Introduzione alla Bash (Bourne Again Shell)

(Rosenblatt)

Introduzione

- La shell e' un interprete di comandi.
- I comandi digitati dall'utente vengono letti dalla shell, interpretati e inviati al kernel per essere eseguiti.
- I comandi vengono digitati dall'utente sulla riga comandi della shell.
- La riga comandi e' una riga digitabile che comincia dal prompt.
- Il prompt e' un carattere o un insieme di caratteri che possono essere personalizzati dall'utente.
  - Il prompt dell'utente root (il superuser) inizia con il carattere '#' (cancelletto), mentre il prompt di un utente qualsiasi inizia con il carattere '$'.
- La shell viene eseguita quando facciamo login e termina al logout

Shell

- La shell è un programma che si trova nella directory /bin.
- Si distinguono più di una shell
  - **/bin/sh** shell Bourne
  - **/bin/bash** Bourne Again SHell
  - **/bin/csh** C shell
  - **/bin/ksh** Korn shell
  - **/bin/tcsh** shell migliorata
  - **/bin/zsh**/ Z shell
- La shell bash è la shell di default per le distribuzioni GNU/Linux.

BASH

- 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**).

BASH (II)

- progetto GNU:
  - sviluppare una versione gratuita di UNIX
  - GNU = Gnu's Not Unix (acronimo ricorsivo)
  - open software, FSF
  - software sotto “copyleft”:
    - ✓ distribuito gratis con il codice sorgente
    - ✓ e deve essere mantenuto tale: non si puo' vendere
- 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' **B**ourne **A**gain **SH**ell.

Il prompt di comandi della shell

- Il prompt di comandi della shell ha generalmente la forma:

  nome@computer ~ $

  - **nome** rappresenta il login dell'utente connesso
  - **computer** rappresenta il nome del computer
  - **~** indica la directory personale /home/utente
  - **$** significa che si è connesso come utente
- Se al posto di **$** si visualizza il simbolo **#**, allora si è connesso come superutente (root).

Compiti della shell

- Supponiamo di dare il comando bash:

  sort –n num_telefoni > num_tel.ordinati

- La shell dovrà:

1. separare i token del comando:
   sort, -n, num_telefoni, >, num_tel.ordinati
2. Determinare il significato dei token
   1. **sort** comando da eseguire
   2. **-n** e **num_telefoni** argomenti
   3. **>** operatore di ridirezione
   4. **num_tel.ordinati** nome del file per la ridirezione
3. Preparare il sistema in modo tale che l'output vada nel file **num_tel.ordinati**
4. Cercare ed eseguire il comando **sort**

L'ambiente shell

- Dopo la connessione, l'utente è connesso al suo ambiente.
- Ciò significa che la shell mette alla sua disposizione delle variabili d'ambiente
  - Per visualizzare il contenuto di una variabile d'ambiente, può essere usato il comando **echo $NOME_VARIABILE**.
- Alcune variabili ambiente:
  - **HOME** contiene la directory d'utente
  - **USER** contiene il login d'utente
  - **PWD** contiene la directory corrente
  - **SHELL** contiene il nome della shell di connessione
  - **PATH** contiene l'elenco delle directory dove si trovano i comandi che l'utente può eseguire
  - **HOSTNAME** contiene il nome del computer
  - **HISTSIZE** contiene la dimensione massima dei comandi eseguiti contenuti nel file cronologia
  - **PS1** contiene le impostazioni di visualizzazione del prompt

L'ambiente shell (II)

- Ad esempio per modificare la variabile PATH potremmo scrivere:

PATH=$PATH:.

- oppure:

export PATH=$PATH:.

- La differenza sarà che nel secondo caso il cambiamento sarà disponibile per qualsiasi processo verrà eseguito dalla shell
  - anche, ad esempio, dopo una chiamata ad una **exec**.

Comandi, argomenti e opzioni

- Una linea di comandi sarà quindi formata da parole (stringhe) separate da spazi o TAB
  - La prima parola e' il comando
  - Il resto sono gli argomenti
  - Gli argomenti sono spesso nomi di file, ma non necessariamente:
    - ✓ ***mail bc@dia.unisa.it***
- Una opzione e' un argomento speciale che impartisce istruzioni al comando, cioe' modifica il comportamento di default
  - ad es. ***ls -l***
  - A volte un'opzione ha un proprio argomento.

File e directory

- Filesystem gerarchico:
  - root **/**
- Nomi di file speciali:
  - dot **.**
  - dot dot **..**
- Pathname:
  - sequenza di zero o più nomi di file separati da **/**
- Pathname assoluto:
  - pathname che descrive la posizione di un file nel sistema a partire dalla root.
- Pathname relativo:
  - pathname che descrive la posizione di un file nel sistema a partire dalla directory corrente.

Directory

- Current working directory (cwd):
  - al login, cwd = home directory
- Conoscere la cwd:
  - **pwd**
- Cambiare directory di lavoro:
  - **cd**
- Tilde:
  - **~** : home directory
  - ***~user*** : home directory dell'utente "user".
- Per avere l'elenco dei file presenti nella cwd :
  - **ls**
    - ✓ Se si vuole conoscere l'elenco dei file presenti in una specifica directory, si fa succedere ls dalla pathname della directory

Filename e wildcard

- Wildcard: permette di specificare più file
  - **?** qualsiasi carattere (1 solo)
  - ***** qualsiasi sequenza di caratteri (0 o +)
  - **[set]** qualsiasi carattere in set
  - **[!set]** qualsiasi carattere non in set
- **?, *, !, [e]** sono caratteri speciali

Filename e wildcard: esempi

- *.txt
  - tutti i file che finiscono in .txt
- p?ppo
  - pippo, poppo, pappo, p1ppo, ...
- *.t[xyw]t
  - file che finiscono in .txt, .tyt, .twt
- [!abc]*
  - file che non iniziano con a op b op c
- *[0-9][0-9]
  - file che finiscono con 2 cifre
- [!a-zA-Z]*
  - file che non iniziano con una lettera

I/O

- I/O sotto Unix è basato su 2 idee:
  - un file I/O è una sequenza di caratteri
  - tutto ciò che produce o accetta dati è trattato come un file (inclusi i dispositivi hardware)
- standard files:
  - standard input (***stdin***)
  - standard output (***stdout***)
  - standard error (***stderr***)
- Alcuni comandi:
  - ***cat*** copia l'input nell'output
  - ***grep*** ricerca una stringa nell'input
  - ***sort*** ordina le linee dell'input

Ridirezione

- Il comando cat usato senza argomenti, prende input da stdin e manda l'output a stdout:

  cat

  Questa è una linea di testo.

  Questa è una linea di testo.

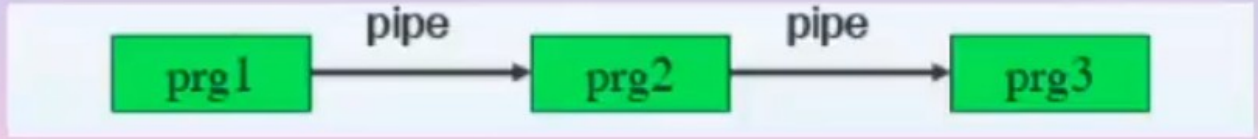
  ^D

- **< file1** fa si' che l'input sia preso da file1
- **> file2** fa si' che l'output vada nel file2
- **>> file** come prima, ma append se file esiste

  cat < file1 > file2

Pipeline

- Pipeline: serve a mandare l'output di un programma nell'input di un altro programma



- La barra verticale **|** rappresenta una pipe

ls -la | more

Processi in background

- Un processo e' un programma in esecuzione.
- Normalmente quando eseguiamo un programma da shell dobbiamo aspettare che il programma termini prima di dare un altro comando.
- Se invece volessimo far eseguire concorrentemente più processi che ad esempio non necessitano di input dall'utente possiamo eseguire in background tramite **&**.

```
tar –xzf archivio.tgz &
[1] 3156

[1]+ Done tar –xzf archivio.tgz
```

- ***jobs***: informazioni sui processi in background

I/O in processi in background

- I processi in background non dovrebbero avere I/O
  - Il terminale e' uno solo
  - Un solo processo puo' usufruirne:
    - ✓ il processo in foreground
- Se un processo in background fa I/O
  - Se vuole ricevere un input si blocca aspettando
  - Il suo output viene mescolato su video con l'output del processo in foreground e della shell.
  - Si possono usare le ridirezioni per evitare il problema

Caratteri speciali

- ~ home directory
- # commento
- $ precede il nome di variabile
- & processo in background
- ? * [ ] per wildcard (nomi file)
- | pipe
- ( ) inizio e fine subshell
- ; separatore di comandi
- { } blocco di comandi
- < > ridirezione
- ! simbolo di negazione
- / (slash) separatore directory nel nome del file
- \ (backslash) simbolo di “escape”

Backslash escaping

- Il backslash toglie il significato speciale al carattere speciale che lo segue
  - Per indicare il backslash ' \' oppure \\

echo 2 * 3 \> 5 espressione vera

2 * 3 > 5 espressione vera

help, echo, read

- **help**: manuale online

  help cd

- ***echo "messaggio"***
  - stampa a schermo messaggio
- ***read var1 var2***
  - legge valori di var1 e var2

    echo –n "Digita i valori di due variabili: "; read a b

    Digita i valori di due variabili: pippo pluto

    echo $a

    pippo

    echo $b

    pluto

history

- **history** visualizza tutti i comandi
- Selezionatore di comandi:
  - **!** inizia una ricerca nella “history”
  - **!!** esegue il comando precedente
  - **!47** esegue il 47-esimo comando
  - **!-23** il comando eseguito sarà dato dal numero del comando corrente – 23
  - **!str** ultimo comando che inizia per str
  - **!?str?** ultimo comando che contiene str

Personalizzare l'ambiente

- bash fornisce 4 importanti strumenti per personalizzare l'ambiente di un processo:
  - File speciali, Alias, Opzioni, Variabili di ambiente.
- File speciali
  - .bash_profile, .bash_logout, .bashrc che sono letti da bash quando avviene il login o il logout o quando viene aperta una nuova shell
- Alias
  - Serve a creare sinonimi per comandi

    alias la="ls -la"
  - bash esegue una sostituzione testuale quando incontra il nome di un alias.
  - **alias** senza argomenti restituisce la lista di tutti gli alias.
  - ***alias nome*** senza il segno di =
    - ✓ restituisce il valore di nome dove nome e' un alias precedentemente definito.

Personalizzare l'ambiente (II)

- Opzioni
  - Controllano vari aspetti dell'ambiente
- Per verificare lo stato delle opzioni dare il comando

  set -o
- Variabili di ambiente
  - Contengono valori che possono essere cambiati così che la shell o altri programmi possono avere comportamenti diversi secondo il contenuto di tali variabili.
  - Controllano il modo in cui la shell si comporta
  - Forniscono informazioni ai processi

Script

- Uno script e' un file con comandi shell.
  - Se scriviamo in un file **scr** i comandi shell:

```
#!/bin/bash
echo Questo e\' un script
echo I file in Vbin sono:
ls /bin
```

- Per eseguirlo:
  - ***source filename***
  - oppure ***chmod +x filename*** seguito da ***./filename***
- Il secondo metodo lancia una subshell ed esegue lo script nella subshell.
- Prima riga **#!comando**, opzionale,
  - viene usato **comando** per interpretare lo script.