

Basic Unix Commands

Bash Commands

- **man <cmd>**
 - Shows manual for a command.
- **sudo <cmd>**
 - Execute a command as the superuser.

File System Navigation

- **cd <path>**
 - Change current working directory.
- **ls [<opts>] [<path>]**
 - List the contents of a directory (current working directory by default).
 - Options:
 - **-l**: use a more detailed listing format.
 - **-a**: show hidden files and folders.
- **pwd**
 - Print name of the current working directory.

File System Manipulation

- **mkdir <dirpath>**
 - Create a new directory, if it does not already exist..
- **rmdir <dirpath>**
 - Remove a directory, if it is empty.
- **mv <srcpath> <destpath>**
 - Move (rename) the file <srcpath> to <destpath>.
- **rm [<opts>] <path>**
 - Remove a file.
 - Options:
 - **-R**: remove directories and their contents recursively.
- **cp [<opts>] <srcpath> <destpath>**
 - Copy the file <srcpath> to <destpath>.
 - Options:
 - **-R**: copy directories and their contents recursively.
- **ln [<opts>] <target> <destpath>**
 - Create an hard link to file <target> in <destpath>.
 - Options:
 - **-s**: make symbolic links instead of hard links.

Process Management

- **ps [<opts>]**
 - List active processes.
 - Options:
 - **-e**: list all processes.
 - **-l**: use a more detailed listing format.
- **top**

- Show a a dynamic real-time view of running processes.

- **kill -<signal> <pid>**
 - Send the signal <signal> to the process specified by <pid>.
 - Signals:
 - **-9**: terminate a process.
- **<cmd> &**
 - Execute command in background.

File Operations

- **cat <filepath>**
 - Print file on the standard output.
- **more <filepath> / less <filepath>**
 - Print file on the standard output page by page.
- **sort [<opts>] <filepath>**
 - Sort lines of text files (alphabetically by default).
 - Options:
 - **-r**: reverse order.
 - **-n**: use numerical order.
 - **-k <key>**: sort via key.
- **cut [<opts>] <filepath>**
 - Remove sections from each line of a text file.
 - Options:
 - **-f <field1,field2,...fieldN>**: specify fields to select.
 - **-d <delim>**: use <delim> instead of TAB for field delimiter.
- **tr [<opts>] <set1> [<set2>]**
 - Translate standard input characters from <set1> to <set2>.
 - Options:
 - **-d**: delete characters from <set1> instead of translating them.
- **uniq [<opts>] <filepath>**
 - Omit repeated lines.
 - Options:
 - **-d**: report only repeated lines instead of omitting them.
- **grep [<opts>] <filepath>**
 - Print lines of a file matching a pattern.
 - Options:
 - **-e <pattern>**: specify a pattern to be matched.
 - **-E <pattern>**: specify extended regular expression pattern to be matched.
 - **-H**: print the file name for each match.
 - **-n**: print the line number for each match.
 - **-i**: ignore case.
 - **-v**: invert the sense of

matching, to select non-matching lines.

- **--quiet, --silent, -l** no output is produced.
- **--files-with-matches, -l** print out only file names.

- **wc [<opts>] <filepath>**
 - Print newline, word, and byte counts for a file.
 - Options:
 - **-l**: print newline count only.
 - **-w**: print word count only.
 - **-c**: print byte count only.

File Search

- **find [<opts>] [<rootpath>]**
 - Search for files in a directory hierarchy (with a specified root).
 - Options:
 - **-name <pattern>**: search files whose name matches the pattern.
 - **-regex <pattern>**: search files whose path matches a regular expression.
 - **-regextype posix-extended**: specify posix-extended format for regular expressions
 - **-type <f|l|d>**: search files of a specific type.
 - **-mindepth <depth>**: search files starting from the specified directory tree depth.
 - **-maxdepth <depth>**: search files up to the specified directory tree depth.
 - **-size <[+,-]n[ckwMG]>**: search files whose size starts from (+) or goes up to (-) the specified size. (c=bytes, w=words, k=kilobytes, M=megabytes, G=gigabytes).
 - **-exec <cmd>**: execute command on each matched file.
 - **"{}"** can be used as a placeholder for the file path.
 - The command must end with **"\;"**.
- **chmod [<opts>] <mode> <file>**
 - Change file permissions. <mode> can be specified symbolically (lugoa) [+][rwx] or numerically (octal digits).
 - Options:
 - **-R**: change permissions

File Permissions Management

of files and directories recursively.

String manipulation

- **basename path**
 - Strip directory and suffix from path.
- **dirname path**
 - Strip last component from path.

Redirections

- **cmd1 | cmd2**
 - Redirect standard output of cmd1 to standard input of cmd2.
- **cmd < file**
 - Redirect standard input of cmd from file.
- **cmd > file**
 - Redirect standard output of cmd to file.
- **cmd 2> file**
 - Redirect standard error of cmd to file.
- **cmd &> file**
 - Redirect standard output and standard error of cmd to file.
- **cmd >> file**
 - Append standard output of cmd to file.

Shortcuts

- **CTRL+C**
 - Terminate the current foreground process.
- **CTRL+Z**
 - Stop the current foreground process.
- **TAB**
 - Autocompletion.

The Bash Language

Assegnazione di variabili

nome_var=valore

Comando echo (per stampa su stdout)

echo [OPZIONI] [STRINGA]

- Opzioni:
 - **-n**: per non andare a capo.
 - **-e**: per interpretare i caratteri di escape.

Comando read (per la lettura da stdin)

- read
- con una o più variabili passate come argomento
 - uso della variabile \$REPLY

Quoting

- **Single quoting "**
 - le variabili non vengono espanse
- **Double quoting '"**
 - le variabili vengono espanse
- **Es:**
 - **a=pippo**
 - **echo "\$a pippo" "\$a" pluto**
 - **pippo pippo \$a pluto**
 - **echo \$a pippo '\$a' pluto**
 - **pippo pippo \$a pluto**

Utilizzo delle parentesi { } per delimitare il nome di una variabile

- **Es.**

o nome=Gian
o echo \${nome}marco
o Gianmarco

Cattura del stdout di un comando

- \$(comando)

Esecuzione di script

- #!/bin/bash
 - da file
- (cd /home; ls)
 - diretta, tramite l'utilizzo delle parentesi ()

Comando exit

- exit [numero]
 - termina l'esecuzione di un processo restituendo un valore al processo chiamante
- Es:
 - exit 0
 - restituisce un valore vero

Esecuzione di calcoli aritmetici

- un metodo a scelta dello studente
- Es.
 - let s=\$n1+\$n2
 - Assegna alla variabile \$s la somma di \$n1 e di \$n2

Variabili di shell speciali

- \$0, \$1, \$2, ...
 - passaggio di parametri sulla riga di comando
- \$*
 - lista completa dei parametri, escluso il nome dello script
- \$#
 - Numero di parametri
- \$\$
 - PID del processo
- \$?
 - valore di ritorno dell'ultimo processo eseguito

Costrutto if-then-else (e elif)

if condizione ; then
statements
elif condizione
then
statements
else
statements
fi

Costrutto while (compresa ridirezione di stdin e stdout)

while condizione
do
statements
done << \$fileIn >> \$fileOut

Formati richiesti nella condizione dei costrutti if e while

Solo le condizioni espresse tra simboli [...] sono richieste (non sono invece richieste le condizioni basate sulla parola chiave test)

- Confronti numerici:
 - eq uguaglianza (==)
 - ne non uguaglianza (!=)

o -gt maggiore (>)
o -ge maggiore o uguale (>=)
o -lt minore (<)
o -le minore o uguale (>)

- Confronti tra stringhe:
 - = uguaglianza
 - != non uguaglianza
- Condizioni su file:
 - d <arg> vero se <arg> è un directorio
 - f <arg> vero se <arg> è un file
 - r <arg> vero se <arg> ha il permesso di lettura
 - w <arg> vero se <arg> ha il permesso di scrittura
 - x <arg> vero se <arg> ha il permesso di esecuzione
- Operatori logici utilizzabili all'interno di una condizione:
 - ! not
 - a and
 - o or
- Operatori logici utilizzabili in un elenco di condizioni:
 - && and
 - || or

Costrutto for

for var in [list]
do
statements
done

Istruzioni

- break
- continue

Vettori

Dichiarazioni
array[3]="valore"
array=(4 8 7)
array=([0]=4 [1]=8 [2]=7 [5]=10)
Accesso
echo \${array[1]} # Accesso all'elemento 1 dell'array (valore 8)
echo \${array[*]} # Stampa tutti gli elementi dell'array
echo \${#array[*]} # Numero di elementi contenuti nell'array

AWK

Invocare AWK

Il comando per invocare AWK è il seguente: awk [opzioni] {script} [file1] ... [filen]
In cui:
[opzioni]: -v var=val
- definisce la variabile var e le assegna il valore val

{script}: Indica il file comandi da eseguire

- awk [opzioni] 'comando'
- [file1] ... [filen]
- awk [opzioni] -f scriptFile [file1] ... [filen]

[filei]: Uno o più file di ingresso

AWK SCRIPT

Variabili built-in

RS = Separatore di record [default \n]

FS = Separatore di campo

NR = numero di record

NF = numero di campi

\$0 = intero record

S1= primo campo ; \$2=secondo campo

FILENAME = nome file corrente in lettura

ARGV[0] memorizza il nome dello script
ARGV[1] ... ARGV[ARGC-1] memorizzano i parametri

Condizioni

(nulla)
- Verificata per ogni record

/regExp/
- La condizione è vera se l'espressione ha un match con il record

espressione
-La condizione è vera se diversa da zero oppure non è nulla (nel caso in cui ritorni una stringa)

pat1, pat2
- Coppia di pattern separati da virgola
- Specificano un intervallo di record

BEGIN
- Serve per eseguire un'azione all'inizio dello script, ovvero prima di analizzare il file di ingresso
END
- Serve per eseguire un'azione al termine dello script, ovvero alla fine del file di ingresso

Funzioni / Comandi utili

Stampa

print [p1] ... [pn] [> file]
I parametri [pi]
- Non sono

separati da virgola
- Sono stampati

separati da uno spazio

printf (format, ...) [> file]
- Stampa (formattata) su file

stile C

getline [var] [<altroFile]
permette di
Prelevare il record successivo
mentre si sta manipolando quello corrente
Leggere righe da un secondo
file

int(x), sqrt(x), exp(x), log(x), sin(x), rand(), etc.
Principali funzioni matematiche numeriche

length (str)
Ritorna la lunghezza della stringa str
Se str è un numero ritorna la lunghezza del numero trasformato in stringa

toupper (str)
Restituisce la stringa str trasformata in maiuscolo

tolower (str)
Ritornano la stringa str trasformata in minuscolo

system (command)
Chiama una shell per eseguire il comando

match (str, regExp)
Ricerca all'interno della stringa str l'espressione regolare regExp

gsub (regExp, str [, src])
Sostituisce nella stringa src ogni occorrenza (non sovrapposta) dell'espressione regolare regExp con la stringa str

split (str, vet [, del])
Suddivide la stringa str in sotto-stringhe secondo il delimitatore del e mette nel vettore vet ogni sotto-stringa

substr (str, i [, n])
Ritorna al massimo n caratteri della stringa str a partire dal carattere i-esimo

Operatori ed espressioni

Aritmetici
+, -, *, /, %, ^ (oppure **)

Di concatenazione di stringhe
str1 str2 str3 ...

Di confronto
==, !=, <, >, <=, >=

Logici
&&, ||, ! (negazione)

Di confronto con una espressione regolare
~, !~