02 - Shell Comandi

Shell

E' un programma che permette di far interagire l'utente (interfaccia testuale) con il sistema operativo tramite comandi ed e' uno strumento alternativo all'interfacia grafica. La shell e' un programma che resta sempre in attesa di un comando e viene mandato in esecuzione tramite la pressione del tasto <ENTER>.
La shell e' un interprete di comandi evoluto

- potente linguaggio di scriping
- interpreta ed esegue comandi da standard input o da file comandi

Esistono differenti tipi di shell: **Bourne shell** (standard), C shell, Korn shell, ...
L'implementazione della Bourne shell in Linux e' Bash (/bin/bash). Ogni utente puo' specificare la propria shell preferita andando a modificare un file contenente le informazioni di tutti gli utenti del sistema che si trova nella directory /etc/passw.

La shell di login e' quella che richiede inizialmente i dati di accesso all'utente, per ogni utente connesso viene genereato un processo dedicato (che esegue la shell).

Accesso al sistema: login

Per accedere al sistema bisogna possedere una coppia **username e password.**
Il sistema operativo verifica le credenziali dell'utente e manda in esecuzione la sua shell di preferenza, posizionandolo in un direttorio di partenza (entrambe le informazioni si trovano in /etc/passw).
E' possibile cambiare la propria password di utente mediante il comando passwd. Se ci si dimentica della password, bisogna chiedere all'amministratore di sistema (utente *root*).

Uscita dal sistema: logout

Per uscire da una shell qualsiasi si puo' utilizzare il comando **exit** (che invoca la system call **exit()** per quel processo).
Per uscire dalla shell di login:

- logout
- CTRL+D (che corrisponde al carattere <EOF>)
- CTRL+C

Per rientrare nel sistema bisogna effettuare un nuovo login.

Esecuzione di un comando

Ogni comando richiede al SO l'esecuzione di una particolare azione. I comandi principali del sistema operativo si trovano all'interno della directory /bin. E' anche possibile realizzare nuovi comandi creando dei file di scripting all'interno del quale esistono una serie di comandi/righe di esecuzione e per ogni comando la shell genera un nuovo processo figlio (tramite l'esecuzione di una fork) e questo nuovo processo, separato dalla shell iniziale ha il compito di eseguire il comando richiesto. Alla fine dell'esecuzione del comando tale processo termina la sua esecuzione e viene quindi eliminato dal sistema operativo. Il processo padre ha due possibilita':

- attendere la terminazione del comando (foreground)
- proseguire in parallelo (background)

Comandi e input/output

I comandi UNIX si comportano come **filtri**. Un filtro e' un programma che riceve in ingresso da input e produce il risultato su uno o piu' output

Manuale

Per ogni comando offerto da Linux esiste un manuale on-line (man). il manuale indica:

- il formato del comando (input) e il risultato atteso (output)
- descrizione delle opzioni
- possibili restrizioni
- file di sistema interessati dal comando
- comandi correlati
- eventuali bug

Per uscire dal manuale basta digitare *q* (**quit** per editor di tipo vi)

Formato dei comandi

Tipicamente tutti i comandi Linux hanno questo formato: **nome -opzione argomenti** (es. ls -l temp.txt)

Convenzione nella rappresentazione della sintassi comandi:

- se un'opzione o un argomento possono essere omessi, si indicano tra quadre [opzione]
- se due opzioni/argomenti sono mutuamente esclusivi, vengono separati da | (arg1 | arg2)
- quando un argomento puo' essere ripetuto n volte, si aggiungono dei puntini (arg...)

Cenni pratici introduttivi all'utilizzo del file system Linux

File

Un file e' una risorsa logica costituita da una sequenza di bit a cui viene dato un nome. Tocca al sistema operativo associare il file (la nostra risorsa logica) a uno spazio su disco. Il file e' un'astrazione mlto potente che consente di trattare allo stesso modo entita' fisicamente diverse come file di testo, dischi rigidi, stampanti, directory, tastiera, video, ecc.

Esistono tre tipi di file:

- **Ordinari**: archivi di dati, comandi, programmi sorgente, eseguibili, ...
- **Directory**: gestiti direttamente solo da SO, contengono riferimenti a file (che possono essere strutturat in gerarchie)
- **Speciali**: dispositivi hardware, memoria centrale, hard disk, ecc. (essi sono gestiti come astrazione file, ma non vengono contati come file ordinari).

In aggiunta ci sono anche

- FIFO (pipe) - file per la comunicazione tra processi
- soft link - riferimenti (puntatori) ad altri file o directory

File: nomi

- E' possibile nominare un file con una qualsiasi sequenza di caratteri (max 255), a eccezione di '.' e '..'
- E' sconsigliabile utilizzare per il nomi di file dei caratteri speciali, ad es. metacaratteri e segni di punteggiatura
- Ad ogni file possono essere associati uno o piu' nomi simbolici (link) ma ad ogni file e' associato uno e un solo descrittore (i-node) identificato da un intero (i-number)

Directory

le directory sono gestite con una struttura a grafo diretto aciclico (DAG) e il punto di partenza e' sempre la radice (root) rappresentata da un singolo slash (/).

Gerarchie di directory

- All'atto del login, l'utente puo' cominciare a operare all'interno di una specifica directory (home). In seguito e' possibile cambiare directory
- E' possibile visualizzare il percorso completo attraverso il comando **pwd** (print working directory)
- Essendo i file organizzati in gerarchie di directory, SO metta disposizione dei comandi per muoversi all'interno di essi

Nomi relativi/assoluti

Ogni utente puo' specificare un file attraverso

- **nome relativo**: e' riferito alla posizione dell'utente nel file system (directory corrente)

- **nome assoluto**: e' riferito alla radice della gerarchia /

Nomi particolari:

- **.** : e' la directory corrente (visualizzato da pwd)
- **..** : e' la directory "padre"
- ~ : e' la propria home utente

Il comando **cd** permette di spostarsi all'interno del file system, utilizzando sia nomi relativi che assoluti. cd senza parametri riporta alla home dell'utente.

Link

Le informazioni contenute in uno stesso file possono essere visibili come file diversi, tramite "riferimenti" (link) allo stesso file fisico.

Il sistema operativo considera e gestisce la molteplicita' possibile di riferimenti:

- se un file viene cancellato, le informazioni sono veramente eliminate solo se non ci sono altri link a esso
- il link cambia diritti? --> meglio di no

Due tipi di link:

1. link fisici (si collegano alle strutture del file system)
2. link simbolici (si collegano solo ai nomi)

comando: ln [-s]

Gestione file: comando ls

Consente di visualizzare nomi di file. Usato all'interno di una directory elenca tutti i file presenti al suo interno.

- Varie opzioni: ls -l (per avere piu' informazioni sul file e quindi non solo il nome)
- possibilita' di usare metacaratteri (wildcard)
    - per es. se esistono i file f1, f2, f3, f4
        - ci si puo' riferire a essi scrivendo **f***
        - o piu' precisamente **f[1-4]**

Opzioni del comando **ls
***ls [-opzioni...] [file...]*

Alcune opzioni:

- **l** (long format): per ogni file una linea che contiene diritti, numero di link, proprietario del file, gruppo del proprietario, occupazione di disco (blocchi), data e ora dell'ultima modifica o dell'ultimo accesso e nome
- **t** (time): la lista e' ordinata per data dell'ultima modifica
- **u**: la lista e' ordinata per data dell'ultimo accesso
- **r** (reverse order): inverte l'ordine
- **a** (all files): fornisce una lista completa (normalmente i file il cui nome comincia con il punto non vengono visualizzati)
- **F** (classify): indica anche il tipo di file (eseguibile: *, directory:/, link simbolico: @, FIFO: |, socket: =, niente per i file regolari)

Comandi vari di gestione

- Creazione/gestione di di directory
  - **mkdir** <nomedir> creazione di un nuovo direttorio
  - **rmdir** <nomedir> cancellazione di un direttorio
  - **cd** <nomedir> cambio di direttorio
  - **pwd** stampa il direttorio corrente
  - **ls** [<nomedir>] visualizz. contenuto del direttorio
- Trattamento file
  - **ln** <vecchionome> <nuovonome> link
  - **cp** <filesorgente> <filedestinazione> copia
  - **mv** <vecchionome> <nuovonome> rinomina / sposta
  - **rm** <nomefile> cancellazione
  - **cat** <nomefile> visualizzazione

Comando ln, link

Serve per creare un novo link.
Le informazioni contenute in uno stesso file possono essere visibili come file diversi, tramite riferimenti (link) allo stesso file fisico.
Il sistema considera e tratta tutto: se un file viene cancellato, le informazioni sono veramente eliminate solo se non ci sono altri link a esso.

cat

Serve per visualizzare i contenuti all'interno di un file.

more (e less)

Il comando **cat** effettua una semplice stampa del contenuto a video di un file. Per una visualizzazione a pagine, certamente piu' comoda per file di larghe dimensioni, e' necessario utilizzare un comando di

paginazione.
Il comando **more** permette di visualizzare un file una pagina per volta.
Esistono diverse alternative piu' recenti e potenti al comando more, per esempio il comando less (che permette anche di far scorrere la visualizzazione all'indietro).

echo

Il comando echo stampa a video la stringa passatagli come parametro.
Il comando echo rappresenta la primitiva fondamentale che la shell mette a disposizione per stampare informazioni a video.

sort

Il comando sort riordina le righe di un file in ordine alfabetico.
Il comando sort ha molte opzioni:

- **-o** <nomefileout> stampa su file
- **-n** interpreta le righe dei file come numeri
- **-k** <n> ordina il file secondo il contenuto della n-esima colonna

**
diff**

Stampa sullo standard output la differenza di contenuto fra i due file. Questo comando e' molto utilizzato nella gestione del codice sorgete.

wc

Il comando wc stampa il numero di righe, parole o caratteri contenuti in un file.
Esempi e opzioni:

- wc -l nome file.txt (conta le linee contenute nel file nome.txt)
- wc -w nomefile.txt (conta le parole contenute nel file nome.txt)
- wc -c nomefile.txt (conta i caratteri contenuti nel file nome.txt)

grep

Il comando grep seleziona le righe di un file che cntengono la stringa passata come paramatro e le stampa a video.
Esempi:

- grep stringa nomefile.txt (seleziona le righe del file nomefile.txt che contengono il testo "stringa" e le stampa a video)
- grep -c stringa nomefile.txt (conta le righe del file nomefile.txt che contengono il testo "stringa" e stampa il numero a video)

- grep -r stringa dir (seleziona le righe di tutti i file nella directory dir che contengono il testo "stringa" e le sta pa a video. ricorsivo sulle sotto directory)

Il comando grep ha numerosissime opzioni ed e' utilizzatissimo, specialmente nella gestione e manipolazione di file e di codice sorgente.

head

Il comando head mostra le prima righe di un file.
esempi:

- head -n 15 nomefile.txt (mostra le prime 15 righe del file)
- head -c 30 nomefile.txt (mostra i primi 30 caratteri del file)

tail

Il comando tail mostra le ultime righe di un file.
esempi:

- tail -n 15 nomefile.txt (mostra le ultime 15 righe del file)
- tail -c 30 nomefile.txt (mostra gli ultimi 30 caratteri del file)

time

Il comando time cronometra il tempo di esecuzione di un comando.

who

Il comando who mostra gli utenti attualmente collegati al sistema (ovverosia che hanno eseguito il login sulla macchina)

man

In ambiente UNIX, il comando man rappresenta l'help do sistema.
Esempio: man grep (mostra l'help del comando grep)
Oltre al comando man, le moderne distribuzioni di Linux mettono a disposizione anche il comando info.

ps

Permette di elencare tutti i processi all'interno del sistema operativo. Se il comando ha un certo nome, il processo relativo avra' lo stesso nome. Il comando ps e' molto utile quando si lancia l'esecuzione di programmi di sistema con errori di programmazione.

Alcune opzioni importanti:

- a -- mostra anche i processi degli altri utenti
- u -- fornisce il nome dell'utente che ha lanciato il processo e l'ora in cui il processo e' stato lanciato
- x -- mostra anche i processi senza terminale di constrollo (proesso demoni)

Attenzione: si ricordi di non usare il trattino (-) per specificare le opzioni del comando ps, in quanto ps adotta la sintassi del tipo "extended BSD" che non prevede l'uso di trattini.
Se si vuole un'aggiornamento periodico dello stato dei processi correnti, si usa il comando **top**.

top

Il comando top fornisce una visione dinamica in real-time del sistema corrente. Esso visualizza continuamente informazioni sull'utilizzo del sistema (memoria fisica e virtuale, CPU, ecc.) e sui processi che usano maggiore share di CPU.
Esiste una versione piu' avanzata e moderna di top, chiamata **htop**. Al contrario di top, di solito htop non e' inclusivo tra le utility di base disponibili sulle macchine Linux e va esplicitamente installato.

pgrep e pkill

Il comando pgrep restituisce il pid (process id) dei processi che corrispondono alla caratteristiche richieste (nome processo, utente, ...).
pkill rappresenta la stessa interfaccia dei comandi pgrep, ma anziche' stampare a video i pid di processi, li uccide (in realta' invia loro in segnale di SIGTERM)

Terminazione forzata di un processo

E' possibile terminare forzatamente un processo tramite il comando **kill**.
Ad esempio: **kill -9 <PID>** provoca l'invio di un segnale SIGKILL (forza la terminazione del processo che lo riceve e non puo' essere ignorato) al processo identificato da PID.

Segnali di interruzione

E' possibile interrompere un processo (purche' se ne abbiano i diritti) tramite il comando **kill -s <PID>**.
Questo comando provoca un segnale (individuato dal parametro s) al processo identificato dal PID.
Alcuni tra i segnali piu' comuni:

- CTRL-C -- invia un SIGINT, terminazione di un processo attualmente in foreground, kill -2
- CTRL-Z -- invia un SIGTSTP, sospensione di un processo, kill -20
- kill -l fornisce la lista dei segnali

Utenti e gruppi

- Sistema multiutente --> problemi di privacy e di possibili interferenze: necessita' di proteggere/nascondere informazione
- Concetto di gruppo (es. staff, utenti, studenti, ...)
- Ogni utente appartiene a un gruppo ma puo' far parte anche degli altri a seconda delle esgenze e coinfigurazioni
- Comandi relativi all'identita' dell'utente:

- whoami
- id

Protezione dei file

- Molti utenti
  - necessita' di regolare gli accessi alle informazioni
- Per un file, esistono 3 tipi di utilizzatori:
  - proprietario, **user**
  - gruppo del proprietario, **group**
  - tutti gli altri utenti, **others**
- Per ogni tipo di utilizzatore, si distinguono tre modi di accesso al file
  - lettura (r)
  - scrittura (w)
  - esecuzione (x) - per una directory significa list del contenuto
- Ogni file e' marcato con
  - User-ID e Group-ID del proprietario
  - 12 bit di protezione

SUID e SGID

SUID (Set User ID) e' un identificatore di utente effettivo. Si applica a un file di programma eseguibile solamente.
Se vale 1, fa si che l'utente che sta eseguendo quel programma venga considerato il proprietario di quel file (solo per la durata dell'esecuzione).
E' necessario per consentire operazioni di lettura/scrittura su file di sistema, che l'utente non avrebbe il diritto di leggere/modificare.

Protezione e diritti sui file

E' possibile cambiare i bit di protezione col comando:

- chmod [u g o] [+ -] [rwx] <nomefile>

I permessi possono essere concessi o negati solo dal proprietario del file.

Comandi, piping e ridirezione

Ridirezione di input e output

E' possibile ridirigere input e/o output di un comando facendo si che non si legga da stdin (e/o non si scriva su stdout) ma da file:

- Ridirezione dell'input:
  - comando < file_input (aperto in lettura)
- Ridirezione dell'output
  - comando > file_output (aperto in scrittura, nuovo o sovrascritto)
  - comando >> file_output (scrittura in append)

Piping

E' possibile utilizzare il piping per ridirigere l'output di un comando verso l'input di un altro comando.

- In DOS: realizzazione con file temporanei
- In UNIX:: pipe come costrutto parallelo (L'output del primo comando viene reso dispinibile al secondo e consumato appena possibile, non ci sono file temporanei)

Si realizza con il carattere speciale " | ".

Metacaratteri ed espansione

Metacaratteri

Shell riconosce caratteri speciali (**wild card**)

- * --> una qualunque stringa di zero o piu' caratteri in un nome di file
- ? --> un qualunque carattere in un nome file
- [zfc] --> un qualunque carattere, in un nome file, compreso tra quelli nell'insieme. Anche range di valori: **[a-d]**
- # --> commento fino a fine della linea
- \ --> escape (segnala di non interpretare il carattere successivo come speciale)

Variabili nella shell

In ogni shell e' possibile definire un insieme di variabili (trattate come stringhe) con nome e valore.

- i riferimenti a i valori delle variabili si fanno con il carattere speciale $ ($nomevariabile)
- si possono fare assegnamenti: nomevariabile=$nomevariabile

Ambiente di esecuzione

Ogni comando esegue nell'ambiente associato (insieme di variabili di ambiente definite) alla shell che esegue il comando

- ogni shell eredita l'ambiente dalla shell che l'ha creata
- nell'ambiente ci sono variabili alle quali il comando puo' fare riferimento:
  - variabili con significato standard: PATH, USER, TERM, ...

- variabili user-defined

Per vedere tutte le variabili di ambiente e i valori loro associati si puo' utilizzare il comando **set**.

Espressioni

Le variabili in shell sono stringhe. E' comunque possibile forzare l'interpretazione numerica di stringhe che contengono la codifica di valori numerici

- comando **expr
  **

Espansione

Prima dell'esecuzione, il comando viene scandito (parsing), alla ricerca di caratteri speciali.

- La shell prima prepara i comandi come filtri: ridirezione e piping di ingresso e uscita
- Nelle successive scansioni, se shell trova altri caratteri speciali, produce delle sostituzioni (passo di espansione)

Passi di sostituzione

Sequenza dei passi di sostituzione:

1. Sostituzione dei comandi
   - comandi contenuti tra ` ` (backquote) sono eseguiti e sostituiti dal risultato prodotto
2. Sostituzione delle variabili e dei parametri
   - nomi delle variabili ($nome) sono espansi nei valori corrispondenti
3. Sostituzione dei metacaratteri in nomi di file
   - metacaratteri * ? [ ] sono espansi nei nomi di file secondo un meccanismo di **pattern matching**

Inibizione dell'espansione

In alcuni case e' necessario privare i caratteri speciali del loro significato, considerandoli come caratteri normali.

- \ --> carattere successivo e' considerato come un normale carattere
- ' ' (apici) --> proteggono da qualsiasi tipo di espansione
- " " (doppi apici) --> proteggono dalle espansioni con l'eccezione di $ \ ` `

Comando cut

- Comando utile per la manipolazione di testo che consente di selezionare parti di una stringa (o di ciascuna riga di un file).

- cut permette di selezionare i caratteri della stringa che si trovano nelle posizioni specificate (opzione -c).
- cut puo' dividere una stringa in piu' campi (opzione -f) usando uno specifico delimitatore (opzione -d) e puo' selezionare i campi desiderati.

tr

- tr e' un comando che permette di fare semplici trasformazioni d caratteri all'interno di una stringa (o di ciascuna riga di un file), ad esempio sostituendo tutte le virgole con degli spazi
- tr esegue anche sostituzioni tra set di caratteri

seq

seq e' un comando che stampa sequenze di numeri. Puo' essere molto utile per la realizzazione di cicli negli script

find

find permette di cercare file all'interno del file system che soddisfano i requisiti richiesti dall'utente, ed eventualmente di manipolarli.

tee

- tee copia lo standard input sia nello standard output sia in un file passato come parametro
- e' molto utile quando si scrivono comandi con molte pipe, per monitorare il funzionamento di uno stadio della pipe

Script: realizzazione file comandi

La shell non e' unica. Nei moderni sistemi di Linux (e UNIX) sono disponibili diversi tipi di shell

- sh: Bourne Shell
- bash: Bourne Again Shell (versione avanzata di sh)
- zsh: Z Shell (versione molto avanzata di sh)
- ksh: Korn Shell
- csh: C Shell (sintassi simile al C)
- tcsh: Turbo C Shell (versione avanzata di csh)
- rush: Ruby Shell (Shell basata su Ruby)
- hotwire: propone un interessante e innovativo modello integrato di terminale e shell

La shell piu' usata e' sicuramente Bash, che e' molto simile alla Shell di Bourne (/bin/sh).

File Comandi

Shell e' un processore di comandi in grado di interpretare file sorgenti in formato testo e contenenti comandi --> file comandi (script).

Linguaggio di comandi (vero e proprio linguaggio di programmazione). Un file comandi comprende:

- statement per il controllo di flusso
- variabili
- passaggio dei parametri

Scelta della Shell

La prima riga di un file comandi deve specificare la hell che si vuole utilizzare:
es. #! /bin/bash

Passaggio di parametri

./nomefilecomandi arg1 arg2 ... argN

Gli argomenti sono variabili posizionali nella linea di invocazione contenute nell'ambiente della shell

- $0 rappresenta il comando stesso
- $1 rappresenta il primo argomento

e' possibile far scorrere tutti gli argomenti verso sinistra con **shift**.

Altre informazioni utili

Oltre agli argomenti di invocazione del comando abbiamo:

- **$*** --> insieme di tutte le variabili posizionali, che corrispondono arg del comando ($1, $2, ecc.)
- **$#** --> numero degli argomenti passati ($0 escluso)
- **$?** --> valore (int) restituito dall'ultimo comando eseguito
- **$$** --> id numerico del processo in esecuzione (pid)

Forme di input/output

- read var1 var2 var3 (per l'input)
  - la stringa in ingresso viene attribuita alla/e variabile/i secondo la corrispondenza posizionale
- echo var1 vale $var1 e var2 vale $var2 (per l'output)

Strutture di controllo

Ogni comando in uscita restituisce un valore di stato che indica il suo completamento o fallimento. Tale valore e' posto nella variabile $?, se $? vale 0 il comando e' stato completato correttamente, se invece ha un valore positivo abbiamo un errore.

test

Comando per la valutazione di un espressione

- test -<opzioni> <nomefile>

Restituisce uno stato uguale o diverso da zero

- valore zero --> true
- valore non-zero --> false

Alcuni tipi di test:

- -f <nomefile> --> esistenza di file
- -d <nomefile> --> esistenza di direttori
- -r <nomefile> --> diritto di lettura su file (-w e -r)
- test <stringa1> = <stringa2> --> uguaglianza stringhe
- test <stringa1 != <stringa2> --> diversita' stringhe

Gli spazi intorno a = o a != SONO NECESSARI