Binbesc

Comandi

pwd mostra directory di lavoro corrente.
cd percorso_directory cambia la directory di lavoro corrente
mkdir percorso_directory crea una nuova directory nel percorso specificato
rmdir percorso_directory elimina la directory specificata, se è vuota
ls -alh percorso stampa informazioni su tutti i files contenuti nel percorso
rm percorso_file elimina il file specificato
echo sequenza di caratteri visualizza in output la sequenza di caratteri specificata
cat percorso_file visualizza in output il contenuto del file specificato
cut -b 1-3 o -c 3- (da utilizzare in pipe) da - a caratteri che mantieni b=byte c=char
env visualizza le variabili ed il loro valore
which nomefileeseguibile visualizza il percorso in cui si trova (solo se nella PATH) l'eseguibile
mv percorso p_nuovo sposta il file in una nuova posizione (se nella stessa dir, rinomina)
ps aux stampa informazioni sui processi in esecuzione
du percorso_directory visualizza l'occupazione del disco.
kill -9 pid_processo elimina processo avente identificativo pid_processo
killall nome_processo elimina tutti i processi con quel nome nome_processo
bg ripristina un job fermato e messo in sottofondo
fg porta il job più recente in primo piano
df mostra spazio libero dei filesystem montati
touch percorso_file crea il file specificato se non esiste, oppure ne aggiorna data.
more percorso_file mostra il file specificato un poco alla volta
head percorso_file mostra le prime 10 linee del file specificato -n *numRighe*
tail percorso_file mostra le ultime 10 linee del file specificato -n *numRighe*
man nomecomando è il manuale, fornisce informazioni sul comando specificato
find directory cercare dei files -maxdepth *num* -mindepth *num* -name **.h*
grep string nomefile cerca tra le righe di file quelle che contengono alcune parole
read nomevariabile legge input da standard input e lo inserisce nella variabile specificata
wc conta il numero di parole, caratteri o linee di un file wc -l "nomefile" (conta le linee)
source file.sh esegue lo script nella bash stessa e non figlia
true restituisce exit status 0 (vero)
false restituisce exit status 1 (non vero)
exit num imposta numero di exit status
uniq -c percorso_file stampa righe file univoche con il numero di eventuali ripetizioni, questo comando deve essere utilizzato in pipe con **sort**, altrimenti li prende solo se uno subito sotto l'altro.
export nomevar = valorevar rende nomevar una variabile d'ambiente per questa bash e le figlie
unset nomevar = distrugge la variabile nomevar
$# il numero di argomenti passati allo script
$0 il nome del processo in esecuzione, **$1** primo argomento, **$2** secondo argomento…
$* tutti gli argomenti passati a riga di comando concatenati e separati da spazi
$@ come $* ma se quotato gli argomenti vengono quotati separatamente
$? stampa exit status precedente
$! ID dell'ultimo processo in background
$$ ID del processo stesso

Specificatori

-d se esiste ed è una directory	-u legge riga da FD invece di stdin
-f se esiste ed è un file	-nt newer than
-r permesso di lettura	-ot older than
-l library	-eq equal
-c char	-ne not equal
-b byte	-lt lower than
-n numero oppure not null	-le lower equal
-z null	-gt greater than
-e se esiste	-ge greater equal

Manipolazioni di stringhe

%%**pattern*** suffisso più lungo (L'asterisco lo devi mettere
%**pattern*** suffisso più corto nell'ordine giusto
##***pattern** prefisso più lungo sennò non va!!!)
#***pattern** prefisso più corto

echo ${#VAR}: indica la lunghezza in byte
echo ${VAR/pattern/string}: sostituisce la variabile con string, partendo da pattern
es: ${VAR/b*a/x}: scrive x partendo dalla prima b che trova (alfabetagamma -> alfax)
echo ${VAR//pattern/string}: tutte le sottostringhe con quel pattern vengono sostituite
echo ${VAR/#pattern/string}: il pattern viene sostituito solo se si trova all'inizio della var
echo ${VAR/%pattern/string}: il pattern viene sostituito solo se si trova alla fine della var
echo ${VAR:offset}: stampa la variabile shiftata di offset (parte da 0)
echo ${VAR:offset:length}: stampa come sopra, da … a …
echo ${!Prefix*} restituisce un elenco con tutti i nomi delle variabili con prefisso Prefix

Manipolazione caratteri nei file

sed 's/togli/metti/' file.txt (sostituisce secondo parametro con terzo).
sed 's/a//' file.txt (rimuove tutti i caratteri a nel file)
sed 's/^.//' file.txt (rimuove primo carattere di ogni riga)
sed 's/.$//' file.txt (rimuove ultimo carattere di ogni riga)
sed -r 's/.{4}//' file.txt (rimuove i primi 4 caratteri ad inizio linea)

exec e read

exec {fd} < file.txt
read -N 4 -u < $fd STRINGA (legge primi 4 caratteri del file e li mette nella var STRINGA)

while read riga ; do
...
done < file.txt (legge dal file file.txt, $riga)

while read -u ${FD} RIGA ; [[$? == 0 || ${RIGA} != ""]] ; do
	echo "${RIGA}"
done 	(LEGGE TUTTO IL FILE ANCHE CON SPAZI ALLA FINE)

echo $var > file.txt 	(scrive su file, $var)

TIPS - Esempi

Espressioni condizionali su file (tramite parentesi quadre [...])
Espressioni matematiche con variabili d'ambiente (tramite doppie parentesi tonde ((...)))

for ((i=0;i<$#;i++)) ; do 	(equivalente a for var in $@)

done 	(foreach da 0 a $#)
for data in {0..9}; do
	qualcosa $data
done 	(foreach da 0 a 9)

offset: ${varDaOffsettare:numOffset:numLettereDaPrendereDiOffset}
	(se numOffset è negativo prende le lettere alla fine)
	es:
	OUT=`head file.txt -n 2`
	echo ${OUT: -2} 	ricordarsi di mettere lo spazio dopo i ":"

wget *link:* per scaricare file
tar -xvzf *nomeFile:* per unzippare

NumArg=$#
if [[$numArg -eq 1]]; then 	(se il numero di argomenti è …)

if [[! -e $directory]]; then 	(se esiste la directory)

1>&2 	(mette lo stdout dentro lo stderr)
echo "pene ingrassato" 1>&2

find '/usr/include' -mindepth 1 -maxdepth 2 -name '*.h'
	(stampa tutti i file con estensione .h nella directory e nella prima sottodirectory)
	(usare -type f per file e -type d per directory)

VAR="[13] qualcosa [10] qualcosa"
echo ${VAR**%%]***} (stampa: [13) suffisso più lungo dal carattere]
echo ${VAR**#*]**} (stampa: qualcosa [10] qualcosa) prefisso più corto dal carattere]

head file.txt -n 2 | cut -b -3 	(stampa primi 3 caratteri delle prime 2 righe del file)
tail file.txt -n 2 | cut -b -3 	(stampa primi 3 caratteri delle ultime 2 righe del file)
	find file.txt -exec tail -n 2 '{}' \; | cut -b -3 	(fa la stessa ma devi CERCARE il file)

OUT=`**grep** *21* file.txt` ; 	(mi raccomando i backtick)

if grep -Fxq “parola” test.txt (se il file contiene “parola” ritorna true)

echo ${OUT} (stampa tutta la riga dove è contenuta la stringa presa **char a char**)
se c’è scritto: **s 21 x** stampa: **s 21 x**
se c’è scritto: **luca21 n** stampa: **luca21 n**

echo `cat file.txt | **wc -l**` (se file.txt ha 8 righe, stampa: 8)

sort file.txt | **uniq -c** (stampa le righe **se non duplicate** di file.txt, con il numero delle ripetizioni)

if [[$((var **%** 2)) == 0]] (per calcolare il **modulo**)

Se fai la **find**, ls non funziona, devi utilizzare il for sulla $var della find

if [[! -z $VAR]] —> controlla se la stringa non è vuota

cut -b 1-3 —> prende i primi 3 caratteri e li stampa con l'echo

echo $RIS $(wc -l < “$RIS”) —> stampa il file e il numero di righe di quel file

Scrivere uno script bash cerca.sh che cerca tutti i file con estensione .h nella directory /usr/include/linux/netfilter/ cercando anche nelle sue sottodirectory.
Per ciascun file trovato controllare se in quel file è presente la stringa int
Stampare a video il nome (inteso come percorso assoluto) di quei file che NON contengono la stringa int :

```
for name in `find /usr/include/linux/netfilter -type f -name "*.h" -print` ; do
NUMLINES=`grep int ${name} | wc -l` ; if (( ${NUMLINES} == 0 )) ; then echo ${name} ;
fi ; done ;
```

script per trovare il file più recente in una cartella tra i “.h”

```
//inizializza solo il primo elemento che trova nella find
newest_file=""

for elem in ‘find/usr/include -type f -name “*-h” ’;
do
        newest_file=$elem
        break
done

for file in $(find /usr/include/ -type f -name "*.h")
do
   if [[ $file -nt $newest_file ]]
   then
      newest_file=$file
   fi
done

echo $newest_file
```

C - Thread

Ciclo di vita del Thread

Per **creare i thread**:
int pthread_create (pthread_t * thread, pthread_attr_t *attr, void* (*start_routine)(void *), void * arg);
se non riesce a creare il thread PrintERROR_andExit(rc, “errore”);

Per **terminare i thread**:
void pthread_exit (void *retval);

Per **restituire identificatore thread:**
pthread_t pthread_self (void);

Per **restituire 1** se i due identificatori di **pthread sono uguali:**
int pthread_equal (pthread_t pthread1, pthread_t pthread2);

Per **conoscere il risultato del thread terminato:**
pthread_join (pthread_t * thread, void** returnedVar)

Si deve poi eseguire la **free()** del puntatore del thread.

Thread **detached:**
pthread_attr_init(&attr);
pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_DETACHED);
pthread_attr_destroy(&attr);

Mutex

Per **creare mutex statico:**
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

Per **creare mutex dinamico:**
pthread_mutex_t mutex;
pthread_mutex_init (&mutex, &attribute);
se non riesce a creare il mutex PrintERROR_andExit(rc, “errore”);

Per **bloccare e sbloccare il mutex:**
pthread_mutex_lock(&mutex);
pthread_mutex_unlock(&mutex);

Per **distruggere il mutex:**
pthread_mutex_destroy(&mutex);

Condition variabols

Per **creare condition statica:**
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;

Per **creare condition dinamica:**
pthread_cond_t cond ;
pthread_cond_init(&cond , &attribute);

Per **avviare la wait:**
pthread_cond_wait(&cond, &mutex);

Per **cambiare la condition** e svegliare un altro thread bloccato nella wait:
pthread_cond_signal(&cond);

Per **cambiare la condition** e svegliare tutti i thread bloccato nella wait:
pthread_cond_broadcast(&cond);

Per **distruggere la condition:**
pthread_cond_destroy(&cond);

DBGpthread

Per stampare automaticamente messaggi di errori si utilizzano i comandi con DBG. Deve essere implementata la funzione DBGpthread.h

- DBGpthread_mutex_lock(&mutex, Label);
- DBGpthread_mutex_unlock(&mutex, Label);
- DBGpthread_cond_wait(&cond, &mutex, Label);
- DBGpthread_cond_signal(&cond, Label);
- DBGpthread_cond_broadcast(&cond, Label);
- DBGpthread_mutex_init(&mutex, NULL, Label);
- DBGpthread_mutex_destroy(&mutex, Label);
- DBGpthread_cond_init(&cond, NULL, Label);
- DBGpthread_cond_destroy(&cond, Label);
- DBGsleep(NumSeconds, Label);

Altrimenti si può utilizzare il print error standard manuale:
if (rc) PrintERROR_andExit(rc,"Messaggio di errore");

MACRO

#define FOO(x) do { foo(x); bar(x); } while (0)

Makefile

Struttura

```
CFLAGS=-ansi -Wpedantic
TARGET=main.exe
OBJECTS=main.o funzioni.o

all :           ${TARGET}
riga vuota
main.exe :      main.o  funzioni.o
TAB             gcc ${CFLAGS} -o ${TARGET}  main.o funzioni.o
riga vuota
main.o :        main.c  funzioni.h strutture.h
TAB             gcc -c  ${CFLAGS}  main.c
riga vuota
funzioni.o :    funzioni.c  strutture.h
TAB             gcc -c  ${CFLAGS}  funzioni.c
riga vuota
.PHONY:         clean
riga vuota
clean:
TAB             -rm   ${TARGET}  ${OBJECTS}   *~   core
NEWLINE
```

Esempio

```
CFLAGSCONSTRERROR=-ansi -Wpedantic -Wall -D_REENTRANT -D_THREAD_SAFE
-D_POSIX_C_SOURCE=200112L
CFLAGS=-ansi -Wpedantic -Wall -D_REENTRANT -D_THREAD_SAFE
LIBRARIES=-lpthread
DBGPTHREADFUNCTION_SOURCE_DIR=./

all:        programma.exe

programma.exe:  programma.o DBGpthread.o
        gcc -o programma.exe programma.o DBGpthread.o ${LIBRARIES}

programma.o: programma.c ${DBGPTHREADFUNCTION_SOURCE_DIR}DBGpthread.h
${DBGPTHREADFUNCTION_SOURCE_DIR}printerror.h
        gcc -c ${CFLAGS} -I${DBGPTHREADFUNCTION_SOURCE_DIR} programma.c

DBGpthread.o: ${DBGPTHREADFUNCTION_SOURCE_DIR}DBGpthread.c
${DBGPTHREADFUNCTION_SOURCE_DIR}printerror.h
        gcc ${CFLAGSCONSTRERROR} -c ${DBGPTHREADFUNCTION_SOURCE_DIR}DBGpthread.c
-I${DBGPTHREADFUNCTION_SOURCE_DIR}

.PHONY:         clean

clean:
        -rm -f DBGpthread.o  programma.o  programma.exe
```