Sistemi Operativi

Linux introduzione

## Introduzione

Linux e' un sistema operativo nato grazie ad un progetto di Linus Torvalds, uno studente dell'universita' di scienze dell'informazione di Helsinki. L'idea era quella di creare una versione di Unix che funzionasse su PC. Unix venne sviluppato nei laboratori AT&T Bell da Ken Thompson verso la fine degli anni 60, mentre Linux e' nato all'inizio degli anni 90 (circa 30 anni dopo). Linux non solo e' molto simile ad Unix, ma sotto molti aspetti e' praticamente identico.

Linux e' un sistema operativo **multiutente** e **multitasking.**

Qualsiasi sistema Linux puo' essere suddiviso in due parti principali: il **kernel** e la **shell**. Il kernel (nocciolo) e' il programma che gestisce i dispositivi hardware (come dischi nastri e stampanti) ed esegue i programmi, mentre la shell (conchiglia) e' il programma che riceve i comandi dall'utente e li invia al kernel. La shell e' come l'ostrica la cui conchiglia avvolge la perla (il kernel).

## Accesso

L’accesso al sistema avviene quindi tramite login e password. L'amministratore di sistema viene anche chiamato superutente, superuser sysadmin o utente **root**.

## Uscita

Esistono 4 modi per effettuare il logout, cioe' per uscire dal sistema:

* Digitare il comando **exit**
* Premere i tasti CTRL e D contemporaneamente
* Premere i tasti CTRL ALT CANC contemporaneamente
* Digitare il comando **shutdown**

# La shell (storia)

I sistemi operativi che esistevano prima di Linux e di Unix (Linux e' un clone di Unix) possedevano gia' degli interpreti di comandi al loro interno, ma tali interpreti avevano un difetto: non erano modificabili. Quando Ken Thompson e Dennis Ritchie cominciarono a scrivere Unix, decisero di creare una interfaccia utente modificabile a seconda delle esigenze che si potevano presentare di volta in volta. Cio' che venne fuori fu un tipo di shell programmabile, contenente istruzioni simili a quelle del linguaggio C. Questo non e' un caso visto che la prima versione di Unix venne scritta in linguaggio assembler, ma fu successivamente riscritta in linguaggio C ([Dennis Ritchie e' insieme a Brian Kerninghan l'inventore del linguaggio C](http://cm.bell-labs.com/cm/cs/cbook/)). L'interprete dei comandi creato sotto Unix venne chiamato shell.

Una delle prime shell venne creata agli inizi degli anni 70 presso i Bell Laboratories AT&T ad opera di Steven R. Bourne e venne chiamata appunto Bourne shell (sh). Verso la fine degli anni 70 presso l'universita' di Berkley venne creata la C shell (csh) allo scopo di estendere la shell Bourne e renderla piu' simile al linguaggio di programmazione C. Successivamente vennero sviluppate altre shell, come la Korn shell (ksh) e la TC shell (tcsh).

A causa dei problemi di copyright, l'organizzazione GNU decise di sviluppare una shell completamente libera da restrizioni e nacque cosi' la shell **BASH**, cioe' Bourne Again SHell. Esistono molteplici versioni di shell, la shell ash, la shell csh, la shell ksh, la shell pdksh, la shell tcsh, la shell zsh ed altre ancora, ma la shell BASH e' senza dubbio la piu' famosa ed usata in tutti i sistemi Linux.

La shell BASH e' conforme allo standard **POSIX**. Posix sta per Portable Operating System Interface for uniX, cioe' interfaccia standard per il sistema operativo Unix e si tratta di uno standard creato verso la fine degli anni 80 allo scopo di uniformare le varie versioni di Unix. Successivamente, ad opera del gruppo IEEE (Institute of Electrical and Electronics Engineers, cioe' istituto degli ingegneri elettrotecnici ed elettronici) lo standard POSIX venne adottato anche da altri sistemi operativi non Unix come VMS, MVS, NT etc.

## La shell BASH

Per iniziare ad esplorare la shell BASH e' sufficiente disporre di un emulatore di terminale. Se stiamo lavorando in ambiente grafico (X Window) occorre aprire una finestra di emulazione terminale come ad esempio Xterm, altrimenti non occorre fare nulla, in quanto ci troviamo gia' in modalita' terminale. Nel primo caso, supponendo di usare un desktop grafico come KDE, bastera' cliccare sul bottone di avvio di KDE, selezionare 'terminali' e scegliere il tipo di terminale che si preferisce. A questo punto ci si trova all'interno di una shell, che potrebbe essere BASH o CSH o TCSH o altro. Normalmente la shell di default, cioe' quella preimpostata inizialmente e' la shell BASH. Ad ogni modo e' sempre possibile invocare la shell BASH digitando il comando 'sh' oppure il comando 'bash'. Questo comando non fa altro che invocare la shell BASH. Piu' esattamente invoca una subshell, ossia una shell BASH all'interno della shell corrente. Per uscire dalla subshell e' possibile digitare il comando 'exit' o premere i tasti CTRL-D (Ctrl e D contemporaneamente). Premendo CTRL-D da una subshell si ritorna alla shell corrente e premendo nuovamente CTRL-D si esce dalla shell iniziale. Per scoprire quali shell siano installate nel nostro sistema, e' possibile visualizzare il file /etc/shells utilizzando il comando:

cat /etc/shells

## Help

* nome\_comando -h
* nome\_comando --help
* help
* help nome\_comando
* dhelp *cerca all'interno della documentazione Debian*
* dhelp *cerca nella documentazione RedHat*
* man nome\_comando *visualizza la pagine di manuale*
* apropos nome\_comando *cerca un determinato argomento*
* whatis nome\_comando *ricerca nel manuale della guida per una parola*
* *info consultazione dei documenti relativi a prodotti GNU*

# FileSystem

Alcune osservazioni di carattere generale:

* nello scrivere un percorso (path) si usa la slash ("/") (come per un indirizzo internet) e non la back-slash ("\")
* con il S.O. DOS i nomi dei file sono nella forma 8.3 ( PROVATES.TXT ); in Linux si possono usare nomi di file lunghi più di 255 caratteri e con più caratteri punto (".") all'interno del nome. Es: Questo\_e.un.nome-di.file-molto\_lungo
* caratteri maiuscoli e minuscoli rendono due file differenti anche se hanno lo stesso nome. Quindi NOMEFILE.tar.gz è nomefile.tar.gz sono differenti.
* i nomi dei file non sono collegati in alcun modo ad estensioni particolari. Ad esempio un file eseguibile non deve avere estensione .COM, .EXE o .BAT ma per essere eseguito deve avere solo le giuste "permission". In Linux i file eseguiti sono marcati con un asterisco ("\*") dai programmi che visualizzano l'elenco di una directory.
* sotto DOS i file di backup terminano con l'estensione .BAK, in Linux invece terminano con una tilde ('~').
* in Linux un nome di file che inizia con un punto (".") e considerato "hidden" (nascosto/invisibile)
* il passaggio dei parametri in DOS avviene mediante l'uso della slash ("/") mentre in linux si usa il segno meno ("-") singolo o doppio ("--"). E da notare che alcuni famosi programmi DOS (PKZIP e ARJ) usano già questa sintassi Unix.

## Struttura delle directory

Esiste una struttura gerarchica di directory 'abbastanza' standard. Abbastanza standard significa che puo' variare leggermente da distribuzione a distribuzione, ma piu' o meno le directory predefinite di qualsiasi distribuzione Linux sono le seguenti:

* /
* bin
* boot
* dev
* etc
* home
* lib
* lost+found
* mnt
* proc
* opt
* root
* sbin
* tmp
* usr
* var

I s.o. dos/windows automaticamente monta tutte le unità (floppy, hard-disk, cd-rom) usando le lettere dell'alfabeto c:, d:, ecc. In Linux il disco principale contenente il 'ROOT FileSystem' viene montato sotto la root directory, cioe '/'. Gli altri file system, come cdrom e connessioni di rete possono essere "montati" in qualsiasi parte dello stesso, anche se lo standard dice che dovrebbero stare in una sottocartella del percorso "/mnt".

## Gestione delle Directory

Questi sono alcuni dei comandi per la gestione delle directory:

* ls elenca i file contenuti in una directory, cioè DIR
* cd come in DOS serve a cambiare la directory corrente
* mkdir crea una sottodirectory, come MKDIR in DOS
* rmdir rimuove una sottodirectory, come RMDIR in DOS
* cp copia file o sottodirectory, come COPY in DOS
* mv muove o rinomina una sottodirectory, come MOVE in DOS
* rm rimuove file o sottodirectory, come DEL in DOS
* pwd restituisce il nome della directory corrente come CD in DOS
* du dà indicazione sullo spazio usato dai file
* df dà indicazione sullo spazio libero sui dischi

Ecco alcune delle wildcard usate nei comandi Linux che manipolano files o directory:

* “;” Permette di inserire più comandi in successione sulla stessa riga. Esempiople: ls /bin ; ls /sbin ; ls /tmp
* "&" Esegue un comando in background. Esempio: httpd &
* "|" pipe
* "#" commento
* "\" indica alla shell che il comando che si sta scrivendo continua alla riga successiva.
* "$" indica una variabile di sistema. Esempio: $PATH=/bin:/sbin
* "?" wildcard (un carattere qualsiasi ma solo in quella posizione) Esempio: /bin/c?t
* "\*" wildcard (qualsiasi carattere tranne i file nascosti) Esempi: "ls gcc\*" o "ls \*.rpm" o "ls lib\*.so.\*"
* ".\*" wildcard (come '\*' ma include anche i file nascosti)
* "[]" wildcard (solo i caratteri specificati ed in quella posizione) Esempio: "ls [Cc]at", cerca per i files "Cat" e "cat"

### Completamento automatico del Path

Se la shell che state utilizzando è bash vi potrà tornare molto utile la feature del "completamento automatico del path" (Path Completion). In pratica mentre nella shell si sta digitando il nome di un percorso o di un file, la pressione del tasto [TAB] provoca il completamento automatico

## Permessi Utente

Lanciando il comando "ls -al" noterete che visualizza le directory ed i file in un formato esteso rispetto allo standard, questo include informazioni sulle permission e l'ownership.  
  
Ecco come appare:

drwxr-xr-x 2 dosuser dos 256 Apr 28 15:40 ./

drwxr-xr-x 15 dosuser dos 928 Apr 28 14:50 ../

-rw-r--r-- 1 dosuser dos 20 Apr 28 16:05 .bash\_history

-rw-r--r-- 1 dosuser dos 6993 Oct 5 16:01 \_bhist

-rw-r--r-- 1 dosuser dos 1150 Apr 28 15:55 \_blogin

-rw-r--r-- 1 dosuser dos 40 Dec 30 1996 \_blogout

-rw-r--r-- 1 dosuser dos 1131 Dec 31 1996 \_inputrc

-rwxr-xr-x 1 dosuser dos 220 Nov 8 1996 man2txt.sh\*

Il primo campo sono le permission. Sono nella forma: 'drwxrwxrwx', quando uno di questi flag non è presente è indicato con '-'. Il flag 'r' indica che il file è leggibile. 'w' che è scrivibile ed 'x' che è eseguibile.

Il primo flag 'd' indica se il file è una directory. Diversamente '-' indica un file normale mentre 'l' signiifca che il file è un link ad un'altro file.   
La prima serie di 'rwx' sono riferite alle permissione che ha l'owner' (il padrone) del file che è il nome indicato nella terza colonna (dosuser).  
La seconda serie di 'rwx' sono le permission che ha il gruppo sul file. Il gruppo è indicato alla 4a colonna (dos).

La terza serie di 'rwx' sono le permission che hanno tutti gli altri utenti su quel file.  
  
Un utente che crea un file con permission '-rw-------', sarà l'unico ad avere accesso a quel file.

E' da notare che le permission per la directory './' corrispondo alle permission che ha la directory corrente.

Ricordatevi anche che i file nascosti sono quelli che iniziano con un punto, e che per essere visualizzati bisogna usare l'opzione -a con il comando "ls".

|  |  |  |
| --- | --- | --- |
| **chi** | **Azione** | **permessi** |
| u = user (utente) | + = aggiungi | r = read (lettura) |
| g = group (gruppo) | - = togli | w = write (scrittura) |
| o = other (altri) | = = assegna | x = execute (esecuzione) |
| a = all user (tutti) |  |  |

# Comandi di Shell

## Standard input, standard output e standard error

ls > pippo.txt

redirige l’output del comando nel file pippo.txt

ls < pluto

redirige il contenuto del file pluto nell'input standard

cat < pippo > pluto

legge dal file pippo e scrive il risultato nel file pluto.

Ma un comando puo' produrre anche dei messaggi di errore se per esempio vengono introdotti dei valori in input non validi. I messaggi di errore dei comandi vengono scritti in un terzo canale, lo **standard error**. Lo standard error generalmente coincide con lo standard output

### Le pipe

Utilizzate per inviare l'output di un comando non a video ma ad un altro comando prima di essere visualizzato.

ls | more

pwd | cat - miofile

cat pippo | tee pluto

Per visualizzarne il contenuto e contemporaneamente scriverlo all'interno di un altro file, si puo usare il comando tee.

# Esercizi proposti

**Aprire la shell**

**Provare i seguenti comandi e commentare:**

pwd

ls

man ls

ls –al

echo benvenuti a tutti

ls > lista.txt

cat lista.\*

echo benvenuti > testo.txt

mkdir prova

ls

cp l\* prova

cd prova

pwd

ls -l

chmod a+r lista.txt

Riferimenti :

<http://www.wowarea.com/italiano/linux/shell.htm>