Laboratorio su sistemi Linux
Interprete dei comandi
Shell scripting

Prima lezione in laboratorio

NOTA BENE:

A questo punto abbiamo gia' visto ed usato i comandi:

pwd cd ls echo (e forse) env which ps touch

Usare il comando **man** nomecomando per ottenere informazioni sull'uso di uno specifico comando di nome nomecomando.

NB: Per alcuni comandi (set) ed alcune funzioni posix compliant (quelle dei pthread), occorre installare le informazioni per il man installando il pacchetto manpages-posix-dev

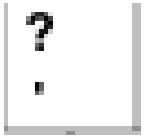
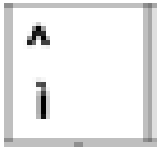
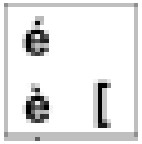
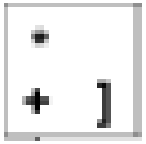
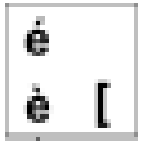
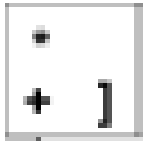
Iniziare con Ubuntu (locale o personalizzata)

Una volta che avrete fatto il boot della macchina virtuale Linux e sarete loggati con le credenziali "studente" e password "studente", potete utilizzare tre applicativi:

- **Navigare**: click in basso a sinistra sul pulsante delle applicazioni, guardare il menù verticale delle applicazioni, click su voce menù Internet, cliccare sull'icoma "Firefox Web Browser" per il **browser web** (o anche chrome, se c'è) .
  - Nella barra degli indirizzi del web browser digitare l'indirizzo
    - http://www.cs.unibo.it/~ghini/
    - Navigare fino ad arrivare alla pagina di Sistemi Operativi
    - Aprire il file con la dispensa relativa alle lezioni di laboratorio
- **Terminale bash**: click in basso a sinistra sul pulsante delle applicazioni, guardare il menù verticale delle applicazioni, click su voce lSystem Tools", click su QTerminalper lanciare il **terminale** interattivo **bash**
  - Farete qui i vostri esercizi.
- Editing visuale: potete utilizzare un **editor grafico** cercando l'applicazione **gedit** con lo strumento di ricerca (icona in alto a sinistra) per aprire la finestra dell'editor visuale.
  - Lo userete per scrivere i vostri script, salvandoli nella vostra home directory.

Combinazioni di tasti per caratteri speciali

- In una tastiera con un layout italiano, le seguenti sono (probabilmente) le combinazioni di tasti da digitare per ottenere alcuni caratteri speciali utili nella programmazione con script bash:
- NB: in un portatile potrebbero essere necessarie combinazioni diverse.

backtick	**`**	AltGr + 
tilde	**~**	AltGr + 
parentesi quadra aperta	**[**	AltGr + 
parentesi quadra chiusa	**]**	AltGr + 
parentesi graffa aperta	**{**	Shift + AltGr + 
parentesi graffa chiusa	**}**	Shift + AltGr + 

Sequenze ASCII per caratteri speciali su Tastierino Numerico o Tastierino Numerico Emulato

- In una tastiera con un layout italiano, ed il tastierino numerico (i tasti con i numeri sul lato a destra della tastiera) le seguenti sequenze di tasti permettono di ottenere alcuni caratteri speciali utili nella programmazione con script bash:
- In un portatile, il tastierino numerico viene emulato premendo il tasto Fn

		su Tastierino	**senza Tastierino**
backtick	**`**	Alt +129	Alt + Fn +129
tilde	**~**	Alt +126	Alt + Fn + 126
parentesi quadra aperta	**[**	Alt +90	Alt + Fn + 90
parentesi quadra chiusa	**]**	Alt +92	Alt + Fn + 92
parentesi graffa aperta	**{**	Alt +123	Alt + Fn + 123
parentesi graffa chiusa	**}**	Alt +125	Alt + Fn + 125

Iniziare con la shell

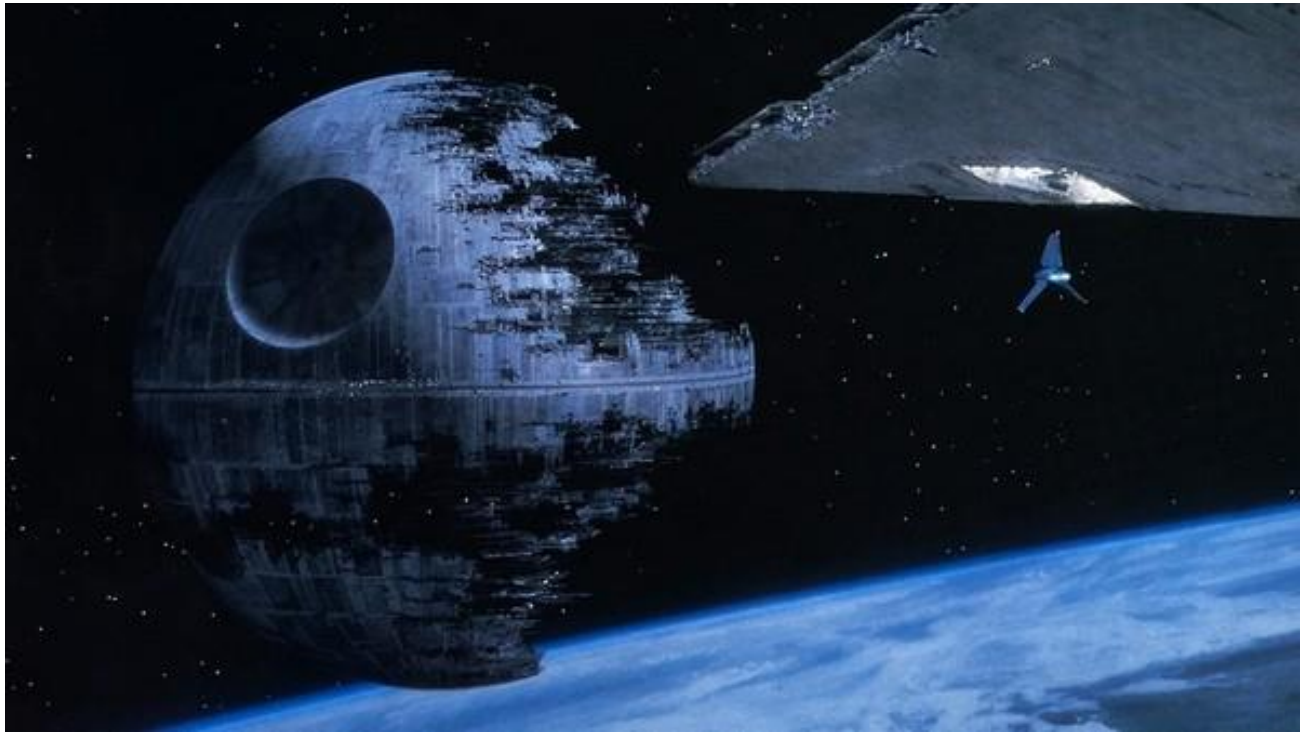
- cliccare sul menù di sistema (in basso a sinistra) , poi cliccare sul sottomenù "System Tools" e cliccare su LXTerminal per aprire la shell di comandi all'interno di una nuova finestra grafica.
- Cliccare sulla parte interna della finestra della shell interattiva dopo che questa e' apparsa per spostare il focus sulla finestra stessa.
- Lanciare ripetutamente il comando man per capire come si usa il comando cd e come si usa il comando ls. Leggere con attenzione .
- Usare il comando pwd per verificare la directory corrente
- Usare il comando ls per guardare i nomi e le proprietà dei file contenuti nella directory corrente.
- Usare il comando ls per verificare la presenza di eventuali file nascosti.
- Usare il comando echo per visualizzare il contenuto della PATH di sistema. Come? Arrangiatevi! Lo sapete già fare.
- Usare il comando which per verificare l'esistenza e il percorso in cui si trova l'eseguibile find.
- Usare il comando ls per visualizzare il contenuto della directory in cui e' collocato l'eseguibile find.

Familiarizzarsi con la shell (1)

- Usare il comando which per verificare l'esistenza e il percorso in cui si trova il compilatore C **gcc**
- Usare il comando ls, **specificando un percorso relativo,** per visualizzare il contenuto della directory in cui e' collocato l'eseguibile gcc.
- Lanciare il comando gcc senza argomenti a riga di comando.
- Lanciare il comando df -h per visualizzare quali filesystem sono attualmente montati e quanto spazio occupano.
- Lanciare il comando cd senza argomenti per ritornare alla propria home directory
- Verificare dove ci si trova.
- Cercare sulla tastiera i caratteri:

due punti	:	punto e virgola	;
doppio apice	"		
apice semplice	'	backtick	`
tilde	~	pipe	|
punto esclamativo	!	asterisco	*
parentesi graffa aperta	{	parentesi graffa chiusa	}
parentesi quadra aperta	[	parentesi quadra chiusa	]

Lezione 2 in laboratorio



- bash exercises: la morte nera in costruzione ...

Familiarizzarsi con la shell (2)

1. Visualizzare a video il contenuto delle variabili USER HOME DISPLAY
2. Verificare se le variabili USER HOME DISPLAY sono di ambiente o locali.
3. Scrivere un comando che crea una nuova variabile di nome PIPPO la quale contiene la concatenazione dei contenuti delle variabili USER HOME DISPLAY
4. Scrivere uno script bash, di nome crea_var.sh, che crea una nuova variabile di nome PIPPO la quale contiene la concatenazione dei contenuti delle variabili USER HOME DISPLAY
5. Dall'interno di una shell bash, eseguire lo script crea_var.sh in un modo tale che la variabile PIPPO sia presente (col nuovo contenuto) dentro la bash chiamante dopo la fine dell'esecuzione dello script crea_var.sh
6. Spostarsi in radice del filesystem e poi stampare a video la propria home directory.
7. Creare nella propria home directory una sottodirectory A1 che contiene una sottodirectory B2 che contiene una sottodirectory C3 che contiene tre file 1.txt 2.txt e 3.txt. Poi usate il comando move (mv) **per spostare** i 3 file nella directory B2. Poi usate il comando copy (cp) per mettere una copia dei tre file nella directory A1.
8. **Provare** a vedere il contenuto di tutti i file nascosti presenti nella propria home directory, usando il comando cat. Notare il problema dato dal fatto che .* viene espanso anche con ..
9. Usare la funzionalità di completamento (tasto TAB) per scrivere velocemente l'argomento del comando ls /usr/include/linux/netfilter/nf_nat.h

Soluzioni di 'Familiarizzarsi con la shell (2)'

1. echo ${USER} ${HOME} ${DISPLAY}
2. env | more e leggere se ci sono USER HOME e DISPLAY

3. PIPPO=${USER}${HOME}${DISPLAY}
4. crea_var.sh contiene PIPPO=${USER}${HOME}${DISPLAY}
5. source ./crea_var.sh

6. cd / ; echo ~

7. cd ; mkdir A1 ; mkdir A1/B2 ; mkdir A1/B2/C3 ;
   for num in 1 2 3 ; do touch A1/B2/C3/${num}.txt ; done ;
   mv A1/B2/C3/?.txt A1/B2/ ;
   cp A1/B2/?.txt A1/
8. # che casino che provoca
   cat .*

9. ls /u<TAB>in<TAB>linu<TAB>netf<TAB>/<TAB>n<TAB>_n<TAB>

Familiarizzarsi con la shell (3)

10. Lanciare il comando history e scegliere uno dei comandi elencati da history. Utilizzare le funzioni di history per rilanciare quel comando.
11. Usare il comando set per disabilitare la memorizzazione di history. Poi lanciare un comando qualsiasi e poi lanciare history e verificare che quel comando non è stato memorizzato. Infine, usare il comando set per riabilitare la memorizzazione di history.
12. Usare il comando set per abilitare la creazione di variabili d'ambiente e verificare se funziona (inventarsi un modo per testare se le variabili sono create come var d'ambiente oppure no)
13. Inserire un comando echo, con un messaggio diverso, nei file .profile e .bashrc Poi lanciare una shell interattiva NON di login e verificare quale dei due file viene eseguito. Poi lanciare una shell NON interattiva e verificare che non viene eseguito nessuno dei due file. Poi riportare i file .profile e .bashrc come erano all'inizio.
14. Usare i metacaratteri per visualizzare con ls le proprietà dei file contenuti nella directory /usr/lib/ che hanno un nome che contiene la stringa plu
15. Usare il comando man per studiare le opzioni del comando ls.
16. Usare ls e le sue opzioni per visualizzare tutti i file nella directory /usr/include/ e nelle sue sottodirectory.
17. Usare ls e le sue opzioni per visualizzare le informazioni di una directory /usr/include/ senza visualizzare tutti i file nella directory

Soluzioni di 'Familiarizzarsi con la shell (3)'

10. ! numerodelcomando
11. set +o history (disabilita) set -o history (abilita)
12. set -a
13. modificare i file etc ect
14. ls -alh /usr/lib/*client*
15. man ls
16. ls -alhR /usr/include/
17. ls -dlh /usr/include/

Familiarizzarsi con la shell (4)

18. Creare una sottodirectory BUTTAMI e crearci dentro due file AbC.txt e ABC.txt
19. Eliminare il file appena creato ABC.txt
20. Eliminare, con un unico comando, la directory BUTTAMI e tutti i files in essa contenuti.
21. Visualizzare, mediante ls e metacaratteri, tutti i files della directory /usr/lib/ il cui nome contiene
    - o un carattere numerico compreso tra 1 e 3
    - oppure un carattere letterale compreso tra c ed m.

    e che termina con l'estensione .0

Soluzioni di 'Familiarizzarsi con la shell (4)'

18. mkdir BUTTAMI ; touch BUTTAMI/AbC.txt ; touch BUTTAMI/ABC.txt

19. rm BUTTAMI/ABC.txt

20. rm -rf BUTTAMI/

21. ls /usr/lib/*[1-3c-m]*.0

    per essere precisi, potrebbe esistere una directory che ha il nome che corrisponde ai nomi di file cercati,

    per evitare di vedere listati anche i files contenuti in quelle directory,

    si può aggiungere ad ls l'opzione -d

    che fa vedere solo la directory e non i suoi files.

    soluzione più corretta:

    ls -d /usr/lib/*[1-3c-m]*.0

Lezione 3 in laboratorio

dura è la strada della bash

NOTA INTRODUTTIVA
Gli esercizi che seguono
cercate di risolverli nel modo più semplice,
usando solo quello che abbiamo visto a lezione

Familiarizzarsi con la shell (5) argomenti a riga di comando

22. Creare tre file **con spazio.txt** **senzaspazio1.txt** e **senzaspazio2.tx**t (notare lo spazio presente tra le lettere n ed s del nome del file con spazio.txt).
23. scrivere due script lancia.sh e stampaargs.sh.

    Lo script lancia.sh determina il numero degli argomenti che gli sono stati passati a riga di comando ed esegue lo script stampaargs.sh passandogli come primo argomento il numero di argomenti trovato seguito da tutti gli argomenti che erano stati passati a lancia.sh stesso.

    Lo script stampaargs.sh stampa a video gli argomenti che ha ricevuto a riga di comando, ciascuno in una diversa riga di output.

    Lanciare lo script lancia.sh passandogli i nomi di tutti i file il cui nome contiene la stringa spazio, e verificare se lo script stampaargs.sh stampa correttamente a video il numero di argomenti ed il nome dei files passati a lancia.sh
24. Ripetere l'esercizio precedente, creando analogamente due file lancia2.sh e stampaargs2.sh. Stavolta però il file lancia2.sh non passa gli argomenti a stampaargs2.sh nella riga di comandi, bensì in una variabile d'ambiente NOMIFILES appositamente creata per lo script stampaargs2.sh.

    Verificare SE E' POSSIBILE identificare bene i confini dei nomi di files (risposta NO).
25. Usare il comando rm per eliminare i 3 files che hanno la stringa spazio nel nome

Soluzioni di 'Familiarizzarsi con la shell (5)' argomenti a riga di comando

22. touch "**con spazio.txt**" **senzaspazio1.txt** **senzaspazio2.txt**

23. file lancia.sh

```
#!/bin/bash
./stampaargs.sh  $#  "$@"
```

file stampaargs.sh

```
#!/bin/bash
for name in "$@" ; do  echo  $name ; done
```

eseguire lancia.sh così:

./lancia.sh *spazio*

24. file lancia2.sh

```
#!/bin/bash
 NOMIFILES="$# $@" ./stampaargs2.sh
```

file stampaargs2.sh

```
#!/bin/bash
 for N in ${NOMIFILES} ; do echo $N ;  done
```

eseguire lancia.sh così: ./lancia.sh *spazio*

purtroppo non funziona, Si perdono i confini degli argomenti. Servirebbero variabili di tipo **Array associativo** (come la $@) che non facciamo.

25. rm "con spazio.txt" senzaspazio1.txt senzaspazio2.txt

Familiarizzarsi con la shell (6) quoting

26) Scrivere uno script che esegue i seguenti compiti:

Creare una directory BUTTAMI, dentro questa creare dei file che si chiamano
* ** *** ;;

Fare un listing di questi file e poi, per ciascuno dei file nella directory aggiungere un nuovo file con stesso nome con in più l'estensione .txt.

Copiare in questa directory tutta la directory /usr/include/ (e i suoi file e sottodirectory ricorsivamente).

Stampare a video tutte le sottodirectory (non i file) della vostra directory BUTTAMI comprese le sottodirectory delle sottodirectory e così via ricorsivamente.

Eliminare la directory "include" nella vostra directory BUTTAMI (eliminare anche tutti i file e sottodirectory di include, ATTENTI A NON ELIMINARE l'ORIGINALE /usr/include

27) Rifare tutto l'esercizio 26 qui sopra dove però il "per ciascuno dei file nella directory" viene realizzato usando anche una command substitution.

NOTARE CHE IN QUESTO CASO NON SI RIESCE AD ESEGUIRE UN SOLO COMANDO touch per ciascuno dei file esistenti. Il problema dipende dalla presenza degli asterischi nel nome. **MORALE: QUANDO POSSIBILE USARE I METACARATTERI PER GENERARE I NOMI DI FILES.**

Soluzioni di Familiarizzarsi con la shell (6) quoting

26)

```
#!/bin/bash
mkdir BUTTAMI
touch BUTTAMI/"*"   BUTTAMI/"**"   \
     BUTTAMI/"***" BUTTAMI/";;“

ls BUTTAMI/*

for  i  in   BUTTAMI/*  ; do
     touch "$i.txt"  ;
done

cp -R /usr/include ./BUTTAMI/
find ./BUTTAMI -type d

rm   -rf   BUTTAMI/include
```

27) PROBLEMATICO USANDO COMMAND SUBSTITUTION

```
#!/bin/bash
mkdir BUTTAMI
touch BUTTAMI/"*"   BUTTAMI/"**"   \
     BUTTAMI/"***" BUTTAMI/";;"

ls BUTTAMI/*

for  i  in   `ls BUTTAMI/*` ; do
     touch "${i}.txt"  ;
done

cp -R /usr/include ./BUTTAMI/
find ./BUTTAMI -type d

rm -rf BUTTAMI/include
```

Script con argomenti a riga di comando

Per realizzare iterazioni e confronti, prendere spunto dagli esempi con while for e if che abbiamo fatto a lezione, quindi riguardatevi le slide.

4. Scrivere un piccolo script **chiama_fattoriale.sh** il quale invoca lo script **fattoriale.sh** passandogli l'argomento 5 e cattura dallo stdout il risultato prodotto dal fattoriale, mettendolo in una variabile denominata RISULTATO. Scrivere poi lo script **fattoriale.sh** che prende in input un argomento intero positivo e calcola il fattoriale di quel numero. Lo script fattoriale.sh scrive il risultato sullo stdout.
5. **COMPLICATO** Partendo dall'esercizio precedente, scrivere uno script **chiama_fattoriale1.sh** il quale crea una variabile condivisa RIS, **invoca in modo opportuno** lo script **fattoriale1.sh** passandogli l'argomento 5, ottiene il risultato che viene messo dallo script fattoriale1.sh nella variabile RIS e ne stampa a video il contenuto. Scrivere uno script **fattoriale1.sh** che chiami sé stesso ricorsivamente per calcolare il fattoriale del numero passato come argomento. A differenza del caso precedente, il risultato di ogni invocazione deve essere scritto in una variabile condivisa (in entrambe le direzioni) tra chiamante e chiamato di nome RIS.
6. Scrivere uno script **script1.sh** che prende a riga di comando un numero variabile di argomenti. Lo script visualizza sullo standard output tutti gli argomenti e poi invoca un altro script **script2.sh** passandogli tutti gli argomenti ricevuti. Anche il secondo script visualizza gli argomenti, ma in ordine inverso, poi termina. Eseguire lo script1.sh passandogli come argomenti a b "c d" e f

Soluzioni esercizi slide precedente

chiama fattoriale.sh

```
#!/bin/bash
RISULTATO=`./fattoriale.sh 5`
echo "${RISULTATO}"
```

fattoriale.sh

```
#!/bin/bash
NUM=1
PRODOTTO=1
while (( ${NUM} <= $1 )) ;  do
  ((  PRODOTTO=${PRODOTTO}*${NUM} ))
  ((  NUM=${NUM}+1  ))
done
echo "${PRODOTTO}"
```

chiama fattoriale1.sh

```
#!/bin/bash
RIS=1
source  ./fattoriale1.sh 5
echo "RIS FINALE= ${RIS}"
```

fattoriale1.sh

```
#!/bin/bash
NUM=$1
if (( ${NUM} > 1 ))  ;  then
        ((  RIS=${RIS}*${NUM}  ))
        ((  NUM=${NUM}-1  ))
        source ./fattoriale1.sh "${NUM}"
fi
```

script1.sh

```
#!/bin/bash
NUM=1
while (( ${NUM} <= $# )) ; do
        echo "arg ${NUM} is ${!NUM} "
        ((NUM=${NUM}+1))
done
./script2.sh "$@"
```

script2.sh

```
#!/bin/bash
NUM=$#
while (( ${NUM} > "0" )) ; do
        echo "arg ${NUM} is ${!NUM} "
        ((NUM=${NUM}-1))
done
```

Familiarizzarsi con i comandi di shell (6)

7. Scoprire quale e' il flag di echo che permette di stampare a video caratteri speciali consentendo di passarli al comando echo stesso mediante le sequenze di escape come in C, quali \n \b \t (abilita interpretazione di backslash escapes).
8. Scoprire quale e' il flag di echo che permette di stampare a video la stringa passata come argomento SENZA poi andare a capo.
9. Visualizzare l'elenco dei file della directory corrente ad esclusione dei file il cui nome comincia per .
10. Scrivere uno script che usa anche il comando for per visualizzare, per ciascun file della directory corrente che non inizia per . , una coppia di righe in cui:
    - la prima riga contiene la stringa "file is " seguita dal nome del file
    - la seconda riga fa il listing delle informazioni sul file. Se il file è una directory, il listing deve contenere solo il nome della directory, non quello dei suoi files.
11. Ripetere la prova precedente, stavolta tutto il comando for do done su una unica riga, aggiungendo le necessarie modifiche per ottenere lo stesso risultato di prima.
12. Modificare lo script per verificare se il comando ls è andato a buon fine e, in caso contrario, stampare a video un avviso "ls produce errore"
13. Creare un file il cui nome contiene uno spazio bianco, ad esempio alfa beta.txt
14. Ora ripetere la prova al punto 9 e verificare se funziona. **DOVREBBE ACCADERE UN ERRORE LEGATO ALLA PRESENZA DELLO SPAZIO BIANCO NEL NOME. NON CERCATE DI CORREGGERLO. OCCORREREBBE MODIFICARE IFS**

Familiarizzarsi con i comandi di shell (6)'

Soluzioni di alcuni esercizi slide precedente

7) echo -e "\tciao\n"

8) echo -n "\tciao\n"

9) ls

10) SOLUZIONE PERFETTA

```
for name in * ; do
        echo "file is ${name}"
        ls -ld ${name}
done
```

11) SOLUZIONE CHE INVECE PROVOCA UN PROBLEMA CON SPAZI BIANCHI NEI NOMI

```
for name in `ls -1` ; do
        echo "file is ${name}"
        ls -ld ${name}
done
```

12) for name in `ls -1` ; do echo "file is ${name}" **;** ls -ld ${name} ; done

13) for name in `ls -1` ; do echo "file is ${name}" **;** ls -ld ${name} ; if (($? != 0)) ; then echo "ls produce errore" ; fi ; done

LA SOLUZIONE PROVOCA UN PROBLEMA CON NOMI DI FILE CHE CONTENGONO SPAZI BIANCHI. NON TENTATE DI RISOLVERLO, OCCORREREBBE SETTARE IFS.

Familiarizzarsi con i comandi di shell (7)

10. Scrivere uno script che usa anche il comando for per visualizzare, per ciascun file della directory corrente che non inizia per . , una coppia di righe in cui:
    4. la prima riga contiene la stringa "file is " seguita dal nome del file
    5. la seconda riga fa il listing delle informazioni sul file.
11. Creare un file il cui nome contiene uno spazio bianco, ad esempio alfa beta.txt
12. Ora ripetere la prova al punto 8 e verificare se tutto funziona a dovere.
13. Correggere eventuali errori, si suggerisce di guardare il valore della variabile IFS e anche il man di ls per vedere che cosa fa il flag –1

SOLUZIONE CORRETTA

```
OLDIFS=${IFS}
IFS=$'\t\n'
for name in `ls -1` ; do      echo "file is ${name}" ; ls -ld ${name} ; done
IFS=${OLDIFS}
```

Familiarizzarsi con i comandi di shell (8)

15. Scrivere uno script bash **triplette.sh** che stampa a video tutte le triplette di forma (X;Y;Z) dove X Y e Z sono i nomi dei file, non nascosti, nella directory corrente.
16. Scrivere uno script bash **argomenti.sh** che prende in input un numero qualsiasi di argomenti e stampa a video una stringa formata dalla concatenazione degli argomenti di indice pari seguiti dagli argomenti di indice dispari.
17. Nello script precedente provare a passare un ; come argomento e correggere eventuali errori.
18. Scrivere uno script bash **sommaquadrati.sh** che prende in input un numero qualsiasi di argomenti interi positivi e stampa a video il numero intero dato dalla somma dei quadrati dei singoli argomenti diminuito della somma degli indici degli argomenti.
19. scrivere quattro script bash **main.sh** **definisci.sh** **usa.sh** ed **elimina.sh** che svolgono queste operazioni: definisci.sh crea la variabile **d'ambiente** VAR e ne setta il contenuto a "INIZIO", usa.sh stampa a video il contenuto della variabile VAR, elimina.sh elimina dall'ambiente di esecuzione la variabile VAR, mentre main.sh chiama opportunamente i tre script definisci.sh usa.sh ed elimina.sh in modo da creare nel proprio ambiente di esecuzione la stringa VAR, di stamparne a video il contenuto e di eliminare la variabile dal proprio ambiente di esecuzione.

    Infine main.sh chiama ancora opportunamente lo script usa.sh per verificare se la variabile VAR è stata effettivamente eliminata.

Familiarizzarsi con i comandi di shell (8)
Soluzioni di slide precedente

15) triplette.sh

```
#!/bin/bash
NOMIFILES=`ls`
for nome1 in ${NOMIFILES} ; do
    for nome2 in ${NOMIFILES} ; do
        for nome3 in ${NOMIFILES} ; do
#           echo "(${nome1};${nome2};${nome3})" ;
            echo \(${nome1}\;${nome2}\;${nome3}\) ;
        done
    done
done
```

16) argomenti.sh

```
#!/bin/bash
STR=""
NUM=0
for (( NUM=2; ${NUM}<=$#; NUM=${NUM}+2 )) ; do
      STR="${STR}${!NUM}";
done
for (( NUM=1; ${NUM}<=$#; NUM=${NUM}+2 )) ; do
      STR="${STR}${!NUM}";
done
echo ${STR}
```

18) sommaquadrati.sh

```
#!/bin/bash
RIS=0
for (( NUM=1; ${NUM}<=$#; NUM=${NUM}+1 )) ; do
      ((RIS=${RIS}+${!NUM}*${!NUM}));
done
for (( NUM=1; ${NUM}<=$#; NUM=${NUM}+1 )) ; do
      ((RIS=${RIS}-${NUM}));
done
echo ${RIS}
```

19) main.sh

```
#!/bin/bash
source ./definisci.sh
./usa.sh
source ./elimina.sh
./usa.sh
```

definisci.sh

```
export VAR="INIZIO"
```

usa.sh

```
echo ${VAR}
```

elimina.sh

```
unset VAR
```

Lezione 4 in laboratorio

- bash exercises:

attenzione .....

leggi bene il testo degli esercizi !

Un po' di Espressioni condizionali

USARE LE ESPRESSIONI CONDIZIONALI nella forma **[[ ]]** per risolvere i due seguenti esercizi

1. Scrivere uno script **cercadir.sh** che cerca tra tutti i file e directory contenuti nella directory /usr/include (non nelle sue sottodirectory) e stampa in output il percorso assoluto dei file che verificano tutte le seguenti proprietà: a) sono delle directory, b) hanno il permesso di lettura da parte dello user attuale, c) la data di ultima modifica del file è strettamente più recente di quella del file /usr/include/stdio.h
2. Scrivere uno script **cercafile.sh** che, per ciascuna lettera che sta tra **c** e **g** cerca i file (o directory) che stanno nella directory /usr/include (non nelle sottodirectory) e che hanno quella lettera come secondo carattere del nome del file, e che verificano una delle due seguenti proprietà: la lunghezza del percorso assoluto del file è minore di 18 OPPURE è maggiore di 23. Di questi file stampa in output il percorso assoluto.

3) USARE poi LE ESPRESSIONI CONDIZIONALI nella forma **[ ]** per risolvere l'esercizio numero 1 qui sopra descritto

4) USARE poi LE ESPRESSIONI CONDIZIONALI nella forma test per risolvere l'esercizio numero 1 qui sopra descritto

Un po' di Espressioni condizionali Soluzioni 1 e 2

1) **cercadir.sh**

```
#!/bin/bash
for name in /usr/include/*  ; do
   if [[   -d ${name}  && -r  ${name} &&  \
           ${name} -nt /usr/include/stdio.h ]] ;
   then
           echo /usr/include/${name}
   fi
done
```

NB: a rigore, dentro le [[ ]] i caratteri **** prima delle andate a capo NON servono, invece servono dentro le [ ] e anche con la forma test

2) **cercafile.sh**

```
#!/bin/bash
for car in {c..g} ; do
   for name in /usr/include/?${car}* ; do
           if [[  -e ${name} && ( ${#name} -lt 18  ||  ${#name} -gt  23 )  ]]  ; then
                   echo ${name}
           fi
   done
done
```

Un po' di Espressioni condizionali altra soluzione 1

1)
```
#!/bin/bash
  for name in `ls /usr/include/` ; do
    if [[   -d /usr/include/${name} &&        \
            -r /usr/include/${name} &&        \
            /usr/include/${name} -nt /usr/include/stdio.h ]] ;
    then
            echo /usr/include/${name}
    fi
  done
```

Un po' di Espressioni condizionali Soluzioni 3 e 4

3)
```
#!/bin/bash
    for name in `ls /usr/include/` ; do
       if  [   -d /usr/include/${name} -a          \
               -r /usr/include/${name} -a          \
               /usr/include/${name} -nt /usr/include/stdio.h  ]  ;
       then
               echo ${name}
       fi
    done
```

4)
```
#!/bin/bash
    for name in `ls /usr/include/` ; do
       if    test      -d /usr/include/${name} -a            \
                       -r /usr/include/${name} -a            \
                       /usr/include/${name} -nt /usr/include/stdio.h    ;
       then
               echo ${name}
       fi
    done
```

Familiarizzarsi con i comandi di shell (9)

20. Stampare a video le sole righe del file /usr/include/stdio.h che contengono almeno un asterisco *
    Suggerimento1: utilizzare il comando grep ma senza usare l'operatore |
    Suggerimento2: Guardare il man di grep per capire come usare grep in modo opportuno.
21. Stampare a video le sole righe del file /usr/include/stdio.h che NON contengono alcun asterisco *
    Suggerimento2: Guardare il man di grep per capire come usare grep in modo opportuno.
22. Scrivere uno script bash **leggere.sh** che legge le righe del file /usr/include/stdio.h e stampa a video la sola terza parola di ciascuna riga, o niente se la terza parola non esiste
23. Scrivere uno script bash **leggerecaratteri.sh** che legge uno per uno i caratteri del file /usr/include/stdio.h e stampa a video il numero dei caratteri letti dal file
24. Capire cosa fa il comando wc -c /usr/include/stdio.h
25. Scaricare il file di testo http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/miofileNoNL.txt che ha anche delle righe vuote **e che termina con una riga che non ha il carattere di andata a capo**. Scrivere poi uno script **leggitutto.sh** che legga riga per riga quel file e stampi a video ciascuna riga letta, compresa l'ultima.

Familiarizzarsi con i comandi di shell (9) Soluzioni

20) grep '*' /usr/include/stdio.h

21) grep -v '*' /usr/include/stdio.h

22) leggere.sh

```
#!/bin/bash
exec {FD}< /usr/include/stdio.h
if (( $? == 0 )) ; then
        while read -u ${FD} A B C D ; do
                echo "${C}"
        done
        exec {FD}>&-
fi
```

23) leggerecaratteri.sh

```
#!/bin/bash
exec {FD}< /usr/include/stdio.h
if (( $? == 0 )) ; then
        NUM=0
        while read -u ${FD} -N 1 -r A ; do
                ((NUM=${NUM}+1))
        done
        exec {FD}>&-
        echo ${NUM}
fi

#  -N 1 serve per leggere 1 carattere per volta
#  -r  serve per NON interpretare i  \ incontrati nel file
#       come  inizio di una sequenza di escape
```

25) leggeretutto.sh

```
#!/bin/bash
exec {FD}<   ./miofileNoNL.txt
if (( $? == 0 )) ; then
        while  read  -u  ${FD}  RIGA ;  [[  $? == 0  ||   ${RIGA} != "" ]]  ; do
                echo "${RIGA}"
        done
        exec {FD}>&-
fi
```

ATTENZIONE, METTERE SPAZI PRIMA E DOPO **!=**

ESEMPIO FILE http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/miofileNoNL.txt

SOLUZIONE http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/leggitutto.sh

DETTAGLIO SULLA SOLUZIONE 25

25) leggeretutto.sh

ATTENZIONE, METTERE SPAZI PRIMA E DOPO **!=**

```
#!/bin/bash
exec {FD}<   ./miofileNoNL.txt
if (( $? == 0 )) ; then
    while read -u ${FD} RIGA ; [[ $? == 0  ||  ${RIGA} != ""  ]] ; do
            echo "${RIGA}"
     done
     exec {FD}>&-
fi
```

ESEMPIO FILE http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/miofileNoNL.txt
SOLUZIONE http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/leggitutto.sh

NELLA SOLUZIONE dell'esercizio 25, posso utilizzare una qualunque delle seguenti condizioni all'interno del while
ma attenzione a mettere gli spazi prima e dopo gli operatori

```
while read -u ${FD} RIGA ; if [[ $? -eq 0  || -n ${RIGA} ]] ; then true ; else false ; fi ; do
while read -u ${FD} RIGA ; [[ $? -eq 0  ||  -n ${RIGA} ]] ; do
while read -u ${FD} RIGA ; [[ $? -eq 0  || ${#RIGA} -ne 0 ]] ; do
while read -u ${FD} RIGA ; [[ $? -eq 0  || "${#RIGA}" > "0" ]] ; do
while read -u ${FD} RIGA ; [[ $? -eq 0  || ${RIGA} != "" ]] ; do
while read -u ${FD} RIGA ; [[ $? -eq 0  || ${RIGA} != "" ]] ; do
while read -u ${FD} RIGA ; [[    $?   !=   0    ||    ${RIGA}   !=   ""  ]] ; do
```

Familiarizzarsi con i comandi di shell (10)

26) Invocare il comando echo in modo da fargli scrivere **nello stderr** la parola CIAO. Verificare che la parola NON vada sullo stdout.

27) Scrivere il file di testo asterischi.txt che contiene le seguenti linee di testo, ciascuna formata da 4 parole. Il file verrà usato come standard input per lo script star.sh

```
uno   due  tre  quattro
alfa beta   *    gamma
one  two  three  four
```

Scrivere poi lo script star.sh che deve leggere una per una le righe provenienti dallo standard input e stampare a video, per ciascuna riga, la quarta e la terza parola della riga, in quest'ordine, separate da spazi.
L'output dovrà essere :

```
quattro   tre
gamma   *
four   three
```

Lanciare lo script star.sh passandogli nello standard input il contenuto del file asterischi.txt. Verificare che l'output sia corretto.

Familiarizzarsi con i comandi di shell (10)'
Soluzioni di alcuni esercizi slide precedente

26) echo CIAO 1>&2

verificare lanciando echo CIAO 1>&2 | grep -v CIAO

se ridiretto su stderr dovremmo vedere CIAO a video nonostante il flag -v

27) script star.sh

```
while (( 1 )) ; do
        read prima seconda terza quarta
        if (( $? == 0 )) ; then
                echo "${quarta} ${terza}"
        else
                break
        fi
done
```

Occhio, i doppi apici sono necessari per disabilitare l'interpretazione degli *

Eseguire così:

./star.sh < asterischi.txt

Lezione 5 in laboratorio

a inizio lezione dire due parole su wget e spiegare la variabile RANDOM

- read, flussi di controllo in bash, processi, etc etc:

  occhio a quello che fai

Familiarizzarsi con i comandi di shell (11)

28) Scrivere un file di testo di almeno 5 righe, ciascuna contenente almeno 7 parole separate da spazi e da tabulazioni. Mettete in qualche parola anche un asterisco *. Poi utilizzare il comando cut per visualizzare a video, di ciascuna riga, solo i caratteri dal terzo al quinto e dal decimo al quindicesimo, compresi gli estremi.

29) Aggiungere al comando precedente, in pipe, tanti comandi (suggerimento: sed) per eliminare dall'output finale tutti gli spazi bianchi, tutte le tabulazioni, tutti gli asterischi. Vi ricordo di quotare con gli ' l'argomento passato a sed, per impedire che la bash ne interpreti il contenuto.

30) Aggiungere in pipe il comando word count (wc) e fargli contare il totale dei caratteri contenuti nelle righe visualizzate sullo standard output.

36) Scrivere uno script **unasiunano.sh** che prende delle righe di testo dallo standard input e visualizza le righe una si ed una no.

Familiarizzarsi con i comandi di shell (11)' Soluzioni di esercizi della slide precedente

28) cut –b 3-5,10-15 nomefile.txt

29) cut –b 3-5,10-15 nomefile.txt | sed 's/*//g' | sed 's/ //**g**' | sed 's/\t//g'

NB: Il carattere g, nell'ordine passato a sed, serve a far eliminare tutte le occorrenze di spazi bianchi e asterischi in ciascuna linea, e non solo il primo incontrato nella linea.

30)

cut –b 3-5,10-15 nomefile.txt | sed 's/*//g' | sed 's/ //**g**' | sed 's/\t//g' | wc –c

35) **unasiunano.sh**

```
SI=1
while  read RIGA ; do
        if ((  ${SI} == 0  )) ; then  echo  "${RIGA}"; SI=1
        else       SI=0
        fi
done
```

Script con controllo di flusso dei comandi (1)

36. Scrivere uno script **random.sh** che controlla ripetutamente il valore della variabile RANDOM e conta quante volte la variabile viene letta. Lo script si interrompe quando la variabile RANDOM assume un valore tale che la divisione modulo 10 di RANDOM valga esattamente 2. Prima di terminare, lo script scrive in output il numero di volte che la variabile e' stata controllata.
37. Scrivere uno script **elenco.sh** che elenca tutti i file e directory presenti nella directory corrente (escludendo dall'elenco la directory corrente, la directory superiore e i files nascosti). Per ciascuno di questi file e directory, lo script controlla se si tratta di una directory o no. Se si tratta di una directory lo script conta la lunghezza del nome della directory e lo accumula in una variabile locale **LungNomiDirectory**. Lo script, inoltre conta i file che non sono delle directory. Al termine lo script visualizza in output il numero di file di tipo non directory e la lunghezza accumulata dei nomi delle directory lette.

Script con controllo di flusso dei comandi (1)' soluzioni

36) random.sh

```
#!/bin/bash
#    inizializzo la variabile RANDOM con  il numero di secondi
#    trascorsi  dal 1970-01-01 00:00:00 UTC  modulo 32768
RANDOM=$(( `date +%s` %  32768  ))
NUM=0
while ((  ${RANDOM}%10   !=  2 )) ; do
       (( NUM=${NUM}+1 ))
done
echo "NUM=${NUM}"
```

37) elenco.sh

```
#!/bin/bash
LungNomiDirectory=0;
NumFileNonDirectory=0
for name in  `ls  ./`  ;  do
   if [[ -d ${name} ]] ; then
      (( LungNomiDirectory=${LungNomiDirectory}+${#name} ))
   else
      (( NumFileNonDirectory=${NumFileNonDirectory}+1 ))
   fi
done
echo "LungNomiDirectory=${LungNomiDirectory}"
echo "NumFileNonDirectory=${NumFileNonDirectory}"
```

Script con controllo di flusso dei comandi (2)

38. Scrivere un file contenente alcune (almeno 5) righe di testo, ciascuna con almeno 4 parole. Scrivere uno script **seconda.sh** che prende come unico argomento il nome di quel file. Dentro lo script utilizzare ripetutamente il comando read per leggere la seconda parola di ogni riga del file. Tutte le seconde parole devono essere concatenata in una variabile di nome OUT. Alla fine dello script, la variabile OUT deve essere visualizzata sullo standard output.

39. Scrivere un file di testo che contenga almeno 5 righe. Passare il contenuto del file nello standard input di uno script **seleziona.sh**. Lo script deve selezionare le sole righe che contengono almeno un carattere A e contare il numero di caratteri totali delle sole righe selezionate. Suggerimento: guardate cosa fa il comando wc -c

40. Verificare se il seguente comando va a buon fine oppure produce errori ed in questo ultimo caso capire quale è il problema.

```
for (( i=0; ls ./ ; i=i+1 )) ; do echo "${i}" ; done
```

Script con controllo di flusso dei comandi (2)' soluzioni

seconda.sh

```
#!/bin/bash
if (( $# != 1 )) ; then  echo "serve nomefile" ; exit 1 ; fi
if [[ ! -r $1 ]] ;  then  echo "il file $1 non esiste"; exit 2; fi
OUT=""
while read PRIMA SECONDA ALTRO  ;  do
     if [[  -n ${SECONDA} ]] ; then
                OUT=${OUT}${SECONDA}
     fi
done < $1
echo "OUT=${OUT}"
```

seleziona.sh

```
#!/bin/bash

  grep A | wc -c
```

Eseguirlo così:
cat nomefile **|** ./seleziona.sh

Divertimento con le stringhe (2)

41. Scrivere uno script **reversebizzarro.sh** che prende una stringa come unico argomento e mette in output la stringa con i caratteri in ordine invertito, il primo andrà per ultimo, l'ultimo per primo. Per complicarvi la vita, per implementare questo script potete utilizzare solo assegnamenti a variabili, cicli ed if a piacere, ed i comandi echo read e cut. Si suppone che la stringa passata come argomento non contenga caratteri e metacaratteri interpretati dalla bash.

41++ per tutti i file contenuti nella directory /usr/include/ (NON NELLE SOTTODIRECTORY) far vedere a video le righe che contengono almeno un carattere * e contare quante sono queste righe

Divertimento con le stringhe (2)
Soluzioni 41

41. reversebizzarro.sh

```
#!/bin/bash

# REVERSE=""
echo $1 |       while (( 1 )) ; do
                    read -n 1 CAR ;
                    if (( $? == 0 )) ; then
                            REVERSE=${CAR}${REVERSE}
                           else
                            echo stringa rovesciata ${REVERSE}
                            break
                    fi
            done
echo GUARDARE SE STAMPA IL CONTENUTO DI STA STRINGA  ${REVERSE}
```

METTETE UN comando ps prima del while
ED un comando ps dentro il while PER VEDERE QUANTI PROCESSI CI SONO
SCOPRIRETE CHE IL LOOP while VIENE ESEGUITO IN UNA SHELL
FIGLIA DELLA SHELL CHE ESEGUE LO SCRIPT

Divertimento con le stringhe (2)""

Soluzioni 41++

41++

```
grep -d skip   '*'    /usr/include/*
```

42. un pochino esagerato (1)

Un file denunce.txt contiene, in ciascuna riga, la descrizione di una denuncia sottoposta al vaglio della polizia. Ciascuna riga contiene 4 o più parole cioè nome e cognome dell'accusato seguito da un identificatore univoco della denuncia (un numero), seguito da una descrizione del reato formata da una o più parole.

Un secondo file processi.txt contiene delle righe con due parole ciascuna, un identificatore della denuncia e un identificatore del processo originato dalla denuncia.

Un terzo file verdetti.txt contiene righe formate ciascuna da un identificatore del processo seguito da una o più parole che descrivono il verdetto.

Scrivere uno script **errorigiudiziari.sh** che, per ciascun verdetto contenuto nel file verdetti.txt stampa una riga contenente: nome e cognome dell'accusato, descrizione del reato e descrizione del verdetto.

I file contengono i seguenti dati (decidete voi i separatori: spazi bianchi o tab).

denunce.txt

vittorio ghini 13 vilipendio delle religioni
giovanni pau 17 tentata strage per indigestione
pierluigi mangani 69 incitazione alla poligamia

processi.txt

13 666
17 777
69 999

verdetti.txt

666 assolto per scomparsa delle prove
777 denuncia ritirata
999 prescrizione per fuga in brasile

SOLUZIONI 42 un pochino esagerato

Vi metto a disposizione due tipi di soluzioni:

una prima soluzione che richiede che tutte le righe di tutti i file di testo da leggere abbiano la andata a capo alla fine di ciascuna riga

https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SCRIPT_ESERCIZI/ERRORI GIUDIZIARI.tgz

una seconda soluzione che invece consente che l'ultima riga dei file di testo da leggere possa non avere l'andata a capo nell'ultima riga.

https://www.cs.unibo.it/~ghini/didattica/sistemioperativi/SCRIPT_ESERCIZI/ERRORI GIUDIZIARI_NO_NL.tgz

I due archivi tgzippati contengono lo script ed anche i tre file di testo con i dati che lo script legge.

Esercizi (1)

43) **Usando wget** scaricare lo script bash al seguente URL:

http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/esprcond_errato.sh

44)Verificare se lo script precedentemente scaricato esprcond_errato.sh funziona correttamente oppure produce errori. **Correggere** gli eventuali errori.

Esercizi (1)'
Soluzioni di esercizi slide precedente

43) wget http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/esprcond_errato.sh

44) aggiungere spazi prima e dopo [[e]]

```
if [[ -e /usr/include/stdio.h ]] ;then echo esiste;fi
```

Accorpamento

45) scrivere una riga di comando che mette sullo standard output le righe che stanno tra la 3° e la 5° posizione nel file /usr/include/stdio.h

46) scrivere una riga di comando che mette sullo standard output delle righe con i primi 3 caratteri delle ultime **4** righe del file /usr/include/stdio.h

47) scrivere una riga di comando che legge **due** righe da standard input e le mette in output in ordine invertito.

48) scrivere una riga di comando che legge **una** riga da standard input e mette in output niente

49) scrivere una riga di comando che legge righe da standard input e per ciascuna di queste mette in output il numero di caratteri che la compongono.

50) usare in sequenza le precedenti prime 2 righe di comando per generare output da passare come standard input all'insieme delle precedenti righe di comando (quelli dal terzo al quinto in quest'ordine), eseguiti in sequenza.

Accorpamento Soluzioni

45) head -n 5 /usr/include/stdio.h | tail -n 3

46) tail -n 4 /usr/include/stdio.h | cut -b -3

47) read RIGA1 ; read RIGA2 ; echo "${RIGA2}" ; echo "${RIGA1}"

48) read RIGA &> /dev/null

49) while read RIGA; do echo ${#RIGA}; done

50)

```
(
 head -n 5 /usr/include/stdio.h   |   tail -n 3       ;
 tail -n 4 /usr/include/stdio.h  | cut -b -3
) | (
 read RIGA1 ;  read RIGA2 ; echo "${RIGA2}" ; echo "${RIGA1}"    ;
 read RIGA &> /dev/null   ;
 while read RIGA; do echo ${#RIGA} ; done
)
```

Familiarizzarsi con processi e PID (1)

51. Scrivere uno script **puntini.sh** che prende come argomento a riga di comando un intero positivo che rappresenta un certo numero di secondi. Lo script deve rimanere in esecuzione per quel numero di secondi e, ad ogni secondo, stampare a video un punto . seguito dal proprio PID. Ma senza andare a capo.

52. Eseguire lo script precedente passandogli un argomento intero >= 30. Poi digitare alcuni comandi per sospendere lo script e mandarlo in background, poi riportarlo in foreground e poi dopo una decina di secondi sospenderlo e riportarlo in background. Infine killare lo script usando il suo pid (process identifier).

53. Eseguire lo script in modo da mandarlo direttamente in background subito, senza sospenderlo. Poi killare lo script.

Familiarizzarsi con Processi e PID (1)' Soluzioni di esercizi slide precedente

51.puntini.sh

```
#!/bin/bash
NUM=0
while (( ${NUM} <= $1 ))  ;  do
        sleep 1
        echo -n ". ${BASHPID}"
        ((NUM=${NUM}+1))
done
```

52. eseguire puntini.sh

```
./puntini.sh    30
CTRL   Z
bg
fg
CTRL   Z
bg
kill   -9   $!
```

53. eseguire puntini.sh

```
./puntini.sh 30  &
kill   -SIGKILL  $!
```

Familiarizzarsi con processi e PID (2)

54. Creare altri due script, **lanciaekilla.sh** e **lanciaeprendipid.sh**. Lo script lanciaeprendipid.sh deve lanciare in background 10 istanze dello script precedente puntini.sh **ridirigendo l'output di questi script sullo standard error**.

    Per ciascuno degli script puntini.sh lanciati, lo script lanciaeprendipid.sh deve pttenere il PID del processo lanciato. I 10 PID li deve mandare sullo standard output separati da spazi.

    Lo script lanciaekilla.sh invece deve lanciare in modo opportuno lo script lanciaeprendipid.sh, deve catturare l'output di questo e visualizzare l'elenco dei PID ottenuti.

    Poi deve usare i PID ottenuti per killare uno alla volta i processi puntini.sh lanciati da lanciaeprendipid.sh .

55. Creare uno script **lanciaricorsivo.sh** che necessita di un unico argomento intero che è il numero totale di processi discendenti che rimangono ancora da lanciare.

    Se il numero di discendenti da lanciare è maggiore di zero allora lo script lancia in background una nuova istanza di sé stesso passando come argomento il proprio argomento diminuito di 1.

    Lanciato il figlio, il padre visualizza in output il pid del figlio, poi aspetta che il proprio figlio termini la propria esecuzione.

    Il figlio nel frattempo crea un suo figlio (un nipote del primo) il quale etc etc.

    Lanciare inizialmente lo script passandogli 5 come argomento.

Familiarizzarsi con Processi e PID (2)'
Soluzioni

54.

lanciaekilla.sh

```
#!/bin/bash
PIDS=`./lanciaeprendipid.sh `
echo ${PIDS}
for  pid  in ${PIDS} ; do  kill -9  ${pid}; done
```

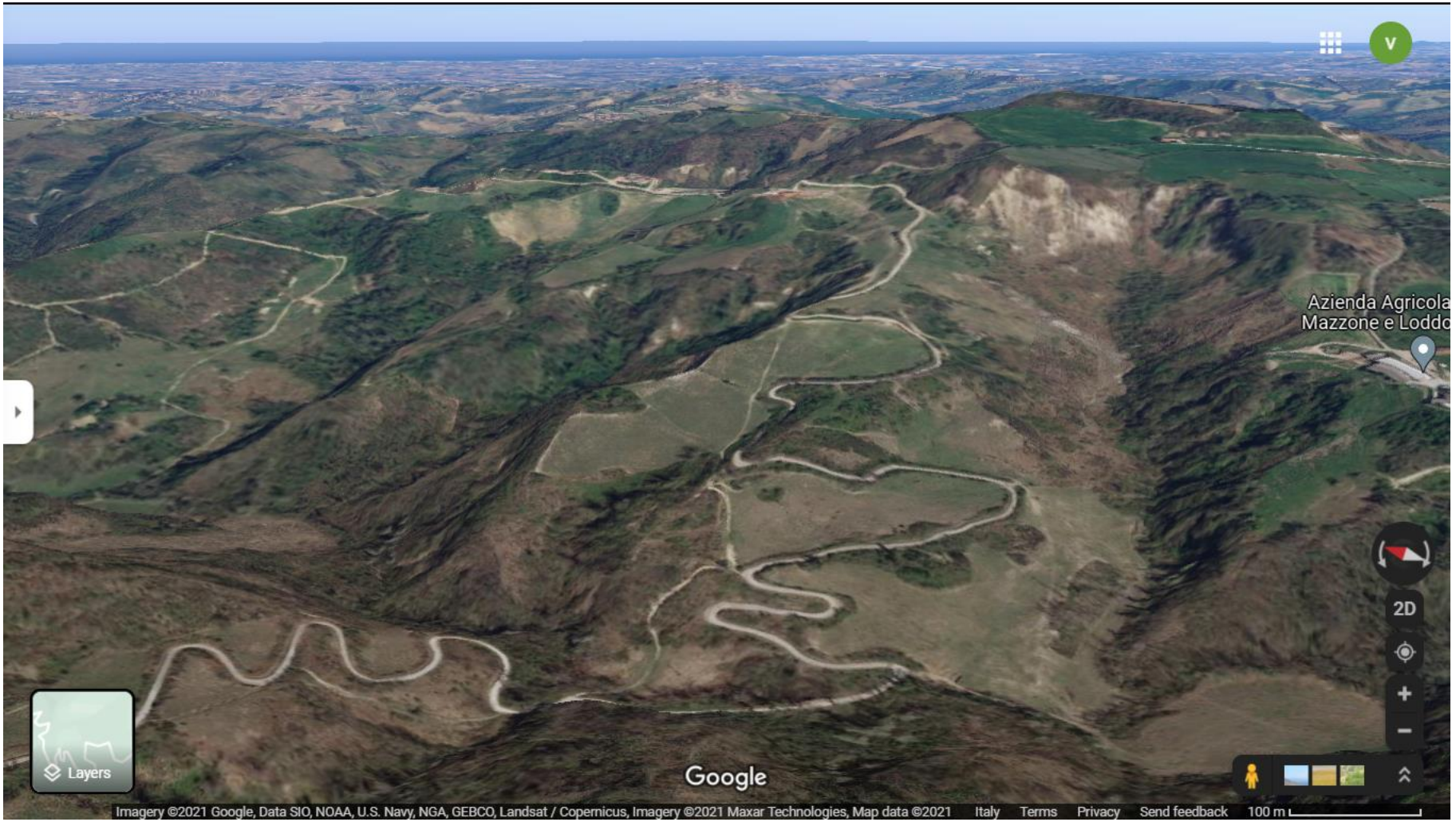
lanciaeprendipid.sh

```
for
((NUM=0;${NUM}<10;NUM=${NUM}+1));do
          ./puntini.sh 30  1>&2   &
          echo -n  "$!  " ;
done
```

55. lanciaricorsivo.sh

```
#!/bin/bash
if ((  $1  >  0  ))  ;  then
          ./lanciaricorsivo.sh  $((  $1  -1  ))  &
          echo  $!
          wait    $!
fi
```

Lezione 6 in laboratorio



- apt-get, installazione pacchetti software
- find, manipolazione di stringhe, etc etc:

occhio a quello che fai, Monte Farneto ti controlla dall'alto

Familiarizzarsi con i comandi di shell (11)

31) **STRANO**:

AGGIUNGERE CODICE ALLA SOLUZIONE DELL'ESERCIZIO 29

Usare lo stesso file di testo usato negli esercizi dal 28 al 30, cioè un file di testo di almeno 5 righe, ciascuna contenente almeno 7 parole separate da spazi e da tabulazioni. Mettete in qualche parola anche un asterisco *.

Supponendo che in ciascuna riga del file di testo ci siano al massimo un carattere di andata a capo di tipo \n ed uno di tipo "carriage return" \r , allora

aggiungere alla riga di comando della soluzione dell'esercizio **29**, in pipe, tanti comandi sed per eliminare dall'output finale anche tutte le andate a capo \n e \r.

Vi ricordo che è possibile inserire, nell' argomento passato a sed, i caratteri di andata a capo \n e \r esprimendoli mediante le sequenze di escape \n e \r
Ricordatevi di quotare con ' l'argomento passato a sed che descrive l'operazione che sed deve compiere, per impedire che la bash ne interpreti il contenuto.

Familiarizzarsi con i comandi di shell (11)
Soluzioni di esercizi della slide precedente

31) STRANO:

```
cut –b 3-5,10-15 nomefile.txt | sed 's/\r//g' | sed -z 's/\n//g' |
                    sed 's/*//g' | sed 's/ //g' | sed 's/\t//g'
```

NB: Quando elimino le andate a capo, metto il flag -z serve affinché sed non metta una andata a capo nell'output dopo avere processato una riga, e al suo posto metta un carattere ascii null di valore 0. Attenzione che se mancano le andate a capo, le successive chiamate a sed non sanno più dove finiscono le righe

Familiarizzarsi con i comandi di shell (12)

32) Scrivere uno script **cerca.sh** che prende come argomenti **un percorso assoluto** di una directory da cui cominciare la ricerca di alcuni ed **una stringa che contiene un nome di file da cercare**. Il nome potrebbe contenere dei metacaratteri, ad esempio il nome di file da cercare potrebbe essere *std*-h

Lo script deve visualizzare a video il percorso assoluto di tutti i file il cui nome corrisponde alla stringa passata come secondo argomento, partendo dalla directory specificata come primo argomento, e includendo nella ricerca tutte le sottodirectory.

Scrivere poi uno script **lanciacerca.sh** che lancia lo script cerca.sh passandogli **nell'adatto modo** proprio la directory iniziale /usr/include/ e la stringa *std*.h

33) Visualizzare a video il nome dei file che soddisfano due condizioni 1) stanno nelle directory immediatamente figlie della directory /usr/include/ 2) hanno nome che termina con net.h

34) Visualizzare a video le prime tre righe di ciascun file che sta nella directory /usr/include/ e in tutte le sue sottodirectory. Attenzione alle directory.

35) Aggiungere un comando in pipe al comando precedente per visualizzare solo i primi 3 caratteri di ciascuna riga.

Familiarizzarsi con i comandi di shell (12)'
Soluzioni di esercizi slide precedente

32) **cerca.sh**

```
find "$1" -name "$2" -print
```

lanciacerca.sh

```
./cerca.sh  '/usr/include/'  '*std*.h'
```

33)
```
find  /usr/include/  -mindepth 2  -maxdepth 2  -name "*net.h"
```

20)
```
find  /usr/include/  -type f  -exec  head -n 3 '{}' \;
```

34)
```
find  /usr/include/  -type f  -exec  head -n 3 '{}' \;  |  cut -b -3
```

oppure

```
for name in `find /usr/include/ -type f` ; do head -n 3 ${name}; done | cut -b -3
```

Installazione pacchetti (1)

Traccia:

sudo eseguire un comando con permessi di root

apt-get installa deinstalla pacchetti

sudo apt-get update aggiorna la lista LOCALE dei pacchetti installabili

sudo apt-get install *nomepkg* cerca di installare se il pkg è nella lista locale

con apt-get posso anche

deinstallare un pacchetto e tutti i files di configurazione

sudo apt-get **purge** *nomepkg*

reinstallare un pacchetto sovrascrivendo la vecchia installazione

sudo apt-get install --reinstall nomepkg

rimuovere pacchetti inutilizzati

sudo apt-get autoremove

sudo apt-get install aptitude aptitude ricerca pacchetti software in lista locale

esempio d'uso: aptitude search nome_software

aptitude search wget wget esegue download files da web

sudo apt-get install wget

Installazione pacchetti (2)

Provare se wget funziona

wget http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/Occhio3cm.jpg

permette download ricorsivo: Attenzione può scaricare TB

wget --recursive --level=2 --page-requisites --convert-links --no-parent URL_INIZIO

o analogamente con parametri corti

wget -r -l 2 -p -k -np URL_INIZIO

ad esempio

wget -r -l 2 -p -k -np
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/HomePageDir/Annunci.html

---

Installare un editor grafico, ad es geany

aptitude search geany geany dovrebbe esistere nella lista

sudo apt-get install geany installare editor grafico geany

dopo l'installazione nel menù Programming c'è l'icona di lancio di geany

Script con operatori di manipolazione di stringhe (1)

56. Scrivere uno script **scrivisustderr.sh** che legge delle righe passategli sullo standard input fino a che non incontra l'EOF. Le righe passate sullo stdin devono essere composte da parole separate da spazi. Per ciascuna riga lo script usa i comandi di estrazione di stringhe per estrarre la prima parola (separata da spazi bianchi) contenuta nella riga. Quella parola estratta deve essere scritta, mediante il comando echo, **sullo standard error**. Per ciascuna parola scritta sullo stderr, deve essere scritta la stringa evviva sullo standard output.

    Provare poi a passare, come input da tastiera, una riga con un * come prima parola.

57. Scrivere uno script **separa.sh** che

    - separa i diversi percorsi contenuti nella variabile di ambiente PATH (che vi ricordo sono separati da :).
    - e li visualizza uno per uno sullo standard output, ciascun percorso trovato in una riga di output che dopo il percorso contiene la lunghezza del percorso.

Script con operatori di manipolazione di stringhe (1) soluzioni

56) **scrivisustderr.sh**

```
#!/bin/bash
while read LINEA ; do
        PAROLA="${LINEA%% *}"
        echo "${PAROLA}" 1>&2
done
```

notare che gli argomenti dei comandi che usano le stringhe sono circondati da coppie di doppi apici "
per evitare che vengano interpretati eventuali metacaratteri * ? [] presenti nelle righe lette dal file

57) **separa.sh**

```
#!/bin/bash
RESIDUO=${PATH}
while [[ -n ${RESIDUO} ]]  ; do
        PRIMOPERCORSO=${RESIDUO%%:*}
        echo ${PRIMOPERCORSO}
        PRECEDENTERESIDUO=${RESIDUO}
        RESIDUO=${RESIDUO#*:}
        if [[ ${PRECEDENTERESIDUO} == ${RESIDUO} ]] ; then
                break
        fi
done
```

PER CASA - Divertimento con le stringhe (2)

58. Scrivere uno script **backslash.sh** che riceve righe di testo dallo standard input. Lo script deve leggere ciascuna riga e modificarla, aggiungendo davanti a ciascun metacarattere * ? [ ] un bel carattere backslash \ . Lo script deve mandare sullo standard output ciascuna riga modificata.
59. Scrivere uno script **separanomi.sh** che separa i nomi di ciascuna directory contenuta nella variabile di ambiente PATH, e li visualizza uno per uno, uno per ciascuna riga di output, sullo standard output. Ad esempio, se PATH="/bin:/usr/bin:/usr/sbin" , lo script dovrà mettere in output 4 righe contenenti rispettivamente bin usr bin usr sbin

PER CASA - Divertimento con le stringhe (2)' Soluzioni

58. backslash.sh

```
#!/bin/bash
while read RIGA ; do
        INDICE=0
        ACCUMULATO=""
        while (( ${INDICE} < ${#RIGA} ))  ; do
                CHAR=${RIGA:${INDICE}:1}
                if [[ ${CHAR} == "*" || ${CHAR} == "?" || ${CHAR} == "[" ||
    ${CHAR} == "]" ]] ; then
                        ACCUMULATO="${ACCUMULATO}\\${CHAR}"
                else
                        ACCUMULATO="${ACCUMULATO}${CHAR}"
                fi
                ((INDICE=${INDICE}+1))
        done
        echo "${ACCUMULATO}"
done
```

PER CASA - Divertimento con le stringhe (2)" Soluzioni

58. backslashPiuFacileDaScrivere.sh

```
#!/bin/bash
while read RIGA ; do
    RIGA="${RIGA//\*/\\\*}"
    RIGA="${RIGA//\?/\\\?}"
    RIGA="${RIGA//\[/\\\[}"
    RIGA="${RIGA//\]/\\\]}"
    echo "${RIGA}"
Done
```

SOLUZIONE CON sed **backslashSED.sh**

```
#!/bin/bash
while read RIGA ; do
        echo "${RIGA}" | sed 's/?/\\?/g;s/*/\\*/g;s/\[/\\\[/g;s/\]/\\\]/g'
done
```

O più semplicemente:

```
sed 's/?/\\?/g;s/*/\\*/g;s/\[/\\\[/g;s/\]/\\\]/g'
```

PER CASA - Divertimento con le stringhe (2)'"
Soluzioni

59. separanomi.sh

```
#!/bin/bash
RESIDUO=${PATH}
while [[ -n ${RESIDUO} ]]  ; do
    PRIMOCHAR=${RESIDUO:0:1}
    if [[ ${PRIMOCHAR} == "/" || ${PRIMOCHAR} == ":"  ]] ; then
            RESIDUO=${RESIDUO:1}
    else
            PEZZO1=${RESIDUO%%:*}
            PEZZO2=${RESIDUO%%/*}
            if ((  ${#PEZZO1} < ${#PEZZO2} )) ; then
                PEZZO=${PEZZO1}
            else
                PEZZO=${PEZZO2}
            fi
            echo ${PEZZO}
            OFFSET=$(( ${#PEZZO}+1 ))
            RESIDUO=${RESIDUO:${OFFSET}}
    fi
done
```