1 Bash

Variabili	Descrizione
PATH	V. Ambiente: Modificabile, sequenza di percorsi assoluti, divisi da ":", di directory contenenti eseguibili (lanciabili senza digitare path). Ricerca secondo ordine specificato in PATH, si ferma a primo eseguibile con nome uguale. Eventuale ErrorNotFoutd. Altre variabili d'ambiente: $HOME, $USER, $SHELL, $TERM.
env	Visualizza l'elenco delle variabili d'ambiente.
IFS=$' \t \n'	Contiene caratteri separatori della parole negli elenchi.
/dev/null	File speciale che scarta tutto quello che gli viene scritto.
LC_CTYPE	Imposta il tipo locale dei caratteri. Usata da `cut` e `ws` per capire numero byte caratteri.
strace *comando*	Dice la sistem-call usata.
$RANDOM	Restituisce un numero intero casuale tra 0 e 32767.

File speciali	Descrizione
/etc/passwd	Righe sono info ogni utente divise da ":".
/etc/shadow	Righe sono password utente codificate.
/etc/group	Righe sono info ongi gruppo divise da ":".
/usr/bin/passwd	Cambia la pass utente.

Directory	Descrizione
cd *percorso*	Sposta logicamente la directory corrente in quella specificata, secondo un path asssoluto o relativo.
mkdir *nomeDir*	Crea una nuova directory.
touch *nomeFile.estensione*	Crea un file vuoto nella dyrectory corrente.
rmdir *nomeDir*	Rimuove una directory, solo se è vuota.
rm *file|dir*	Rimuove una directory vuota o un file.
rm *-r file|dir*	Elimina ricorsivamente sotto cartelle e file.
rm *-f file|dir*	Non fa chiedere le autorizzazioni di eliminazione.
mv file1 file2|dir	Rinomina file1 in file2 o sposta file1 nella directory specificata.
cp file1 dir	Copia file1 nella directory specificata.
ls [*nomefile*]	Visualizza i files/direcotry contenuti nella directory corrente. Se specifico un file mi dice se esiste e applica le opzioni su di lui.
ls -a	Mosta anche file nascosti (anche ., ..).
ls -l	Mostra informazioni file e relativi permessi nel formato: 1°carattere tipo file (- file, d dyrectory, c collegamento seriale, b device a blocchi), 9 caratteri successivi terzine di permessi (r read,w write,x/s execute) per proprietario, gruppo, altri utenti.
ls -h	Rende i dati più leggibili.
ls -d	Fa applicare il comando alla directory stessa/specificata e non ai file.
ls -R	Mostra ricorsivamente contenuto sotto directory.
pwd	Visualizza il percorso assoluto, da / fino alla directory corrente.
which *comando*	Cerca in PATH il comando e se lo trova mostra il path in cui si trova.

Directory `/proc` : info sui processi, con una dir. per ogni processo attivo, create e cancellate continuamente. Directory sulla RAM `/proc/` : contiene i processi in esecuzione e nella sottodirectory `fd` i file descriptor aperti.

Tilde Expansion	Descrizione
/...	Tilde espansa con il percorso assoluto della home directory dell'effective user. Valido caso con solo e solo /.
userName/...	Tilde e userName espansi con il percorso assoluto della home directory dell'utente specificato. Valido caso con solo **userName/**.

Privilegi	Descrizione
chmod *privilegi script.sh*	Modifica permessi file mediante formato numerico (terna 0-7). Ogni numero è la somma dei valori associati hai permessi di r(4), w(2), x/s(1). Ordine: proprietario, gruppo, altri utenti. Può diventare un quartetto aggiungendo per primo l'identificatore numero dei privilegi di *setuid, setgid, sticky bit.*
chgrp *nuovoGruppo file*	Modifica il gruppo di appartenenza di un file.
chown *newOwner nameFile*	Modifica il proprietario (e anche gruppo) di un file.
whoami	Dice all'utente corrente le sue informazioni.
sudo *comando*	fa eseguire il comando come administratore, può essere chiesta userPass. Solo utenti gruppo sudo (gestito dall'admin) possono usarlo.

Controllo Comandi	Descrizione
\	Disabila interpretazione per il carattere successivo (ex: andata a capo) permettendo di stamparlo.
"..."	Delimita un argomento e non fa interpretare nessun comando a eccezzione espansione di variabili e esecuzione di comandi.
'...'	Delimita un argomento e non fa interpretare nessun comando.
$'...'	Espande backslash-escaped direttamente nella stringa espansa in una single-qouted stringa. Backslash-escaped: \a, \e, \f, \r, \v, \', \b, \E, \n, \t, \\, \", \nnn, \xHH, \cn.
pre{s1,...}post	Stringa di testo racchiusa fra separatori (spazio, tab, a capo) con coppia di graffe (non precedute da $) e senza separatori. Le stringhe racchiuse dalle graffe vengono composte con il preambolo (pre) e postscritto (post), che sono opzionali. Sono annidabili (quelle più esterne eseguite per prime). Vengono eseguite prima le brace expansions delle variable expansions.
pre{a_1..a_2}post	Lettere da a_1 a a_2 nell'alfabeto.
pre{n_1..n_2}post	Numeri compresi tra n_1 e n_2.
*	Sostituito con una qualsiasi sequenza di caratteri anche vuota.
?	Sostituito con un singolo carattere (no spazio vuoto).
[c1c2...]	Sostituito con solo uno dei caratteri specificati in elenco. Annidabili.
[a_1..a_2]	Sostituisce con solo una lettere da a_1 a a_2 nell'alfabeto.
[n_1..n_2]	Sostituisce con solo un numero dei numeri compresi tra n_1 e n_2.
[:digit:]	Sostituisce con una cifra.
[:upper:]	Sostituisce con un carattere maiuscolo.
[:lower:]	Sostituisce con un carattere minuscolo.
history	Visualizza comandi, numerati, precedentemente eseguiti, anche da shell precedentemente chiuse. Con !NUMERO lancio il comando corrispondente nell'elenco di history. Con !STRINGA lancio il comando più recente che corrisponde alla stringa.

EXP aritmetica	Descrizione
((...))	Valuta una stringa come un espressione aritmetica (+,-,*,/,%, (), !, &&, ||) di soli interi. Racchiude un espressione più eventualmente un assegnamento. Si possono usare variabili nell'espressione ($variabile). Exit statuts 0=true, altro=false. Non contiene espressioni condizionali e comandi. Per le operazioni in virgola mobile usare **bc**.
$((...))	Come operatore ((...)) ma è concatenabile con stringhe tramite " ".

Conditional EXP	Descrizione
[[...]]	Restituisce exit status 0=true, altro=false. • Non può contenere conmandi, word splitting, brace expansion, pathname expansion. Non esegue assegnamenti a variabile, annidamenti di espressioni condizionali. • Può contenere variable expansion, solo $(()), command substitution (se non genera comandi), process substitution, quote removal. Questi solo per gli operandi. • Può andare a capo. • Si possono usare gli operatori logici !, &&, ||, (). Operatori unari/binari non quotabili ne generaibili da command substitution: • Operazioni sui file: -e (esistenza), -d (cartelle), -f (file), -h (link), -r (leggibile), -s (size>0), -t (fd open e riferisce un terminale), -w (scrivibile), -x (eseguibile), -O (possesso effettivo utente), -G (possesso effettivo gruppo), -L (=h). Confronto date ultima modifica file: f1 -nt f2 (f1 nuovo o f2 inesistente), f1 -ot f2 (f1 veccio o inesistente). • Operatori aritmetici: -eq (==), -ne (!=), -le (<=), -lt (<), -ge (>=), -gt (>). • Operatori lessicografici: ==, =, !=, <, <=, >=, >, -z (size==0), -n (size!=0). -o da vero se opzione è abilitata per la shell. Versioni vecchie: [..], test ... no a capo (semmai \), -a, -o, no (). Mettere sempre spazi prima e dopo quadre.
cmd1 ||cmd2	Esegue cmd1 e solo se cmd1 fallisce (exit status≠0) esegue cmd2.
cmd1 && cmd2	Esegue cmd1 e solo se cmd1 ha successo (exit status=0) esegue cmd2.

Scripting	Descrizione
clear	Pulisce la CLI. Non modifica variabili create.
./ comado	Esegue comando presente nella directory corrente (percorso relativo). Alternativa è inserire il percorso relativo o aggiungere il percorso assoluto alla variabile PATH.
nome=*valore*	Simboli con nome e valore, stringa modificabile, alfanumerici casesensitive. No spazi prima o dopo "=". Sono d'ambiente o ex-novo locali. Solo la bash in cui sono create le variabili le può usare. I programmi lanciati dalla bash hanno una speudocopia della bash.
unset *nomevariabile*	Elimina una variabile esistente (vuota o no). Quotare ("...") sempre variabili per evitare errori con variabili vuote o inesistenti.
${variabile}	Espanzione di variabile, ovvero la stostituisce con il suo contenuto. Graffe opzionali se nome variabile seguito da uno spazio.
${#variabile}	Restituisce il numero di caratteri del contenuto della variabile.
${!variabile}	Fa l'espanzione della variabile che contiene il nome d'un altra variabile con il valore di quest'ultima (riferimento indiretto). Dalla versione 2 di bash.
cmd1 ; ...	Separatore di più comandi, e dei rispettivi argomenti, scritti sulla stessa riga di comando e eseguiti dopo la terminazione del precedente (lista di comandi). L'exit status è quello dell'ultimo comando lanciato. Se racciusi da () eseguiti in una sub-schell.
#...	Commeto.
#!...	Scritto nella prima riga indica quale interprete deve eseguire lo script. Se non specificato viene usato quello corrente.
script.sh c1...	I parametri sono un'insieme ordinato di caratteri separati da spazi successivi al nome del programma. Sono immodificabili dopo la sostituzione dei metacaratteri (*, ?,ecc). Riga di comando = nomeProgramma + parametri. Nella riga di comando gli elementi sono indicizzati da 0 (nomeProgramma): • **$#** Contiene il numero di parametri passati. • **$n** Accede all'n-esimo parametro a partire da indice 0. • **$*** Tutti argomenti concatenati e divisi da spazi. • **$@** Vettore di argomenti quotati ("..."). I parametri $* e $@ sono identici se non quotati (concatenazione di argomenti separati da " "). Se quotati $* quota tutti gli argomenti assieme mentre $@ quota singolarmente ogni argomento. $@ è usato per passare parametri a comandi dentro a degli script.
ècomando|./script.sh`	**Command substitution**. Sostituisce (a run-time) nella riga in cui è specificato il comando o script con l'output (stdout). Comando alternativo $(./script.sh)
$?	Modificato alla terminazione di ogni script, contiene l'*exit status*.

Subshell	Descrizione
bash	Crea una shell figlia (interattiva non di login). Eredita dal padre: dir. corrente, copia variabili d'ambiente. Non sono ereditate le variabili locali. Creata in automatico per comandi raggruppati, script, processi in background. I comandi built-in sono eseguiti in shell corrente/padre.
bash *-c script.sh*	Crea shell non interattiva.Vi si può far eseguire uno script specificato.
bash *-l |–login script.sh*	Crea shell interattiva di login. Vi si può far eseguire uno script specificato.
export nomevar |nomevar=valore	Crea una variabile d'ambiente. Una shell figlia ne riceve una copia modificabile che non influenza la variabile d'ambiente del padre.
var=val ... comando	Scrivendo delle assegnazioni prima dell'esecuzione di un comando si creano delle var. d'ambiente solo per l'imminente subshell. Non saranno ereditate da successive subshell.
. |source *script.sh*	Esegue lo script nella shell corrente. Utile a impostare/modificare variabili shell. Ignorata prima riga opzionale e eseguito con interprete corrente.
exit	Termina bash corrente, elimina l'ambiente e sale alla padre.
exit *exitStatus*	Termina lo script restituendo un valore intero [0-255] per indicarne l'esito di terminazione. 0 indica esecuzione terminata senza errori, qualcosaltro indica un'errore. Viene restituito alla shell esecutrice.
top	Mostra in tempo reale processi in esecuzione e risorse di sistema usate.
ps	Mostra i processi in esecuzione.
ps -all	Mostra i processi in esecuzione con informazioni aggiuntive (PID, PPID,ecc).
set	Visualizza sia variabili locali che d'ambiente della shell corrente (anche funzioni di shell). I parametri [re]settano dei comportamenti della shell:
set +o *comando*	Disalibilita il comando (tipo history).
set -o *comando*	Abilita il comando (tipo history).
set -a	Successive variabili create/modificate diverrano d'ambiente e ereditate da shell figlie. Per [ri]definire variabili locali usare **export -n *variabile***.
set +a	Successive variabili create/modificate diverrano locali (non ereditabili da shell figlie), modalità di default.

File Descriptor	Descrizione
$$	Variabile con PID della shell corrente. Utile per esplorare /proc/ e fd vari processi.
exec {*n*}*modalità file.est*	Apre file e l'associa al fd scento dall'utente (n). Modalità: < (lettura), > (scrittura), >> (aggiungi in coda), <> (lettura e scrittura).
exec {*var*}*modalità file.est*	Apre file e l'associa al fd scento dall'os inserito in var.
exec {n}>&-	Chiusura di file con fd=n.
exec {*var*}>&-	Chiusura di file con fd contenuto in var.

Uso File Descriptor	Descrizione
program *modalità file|prog2 ...*	Ridirezionamento std di program. Modalità: < (strin da file), > (strout sovrascrive file), >> (srtout accoda a file). Possibili più ridirezionamenti contemporaneamente (ordine non conta). Il fine file da tastiera si fa con "Ctrl+D".
program *modalitàEfd* file|prog2 ...	Ridireziona il file o prog su uno specifico fd di program. Modalità: N>, N>>, <N.
program 0/1/2/&> file|&0/1/2	Ridireziona lo std (&=stdout+stderr) indicato di program nel file indicato sovrascrivendo o in un altro str. &> ridireziona solo in fle.
program N>&M ...	Fa puntare lo stream di N allo stesso stream di M (N e M sono fd). Si possono fare più ridirezionamenti (fd e file, fd e fd) in simultanea. Sono eseguiti da sinistra a destra.
prog1|prog2	pipe: collega stdout di prog1 allo stdin di prog2 (usa speudo-file temporaneo). Unidirezionale sinDes.
prog1 |& prog2	Ridireziona lo stderr nello stdout di prog1 e lo passa allo strdin di prog2.
costrutto *modalità file*	Applica il ridirezionamento a tutti i comandi contenuti nel costrutto e alla condizione.
program << *word*	Word specificata è delimitatore di fine input da passare al comando. Nell'input la word deve essere a inizio di una riga con solo essa.
program <<< *word*	Ridireziona in input la word. Se si qouta ("...") si possono passare più dati.
(cmd1;...)	Comandi eseguiti in subshell con stdin/out/err condivisi e concatenati. L'exit status è quello dell'ultimo comando eseguito. Permette ridirigere più comandi simultaneamente.

Standard input e output	Descrizione
read *var*	Legge dallo stdin o file una sequenze di caratteri (fino all'INVIO o a capo) e la inserisce nella variabile passata. Ingnora spazi e tabultazioni inizio e fine riga (ameno che si setti IFS=""). Exit status 0=lettura eseguita, >0=eof (var="") o ultima riga senza \n (var="ultimi caratteri"). Necessario controllo con #var. Una read parte dalla riga precedente, se non era stata finità di leggere (fino a \n).
read *-u ${var1} var*	Leggere dal file descriptor var1.
read *-n numero var*	Numero massimo di caratteri da leggere.(Considera spazi e tabulazioni).
read *-N numero var*	Numero esatto di caratteri da leggere.(Considera spazi e tabulazioni). Considera \n come semplice carattere. Usa prossime righe se attuale ha caratteri insufficente.
read *var1 ...*	Alle variabili viene assegnata i-esima parola della riga tranne per l'ultima variabile che riceve la stringa rimanente. Alle variabili in eccesso viene assegnato valore vuoto.
echo *testo*	Visualizza a video la sequenza di caratteri passata fino al primo "INVIO". Se quoto l'input si disabilità l'interpretazione dei caratteri speciali e andate a capo.
echo *-e testo*	Stampa i caratteri speciali (\...).
echo *-n testo*	Non fa andare a capo.

Utilities	Descrizione
head -n *n file*	Mostra le prime n righe del file. Default 10.
tail -n *n file*	Mostra le ultime n righe del file. Default 10.
sed **'s/old/new/[g|n];...' file**	sostituisce il testo old (o più testi orinali con [o1, o2, ...]) con quello new, solo per la prima occorrenza. Può richiere l'opzione `-r`. • Con l'opzione g si applica per tutte le occorrenze in una riga oppure a "n" occorrenze. Si possono concatenare più sostituzioni d'applicare sulla stessa riga. • ^ significa inizio linea. Va all'inizio. • **$** significa fine riga. Va alla fine. • **.** significa un carattere qualsiasi. **.{n}** significa n caratteri qualsiasi. **.*** significa 0 o più caratteri qualsiasi. • **[...]** caratteri o insiemi di caratteri (a1-an) da sostituire. backreference: In "originale" si specificano tra () stringhe riusabili in "nuovo" traminte i-esima backreference.
cut *-c S-D* file	Estrae colonna/intervallo di caratteri, da ogni riga, dall'indice S all'indice D (inclusi). S o D si possono omettere e si prende dall'inizio o dalla fine fino a S o D. L'indice parte da 1. Posso concatenare più intervalli dividendoli con ",".
cut *-d '...' -fN* file	Divide le righe in parole secondo il carattere delimitatore indicato poi seleziona la parola con l'indice f-ennesimo di ogni riga.
cat *nomeFile.estensione*	Visualizza il contenuto del file.
grep *stringa* [*file*]	Filtra le righe contenenti la stringa passata (che sarà evidenziata), anche se contenuta in altre parole, dal testo che sarà successivamente scritto (terminato da "Ctrl+d") o dal file passato opzionalmente. Spesso usato come secondo comando di |per filtrare output di altri comandi.
grep -v *stringa* [*file*]	Filtra righe senza parola da evidenziare.
grep -i *stringa* [*file*]	Disabilità il caseSensitive.
grep -h *stringa* [*file*]	Disabilita la visualizzazione del nome del file dove è avvenuta la ricerca (nel caso si cerci in più file). (-H è equivalente).
find *file*	Trova percorsi di file in sotto alberi della memoria.
find -type d *file*	Trova solo directory.
find -type f *file*	Trova solo file.
find -maxdepth n *file*	Limità la profondità di ricerca il numero di livelli indicato.
find -mindeph *file*	Indica il livello minino di ricerca.
find -name "..." *file*	Definisce un vincolo sul nome del file da trovare.
find -iname *file*	Disabilità il caseSensitive.
find *file* -exec ... '{}' ... \;	Istruzioni da fare quando viene trovato un elemento. L'output di find viene messo al posto di '{}'.
wc	Dice il testo passatogli il numero di righe (-l), parole (-w) e caratteri (-c).
tee file	Stampa a video e salva l'output di un comando (che gli manda l'input, tipo con —), sul file indicato.
sort -k n	Ordina le righe alfabeticamente secondo la colona di stringhe specificata.
uniq *file*	Rimuove le righe duplicate consecutive.

Costrutti controllo flusso	Descrizione
for varName in elencoParole ; do listCommand ; done	Dopo il for variabile è usabile.
for ((exp1 ; exp2 ; exp3)) ; do listCommand ; done	Le exp sono espressioni valutate aritmeticamente automaticamente.
if listA ; then listB ; [elif listC ; then listD ; ...] ... [else listZ ;] fi	
while list ; do list ; done	

In tutti i costrutti l'exit value è o quello della lista do comandi o 0 se nessun comando viene eseguito. Espressioni condizionate su file o variabile: [condizione di un file]. Valutazione di un espressione matematica applicata a variabili d'ambiente: ((istruzini con espressione)).

Process Identifier	Descrizione
$$	Variabile espansa col PID (identificatore numerico univoco di processo) della shell corrente. Col raggruppamento di comandi `(...; ...)` viene espanso col PID della shell più esterna.
$BASHPID	Da in ogni caso il PID della shell corrente. Comando poco portabile.
script &	Lancia processo in background. Si omette ; (& già terminatore comando). `$!` ha il pid del processo lanciato in bg (anche se sganciato). Non è aggiornato se faccio bg di un processo sospeso da ^Z.
jobs	Mostra lista procesi backgroud e sospesi con rispettivo indice di job. Con `%indiceJob` si gestisce il job.
^Z	Sospende processo in foreground. Lancia il segnale SIGTSTP
^C	Termina processo in foreground. Lancia il segnale SIGINT.
bg [%n]	Riprende l'esecuzione in background di un Job sospesto/del Job specificato. Lancia il segnale SIGCONT.
fg [%n]	Riprende l'esecuzione in foreground di un Job sospesto/del Job specificato. Lancia il segnale SIGCONT.
kill [-valore] pid|%n [-1]	Usato per termina il processso di PID o JobId indicato. In generale manda un segnale (interruzione software per notificare evento asincrono), default SIGTERM=15. In generale un segnale è accettato solo se proviene dall'effective user e poi viene gestito/ignorato dal processo interessato (SIGKILL=9 non può essere ignorato). Con -1 si fanno terminare tutti i figli diretti della bash.
kill -l	Da tutti segnali lanciabili.
killall 'nomeProc'	Termina tutti processi con lo stesso nome o mandare a più processi uno stesso segnale. Manda SIGTERM di default. `-r` Permette di usare le espressioni regolari.
disown pid|%n, ...	Sgancia jobs dalla shell. Viene sganciato l'ultimo job messo in bg se non si specifica nulla.
disown -r	Sgancia tutti job in running.
disown -a	Sgancia tutti job in running e suspend.
nohup *script* &	Lancia in bg e sgancia ultimo processo messo in bg.
%%|%+|%	Riferisce job corrente (ultimo interrotto) in foreground o avviato in backgourd.
%-	Riferisce penultimo job interrotto in foreground o avviato in backgourd.
trap "azione" segnale1,...	Imposta "solamente" l'azione (funzione o lista comandi) da svolge alla ricezione di certi segnali predefiniti. Funzione: `nomeFunzione(){...}`
wait pid|%n, ...	Fa attendere la shell la terminazione del processo figlio di pid o JobId indicato (o un insieme di processi). Ha exit status l'exit status dell'unico o dell'ultimo processo terminato da attendere. `wait` senza parametri fa attendere tutti figli e da exit status 0.
sleep n	Fa attendere n secondi al programma.

Terminologia:

- **Processo**: Minima unità completa ed'autonoma d'elaborazione. Composto da più thread operanti in un contesto = spazio d'indirizzamento + tabella dei descrittori di file aperti.
- **Gruppo di processi**: Insieme variabile di processi avviati da un processo padre.
- **Terminale di controllo**: astrazione della console=video+tastiera (stdni, stdout, stderr) opzionale per un processo.
- **T.c. del gruppo di processi**: terminale controllo padre ereditato dai figli, ameno di sganciamenti volontari.
- **Sessione**: Creata e legata a un terminale di controllo da una shell (leader sessione). sessionId=PID leader sessione. Shell prende dal terminale di controllo stdin, stdout, stderr.
  Processi figli apparteranno alla sessione, ameno di distacchi, e condividono terminale di controllo.
  Un terminale è t.c. per una sola sessione alla volta. La ciusura del t.c. comporta la terminazione dei processi del gruppo (SIGHUP).
- **Processo foregroud**: Controlla terminale da cui è stato lanciato (stdin collegato e impedisce esecuzione fino alla sua fine ad'altri programmi).
  Uno processo foregroud alla volta.
- **Processo backgroud**: Eseguito in parallelo rispetto bash madre (bash può leggere e eseguire indipendentemente). Hanno una copia fd della bash.
  Terminati, ameno di sganciamento, alla chiusura del terminale della bash madre (stdin condiviso).
  Più processi backgroud alla volta.
- **Job control**: Componente che cambia stato processi da background a foreground e viceversa.
- **Jobs**: Processi in background o figli sospesi di una shell
- **Processo zombie**: Processo figlio terminato (risorse state rilasciate dall'os) di cui l'os mantiene il pcb (pid, exit status, ecc).
  Del tutto eliminato quando padre fa la wait (rilascio pcb).
- **Processo orfano**: Processo figlio di cui il padre è terminato senza fare wait. Viene adottato dal processo

`init` (PID=1) di cui farà la wait periodicamente (rilascio pcb).

Espansione Parametri	**Descrizione**
${*variabile*%%pattern}	Rimuove il più lungo suffisso.
${*variabile*%pattern}	Rimuove il più corto suffisso.
${*variabile*##pattern}	Rimuove il più lungo prefisso.
${*variabile*#pattern}	Rimuove il più corto prefisso.
${#*variabile*}	Restituisce il numero di byte della variabile.
${*variabile*/pattern/string}	Sostituisce la prima occorrenza da sinistra del parttern con la stringa indicata. Caratteri speciali da inserire dopo primo "/": • /: Fa sostituire tutte sotto strighe e non solo la prima. • #: Fa sostituzione solo se pattern è all'inizio. • %: Fa sostituzione solo se pattern è alla fine.
${*variabile*:ofset[:length]}	Restituisce la sotto stringa dal carattere i-esimo (ofset da 0). Opzionalmente si indica la lunghezza della sotto stringa da estrarre (length). Sia ofset che length sono valutati aritmeticamente.
${!*variabile**}	Da le variabili il cui nome inizia col prefisso indicato.

Danno una parte o una parte modificata (sostituiscono una parte in genere) del contenuto di una variabile. Il contenuto della variabile non viene modificato. Def. **pattern**: Stringa da matchare nel contenuto della variabile. Può contenere wildcard e variabili.

Gestione risorse	**Descrizione**
sudo apt-get update	Aggiorna la lista locale dei pakage.
sudo apt-get upgrade	Aggiorna i package.
sudo apt-get install *nomepkg*	Installa il package se è nella lista locale.
sudo apt-get install –reinstall *nomepkg*	Reinstalla il package sovrascrivendo l'attuale installazione.
sudo apt-get install -y *nomepkg*	Installa anche dipendenze necessarie.
sudo apt-get purge *nomepkg*	Disinstalla il pakage e i file di configurazione.
sudo apt-get autoremove	Disinstalla package inutilizzati.
aptitude search *package*	Cerca il package nella lista locale.
wget *risorsa*	Scarica file dalla rete.
wget -r *risorsa*	Scarica ricorsivamente le risorse seguendo i link.
wget -l n *risorsa*	Indica di quanto addentrarsi al massimo ricorsivamente.
wget -p *risorsa*	Fa scaricare tutte le risorse per visualizzazione corretta di una pagina.
wget -k *risorsa*	Fa puntare i link alle pagine locali.
wget -np *risorsa*	Non fa risalire alle directory superiori di quella specificata.
sudo fdisk	Crea le partizioni stabilendo intervallo blocchi da usare, il tipo di filesystem.
mkfs.*itpoFileSystem*	Crea il file system dentro una partizione.
lsmod	Elenca tutti i moduli attivi.
modinfo *nomeModulo*	Dice le informazioni sul modulo specificato.
sudo modprobe *nomeModulo*	Carica il modulo specificato.