderlo (<CTRL>+z)
- Digitare il comando **bg** (per “background”)

Per eseguire un job in background si deve terminare la riga di comando con il carattere **“&”**

```
$ apropos shell > shell-commands [INVIO]
<CTRL>+z
[1]+ Stopped                 apropos shell > shell-commands
$ bg [INVIO]
[1]+ apropos shell &
```

Se ci sono diversi job in stato sospeso (Stopped) si deve specificare il numero di job che si vuole portare in background

```
$ bg %4 [INVIO]
```

Operazioni sui JOB

Spostare un job in foreground

Per spostare un job dall'esecuzione in background a quella in foreground si usa il comando **fg**.

Per spostare in foreground il job in background più recente:

```
$ fg [INVIO]
```

Se ci fossero più job in background si deve specificare il numero di job:

```
$ fg %3 [INVIO]
```


Personalizzare la Shell

L'ambiente di lavoro può essere personalizzato in modo da rendere il lavoro più confortevole o migliorarne l'efficienza.
Una delle personalizzazione è la modifica del prompt vista in precedenza.

Alias di un comando

L'alias è un nome che rappresenta un altro comando.

```
$ alias ll=”ls -l --color=tty”
```

Modificare il percorso di ricerca (PATH)

E' possibile modificare il percorso di ricerca tramite la variabile d'ambiente PATH.

Il comando: `$ PATH=/usr/bin:/bin:/sbin`
assegna le directory /usr/bin, /bin e /sbin al percorso di ricerca

Mentre il comando:

```
$ PATH=$PATH:/mia_directory
```

appende */mia_directory* al percorso di ricerca. E' equivalente a scrivere:

```
$ PATH=/usr/bin:/bin:/sbin:/mia_directory
```


Configurare la Shell

Le personalizzazioni operate in precedenza vengono perse se si esegue il logout. Per rendere permanenti le impostazioni si possono inserire i comandi all'interno di file di configurazione che vengono richiamati all'apertura della Shell.

Si usano i seguenti file di configurazione:

/etc/profile Permette una configurazione globale
Modificabile solo dall'utente root (o da un utente amministratore)

.bash_profile, .bash_login, .profile, .bashrc e .bash_logout
Modificabili dai singoli utenti e contenuti nella home directory:

Se la shell è richiamata dal processo di login (login shell) vengono richiamati nell'ordine i file:
/etc/profile, .bash_profile, .bash_login, .profile

Se non è una login shell viene richiamato il file .bashrc

Alla chiusura della shell viene richiamato il file .bash_logout

Configurare la Shell

Un esempio di file di configurazione della shell è riportato di seguito:

```
# Make color directory listing
      alias ll=”ls –color=auto”

#Set a custom PATH
      PATH=”/usr/local/bin:/usr/bin:/bin”

#Set a custom shell prompt
      PS1=”[\W] $ “
```

Queste righe possono essere inserite in uno dei file visti in precedenza, è cura dell'utente scegliere quello corretto in base al risultato che vuole ottenere.

Editor di testo

In Linux esistono una moltitudine di editor di testo alcuni utilizzabili in terminale altri in ambiente grafico.

Noi approfondiremo l'uso di **“VI”** (si legge “vi-ai” o “vai”), uno degli editor più utilizzati nel mondo Unix.

In particolare nelle distribuzioni Linux più recenti si trova VIM un clone di VI, migliorato in alcune funzionalità.

Un altro editor molto popolare in ambiente Unix è **Emacs**.

Da segnalare anche **LaTeX** che è uno dei migliori linguaggi per la stampa e la realizzazione di testi di carattere scientifico.
A differenza di altri word-processor del tipo "WYSIWYG”, LaTeX è un vero e proprio linguaggio di programmazione che consente di gestire il testo e le immagini nel modo desiderato, oltre ad offrire la possibilità di implementare nuove funzioni per specifiche esigenze.

VI

In VI esistono tre modi di funzionamento:

COMMAND MODE	Il cursore è posizionato sul testo. La tastiera è utilizzabile solo per impartire comandi. I caratteri digitati non vengono visualizzati.
INPUT MODE	Tutti i caratteri digitati vengono visualizzati ed inseriti nel testo.
DIRECTIVE MODE	Ci si trova posizionati con il cursore nella linea direttive, l'ultima linea dello schermo, e si possono impartire a "vi" tutti i comandi per il controllo del file.

I passaggi di stato avvengono con i seguenti caratteri:

DIRECTIVE MODE	--->	COMMAND MODE	<RETURN> o due volte <ESC>
COMMAND MODE	--->	INPUT MODE	i, a, A, o, O, c, R...
INPUT MODE	--->	COMMAND MODE	<ESC>
COMMAND MODE	--->	DIRECTIVE MODE	**:**

VI - Command Mode

Per editare un file si usa il comando:

$ VI nome_file [nome_file1 [file2..... file_n]]

All'apertura ci si trova in CM i comandi più frequenti di questa modalità sono:

SPOSTAMENTI DI BASE

h,l,k,j	muovono il cursore rispettivamente a sinistra, destra, sopra e sotto
G	posizionamento sull'ultima riga di testo
#G	posizionamento sulla riga numero “#” (es: 3G, 123G)

OPERAZIONI SUL VIDEO

^D, ^U	si posiziona mezza pagina AVANTI o INDIETRO
^F, ^B	si posiziona sulla pagina SUCCESSIVA o PRECEDENTE
H, M, L	posiziona il cursore ad INIZIO, META' o FINE PAGINA

LINEA

$	posiziona il cursore alla fine della riga corrente.
^	posiziona il cursore sul primo carattere NON-BLANK
0	posiziona il cursore sulla prima colonna
#|	posiziona il cursore sulla colonna numero “#”

VI - Command Mode

EDIT

.	ripete l'ultimo comando impartito
o	OPEN: inserisce una riga sotto quella corrente
O	OPEN: inserisce una riga SOPRA quella corrente
i	INSERT: inserisce caratteri a sinistra del cursore
I	INSERT: inserisce caratteri all'inizio della riga
a	APPEND: aggiunge caratteri a destra del cursore
A	APPEND: aggiunge caratteri alla fine della riga
J	JOIN: concatena la riga successiva alla precedente
#s	SUBSTITUTE: sostituisce "#" caratteri, di default 1 carattere
#S	SUBSTITUTE: sostituisce "#" linee, di default 1 linea
#r	REPLACE: sostituisce "#" caratteri, di default 1 carattere
R	REPLACE: entra in modalità di sovrascrittura
cw	CHANGE WORD: sostituisce la parola corrente
cc	CHANGE: cambia l'intera linea
C	CHANGE: cambia fino alla fine della linea

Nota: in tutte le modalità di inserzione di testo elencate sopra (i,a,o,c,r), è possibile anteporre un numero a queste; ciò comporterà l'esecuzione della funzione desiderata tante volte quante specificate con quel numero.

VI - Command ed Input Mode

EDIT

u	annulla l'ultima modifica
p	PASTE: incolla il contenuto del buffer prima del cursore
P	PASTE: incolla il contenuto del buffer DOPO il cursore
“ [1-9]P	PASTE: incolla uno dei buffer numerati dove vengono memorizzate le cancellazioni con il metodo First-In-Last-Out
#dd	cancella “#” righe, di default 1
D	cancella dalla posizione corrente fino al termine della riga
x	cancella il carattere su cui è posizionato il cursoe (#x)
X	cancella il carattere che PRECEDE il cursore (#X)
/stringa	trova la parola “stringa” cercando in avanti
?stringa	trova la parola “stringa” cercando INDIETRO
/, ?	ripetono la ricerca in avanti o indietro

INPUT MODE

In questo modo si scrive e si può effettuare un editing semplice del file di testo.

- I tasti cursore permettono di muoversi nel testo
- Il tasto backspace cancella il carattere posto a sinistra del cursore
- Il tasto canc cancella i caratteri a destra del cursore

VI – Directive Mode

In directive mode i comandi impartiti sono prevalentemente diretti all'uso dei file

:#1, #2 co #3	copia dal numero di linea #1 a #2 dopo la linea #3
:#1, #2 d	cancella le linee da #1 a #2
:#dd	cancella "#" righe, di default 1
:#1, #2 m #3	MOVE: sposta le righe da #1 a #2 dopo la riga #3
:#1, #2 p	stampa le righe da #1 a #2
:r [nome file]	apre il file "nome file" inserendolo nella posizione in cui si trova il cursore
:n	passa al file successivo qualora siano stati aperti più file
:rew	ricomincia ad editare partendo dal primo file della lista
:q	esce da VI
:q!	forza l'esecuzione del comando che lo precede in questo caso esce senza registrare le modifiche
:w	registra le modifiche su disco
:w nome_file	registra le modifiche nel file "nome file" (salva con nome)

Si possono impartire più comandi in una sola volta.
es:

:wq!	Forza la registrazione del file e poi esce.

VI – Opzioni

Le opzioni influenzano l'ambiente dell'editor VI con lo scopo di adattarlo alle esigenze dell'utente.

L'editor 'vi' ha 3 tipi di opzioni:
- a valori numerici
- a valori stringa
- a valori booleani (vero, falso).

Si attribuisce un valore alle opzioni numeriche o a valori stringa con il comando:
:set opzione=valore

Si attribuisce il valore "vero" alle opzioni booleane (cioè le si attiva) con:
:set opzione

Si attribuisce un valore "falso" alle opzioni booleane (cioè le si disattiva) con:
:set noopzione

Per avere una lista di tutte le opzioni: :set all
Per avere una lista di tutte le opzioni modificate: :set
Per avere il valore di una opzione di tipo numerico o stringa: :set opzione

Una opzione molto utile è quella che visualizza i numeri di riga:
:set nu (:set nonu li nasconde)

Processi

Ogni comando che si lancia ed ogni servizio attivo sul sistema danno origine a uno o più processi.

Un processo consta di:
codice eseguibile, posizione di esecuzione, i dati che gestisce, i file aperti, l'ambiente di esecuzione e le credenziali.

Quando lo stesso programma è eseguito più volte nel sistema, anche da parte di utenti diversi, alcune parti dello stesso possono essere condivise (shared) in memoria: il codice in esecuzione e le eventuali librerie di sistema caricate ad esempio.

Il Process IDentifier (PID) è un numero univocamente associato ad un processo nel momento in cui viene generato.
Ogni processo deve essere generato da un altro processo di cui si definisce il PPID (Parent PID).
Si parla quindi di processo padre (parent) e processo figlio (child).

Il processo INIT è l'unico che non ha padre essendo il primo processo mandato in esecuzione, per questo si definisce anche il Padre di tutti i Padri ed il suo PID = 1.

Stato di un processo

Un processo si può trovare in uno dei seguenti stati:

Ready	E' pronto per essere eseguito.
Running	E' in esecuzione su una CPU del sistema.
Sleeping	Attende un evento.
Swapped	Parte del processo e' stato trasferito su disco, per liberare memoria per altri processi.
Terminated	Il processo e' terminato. Invio del segnale SIGCHLD al parent.
Zombie	Il processo ha terminato la sua esecuzione, ma il parent non ha raccolto il segnale SIGCHLD. Il processo mantiene ancora allocate delle risorse.

Ad ogni processo è associata una priorità, un valore che varia fra -20 e 19 (valori più alti indicano una priorità più bassa), che determina quanta CPU time rispetto agli altri il sistema gli deve dedicare.

Un processo può generare una copia di se stesso (fork), cha ha PID diverso e PPID uguale al proprio PID (uso frequente per i servizi di rete).

Monitoraggio dei processi

ps -ef

mostra una lista dei processi attivi

pstree [-c]

mostra la lista dei processi con una struttura ad albero, così da individuare facilment i padri e i figli.
L'opzione -c elenca tutti i processi anche se hanno lo stesso nome, diveramente vengono raggruppati.

top

E' un programma che visualizza in tempo reale lo stato dei processi.
Durante la visualizzazione è possibile ordinare i processi secondo una delle seguenti proprietà:

- N - Ordina secondo PID
- A - Ordina secondo età (per prima i più recenti)
- P - Ordina secondo utilizzo CPU
- M - Ordina secondo memoria utilizzata

Operazioni sui processi

Segnali

Per gestire un processo è possibile inviargli dei segnali ognuno dei quali svolge una diversa funzione.

Quelli di uso più frequelnte sono (tra parentesi il numero associato):

SIGKILL (9)	termina il processo incondizionatamente
SIGTERM (15)	termina il processo.Permette il rilascio controllato delle risorse.
SIGINT (2)	equivale a <CTRL>+c (interrompe il processo)
SIGTSTP	equivale a <CTRL>+z (sospende il processo)

Per SIGTSTP non è stato indicato il numero del segnale poiché può variare secondo il sistema, si usa pertanto il nome simbolico.

Comandi per i processi

kill -[signal] pid	invia un segnale al processo avente numero pid.
Kill -9 pid	“uccide” il processo
killall [opzioni] nome	invia un segnale a tutti i processi aventi nome.
bg	porta il processo in background
fg	porta il processo in foreground
nice <priorità> <comando>	esegue il **comando** con la **priorità** specificata

Gestione e analisi dei log

La gestione e l'analisi dei log è un'attività sistemistica che richiede una certa attenzione.

Su macchine ad alto traffico, o sotto attacco DOS o con particolari problemi ricorrenti, le dimensioni dei log possono crescere a dismisura in pochissimo tempo, fino a SATURARE il file system.

I log sono contenuti nella directory **/var/log**

Per questo motivo è sempre consigliabile montare la directory /var usando una partizione indipendente, che, anche se piena, non blocca il funzionamento del sistema.

E' importante gestirli programmando una rotazione ad intervalli regolari (logrotate) e compattando i log vecchi.

Può essere utile anche monitorare i log (logwatch) con opportune applicazioni che possano avvertire il sysadm quando si verificano determinate condizioni.

Syslogd

In tutti i sistemi Unix è il demone ***syslogd*** che si occupa di gestire i diversi log di sistema ed è configurabile tramite il file **/etc/syslog.conf**

Il file /etc/syslogd.conf presenta una sintassi semplice, per ogni riga si indica un'attività di logging nella forma:

facility.loglevel destination

La facility indica la parte del sistema che genera l'errore.
Può essere auth, cron, daemon, kern, lpr, mail, news, user, syslog, ecc..

Il loglevel indica il livello di criticità del messaggio: debug, info, notice, warning, error, crit, alert, emerg.

La destination è il file nel quale verranno memorizzati gli errori.

E' possibile usare wildmask (*), negazioni (!), uguaglianze (=) e (;) per separare più valori di faciliy.loglevel.

es:

*.info;mail.none	/var/log/messages	qualunque messaggio eccetto mail
mail.*	/var/log/mailog	qualunque messaggio riguardante le mail
cron.*	/var/log/cron	log del crontab

Gestione degli utenti

La gestione degli utenti comprende le operazioni di

- Aggiunta
- Modifica
- Assegnazione / cambio password

di tutti gli utenti che possono accedere in shell alla macchina.

Per aggiungere manualmente un utente al sistema si deve (come root):

- Editare /etc/passwd aggiungendo una riga per il nuovo utente;
- Editare /etc/group aggiungendo un nuovo gruppo per il nuovo utente (non indispensabile);
- Se esiste il file /etc/shadow editarlo aggiungendo una nuova riga per l'utente;
- Creare la home del nuovo utente: mkdir /home/nomeutente
- Ricreare l'ambiente base nella nuova home: cp /etc/skel /home/nomeutente
- Modificare il proprietario della home:
  chown -R nomeutente:nomegruppo /home/nomeutente
- Modificare i permessi della home: chmod -700 /home/nomeutente
- Modificare la password dell'utente: passwd nomeutente

Gestione utenti: comandi e file

Le operazioni necessarie per la creazione di un utente possono essere semplificate utilizzando i seguenti comandi:

useradd [opzioni] nomeutente	Aggiunge un utente, è possibile tramite le varie opzioni disponibili modificare tutte le impostazioni di default legate all'utente.
userdel [-r] nomeutente	Elimina un'utente, da sottolineare che la sua home con il suo contenuto non viene cancellata.
groupadd [-g GID] nomegruppo	Aggiunge un gruppo.
groupdel nomegruppo	Elimina un gruppo.
passwd [nomeutente]	Modifica la password di un'utente
chsh [-s shell] [nomeutente]	Cambia il tipo si shell disponibile al login di un'utente

NOTA: Il file /etc/shadow è il database delle password usato nei sistemi Unix più evoluti. In esso sono elencate per ogni utente la password (criptata) e vari parametri ad essa connessi (ultima modifica, durata massima e minima, ecc...). Ad esso fanno riferimento diversi files, fra cui /etc/passwd e tutti i comandi per la gestione degli utenti (useradd, userdel, usermod).

Backup e archiviazione

Per effettuare il backup di file e cartelle si usa il programma ***tar*** (**t**ape **ar**chive) nato per effettuare la copia su un dispositivo sequenziale (nastro magnetico /dev/st0), ma è possibile usando l'opzione **"-f"** specificare anche un file.

tar è comunemente utilizzato in abbinamento con un'utility di compressione per ridurre le dimensioni del file di archivio: gzip, bzip2 o compress (ormai obsoleto).

Per convenzione si usano le seguenti estensioni:

.tar	per i file di archivio
.tar.gz o .tgz	per i file di archivio compressi con gzip
.tar.bz2 o tbz2	per i file di archivio compressi con bzip2
.tar.Z	per i file di archivio compressi con compress

Utility di compressione disponibili sotto Unix

gzip [num] file	Comprime nel formato gz. [num] indica il livello di compressione.
gunzip file	Decomprime un file in formato gz
bzip2 file	Comprime in formato bz2. E' più efficiente di gzip
bunzip2	Decomprime un file in formato gz
(un)compress	comprime e decomprime nel formato .Z
zip e unzip	sono disponibili per compatibilità con il mondo windows

Uso di TAR

Sintassi:

tar <operazioni> [opzioni]

Operazioni comuni

[-]A --catenate –concatenate	[-]c --create
[-]d --diff –compare	[-]r --append
[-]t –list	[-]u --update
[-]x --extract –get	--delete

Opzioni comuni

-f, --file F	-Z, (compress)
-j, --bzip	-z, --gzip
-p, --preserve-permissions	-v, --verbose

Esempi

tar -cvzf copia.tgz *	crea l'archivio compresso con gzip, **copia.tgz**, di tutti ifile presenti nella directory corrente (sottodirectory incluse) visualizzando i dettagli durante l'operazione.
tar -xjf copia.tbz2	decomprime l'acrhivio compresso con bzip2, **copia.tbz2**, nella directory corrente, senza visualizzare i dettagli.

rsync

rsync permette la copia differenziale di file via rete, ottimizzando i tempi di backup e ripristino dei dati.
Es: rsync -av 10.0.0.10:dircondivisa/* /backup/10.0.0.10/home/

-av	Modo archivio. Mantiene permessi ed ownership.
10.0.0.10	è l'indirizzo IP della sorgente
dircondivisa	è la cartella da copiare
/backup/10.0.0.10/home	path locale della macchina su cui vengono copiati i file.

La prima volta che si esegue questo comando tutti i file presenti nella *dircondivisa* di 10.0.0.10 vengono copiati, le volte successive vengono copiati solo quelli modificati dall'ultimo backup.

Sull'host 10.0.0.10 deve essere in esecuzione rsync in modalità server (rsync --daemon) ed il file /etc/rsyncd.conf deve contenere una struttura simile:

[dircondivisa]	Il nome della condivisione che rsycnd esporta.
path = /home/	Il path locale corrispondente a questa condivisione
read only = yes	Per un backup non è necessario avere permessi in scrittura
uid = root	L'utente con cui rsync accede ai file: root può leggere tutti i file
gid = root	Il gruppo con cui rsync accede al file system.
hosts allow = 10.0.0.2	Gli IP da cui è possibile collegarsi via rsync (ACL).
list = false	Impedisce di elencare le condivisioni disponibili sul server rsync

BackupPC

Sono disponibili anche degli strumenti per eseguire un backup avanzato, uno di questi è BackupPc

BackupPC un sistema di backup che presenta una tipologia client/server, di livello enterprise ad alte performance per eseguire il backup di computer Linux e Windows sui dischi di un server o su un NAS.
E' altamente configurabile tramite un interfaccia web scritta in Perl.
In aggiunta si possono trasferire su nastro i dati già copiati sul server

E' stato testato sulle seguenti piattaforme:
Linux, Freenix e Solaris per la parte server
Linux, Win95, Win98, Win2000 e WinXP per la parte client.

La configurazione e l'uso di questo prodotto esula dallo scopo del corso.

Sul sito web: http://backuppc.sourceforge.net/ è possibile trovare tutte la documentazione necessaria per la sua installazione e l'utilizzo.

Corso Base su Linux

Fine Lezione 3