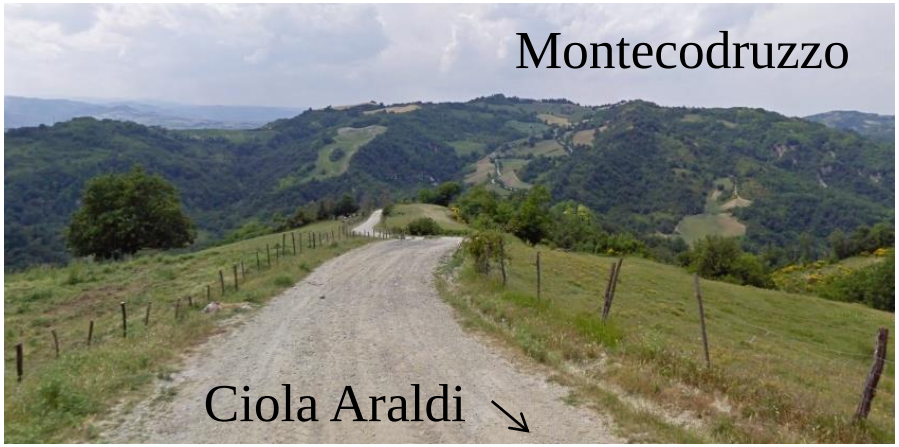
Lezione 6 in laboratorio - parte 2
stringhe in C, processi, macro, bash

hic sunt
canes stercore



NOTA BENE:

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

ps nohup disown bg fg kill wait gcc

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

Rimembranze di C: cercare e risolvere errori

Esercizio666: cercare e risolvere errori nel seguente file stringa.c

Il programma deve creare una stringa, modificarne il secondo carattere e poi stampare a video la stringa.

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
char *str="9876543210";
int main(void) {
        str[1]='f';
        printf("str= %s \n", str );   /* stampa 9f76543210  ? */
        fflush(stdout);
        return(0);
}
```

Compilare e linkare il codice usando i seguenti comandi:

```
gcc -c -ansi -Wpedantic -Wall  stringa.c
gcc -o stringa.exe stringa.o
```

Rimembranze di C: cercare e risolvere errori SOLUZIONE

Esercizio666: cercare e risolvere errori nel seguente file stringa.c

Il programma deve creare una stringa, modificarne il secondo carattere e poi stampare a video la stringa.

Correggere la dichiarazione della variabile str, facendola diventare così :

```
char str[]="9876543210";
```

oppure così :

```
char str[11]="9876543210";
```

Perché così funziona ?????

Se non riuscite a capirlo, usate il gcc per tradurre in assembly il codice C che provoca errore e verificare cos'è e dove viene collocata quella stringa "9876543210".

Spiegherò il problema tra circa 20 minuti.

comandi condizionali (1)

2. Capire che exit status viene restituito dal seguente script **bastardo.sh**
Ipotizziamo che la directory /usr/include esista.

```
( sleep 2; ls -d  /usr/include/ ) && {  [[ (! ( "false" > "true" ))  ||
            ( 3 -le 5 ) ]] ; } && if  [[ $? < "01" ]] ; then exit 0; else exit 1 ; fi ;
exit $?
```

comandi condizionali (1)
Soluzioni di esercizi slide precedente

2. bastardo.sh

l'esecuzione produce exit status 0

esercizio 95 - discendenti.sh

Scrivere uno script bash **discendenti.sh**, che prende un argomento intero a riga di comando. L'intero indica il numero di script figli da lanciare.
Ad esempio, all'inizio lo script potrebbe essere lanciato passandogli come argomento "3".

Lo script controlla l'argomento che gli è stato passato.

- Se il valore dell'argomento è maggiore di 0, lo script lancia in background lo script stesso tante volte quanto il valore dell'argomento intero e passa come argomento a ciascuno script proprio quel valore intero diminuito di 1. Poi lo script attende la fine di tutti i suoi processi figli. Poi stampa a video l'argomento che gli è stato passato. Infine termina restituendo 0.

- Se invece il valore dell'argomento è uguale a zero, allora lo script stampa a video l'argomento che gli è stato passato e poi termina restituendo 0.

soluzione esercizio 95 - discendenti.sh

```
#!/bin/bash

if (( "$#" != "1" )) ; then echo "serve un argomento intero" ; exit 1 ; fi
if (( "$1" < "0" )) ; then echo "serve un argomento intero maggiore o uguale a 0" ;
exit 1 ; fi

NUMFIGLI=$1
for (( i=0; $i < ${NUMFIGLI}; i=$i+1 )) ; do
        ./discendenti.sh $(( ${NUMFIGLI}-1 ))  &
done

wait

echo " ${NUMFIGLI}"

exit 0
```

Rimembranze di C: macro

Esercizio667: implementare macro che deve stampare il prodotto dei due valori interi passati come argomento

```
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#define STAMPAPRODOTTO( X, Y )      IMPLEMENTARE_LA_MACRO_QUI
int main(void) {
        STAMPAPRODOTTO( 3, 2 );          /* stampa 6 */
        STAMPAPRODOTTO( 3+5, 2 );        /* stampa 16 */
        return(0);
}
```

Compilare e linkare il codice usando i seguenti comandi:

```
gcc -c -ansi -Wpedantic -Wall  prodotto.c
gcc -o prodotto.exe prodotto.o
```

Rimembranze di C: macro SOLUZIONE

Esercizio667: implementare macro che deve stampare il prodotto dei due valori interi passati come argomento

```
#define STAMPAPRODOTTO( X, Y )          printf( "%i\n",  (X) * (Y) )
```

Notare che ho messo le parentesi tonde attorno agli argomenti quando li uso.

Esercizio **113 - insuff2.sh**

I due file RisultatiProvaPratica**1**.txt e RisultatiProvaPratica**2**.txt contengono in ciascuna riga di testo il Nome, il Cognome, la Matricola e il Voto ottenuti dallo studente nellla prova pratica N° 1 e N° 2 rispettivamente. Ciascun Nome e ciascun Cognome è composto da una sola parola. Il numero di matricola è univoco. Il Cognome e il nome, invece, potrebbero essere ripetuti. Il voto può essere non sufficiente (voto < 18).

Scrivere uno script bash **insuff2.sh** che metta in output l'elenco dei soli studenti che rispettano TUTTE le seguenti caratteristiche:

--- Hanno sostenuto la **seconda** prova prativa, ottenendo un voto NON sufficiente,

--- **Non** hanno sostenuto la **prima** prova pratica.

L'output deve essere formattato su più righe di testo. Ciascuna riga contiene le informazioni su uno studente, in particolare **la Matricola, il Nome, il Cognome ed il voto ottenuto nella seconda prova pratica, in quest'ordine**. Le righe dell'output devono essere **ordinate secondo il Cognome**, in senso crescente.

Esempio di file:

RisultatiProvaPratica1.txt

```
Avio  Verdi  876754  21
Dee  Bord  666666  20
Rino Ceronte  222222  13
Caio  Baro  777777  27
```

RisultatiProvaPratica2.txt

```
Carmine Ati 8888  23
Paolo Venzi  333333  9
Dee  Bord  666666  12
Sante Bo  888888  14
```

OUTPUT

```
888888 Sante Bo  14
333333 Paolo Venzi 9
```

SOLUZIONE

Esercizio **113 - insuff2b.sh**

```
#!/bin/bash

while read NOME COGNOME MATRICOLA VOTO ; do
        if (( ${VOTO} < "18" )) ; then
                LINES=`grep ${MATRICOLA} RisultatiProvaPratica1.txt | wc -l`
                if [[ "${LINES}" == "0" ]] ; then
                        echo  ${MATRICOLA} ${NOME} ${COGNOME} ${VOTO}
                fi
        fi
done < RisultatiProvaPratica2.txt | sort  -k 3
```

Il sort effettua l'ordine in base al terzo campo di ciascuna riga, il cognome

ALTRA SOLUZIONE BIZZARRA

Esercizio **113** **- insuff2.sh**

UN PO' BIZZARRA, USA UN TRUCCO: AGGIUNGE IL CAMPO COGNOME A INIZIO RIGA PER ORDINARE LE RIGHE PRODOTTE, E POI ELIMINA QUEL PRIMO CAMPO COGNOME PRIMA DI MANDARE LE RIGHE IN OUTPUT

```
#!/bin/bash
while read NOME COGNOME MATRICOLA VOTO ; do
    if (( ${VOTO} < "18" )) ; then
        LINES=`grep ${MATRICOLA} RisultatiProvaPratica1.txt | wc -l`
        if [[ "${LINES}" == "0" ]] ; then
            echo ${COGNOME} ${MATRICOLA} ${NOME} ${COGNOME} ${VOTO}
        fi
    fi
done < RisultatiProvaPratica2.txt | sort | cut -d ' ‘  -f2-
```

Il sort ordina in base al campo COGNOME a inizio riga.
Il cut finale elimina il campo COGNOME a inizio di ciascuna riga prima di mandarla in output. Notare che uso come delimitatore il carattere spazio (opzione -d ' ')

usare iterazioni bash e file

Esercizio1: piu' for per tutti

In una propria directory, creare 10 directory avente nome

1.0 1.1 1.2 1.3 1.4 ..... 1.9

Utilizzare il comando **for** ed il comando **mv** della bash, per cambiare i nomi delle directory rispettivamente in :

2.0 2.1 2.2 2.3 2.4 ..... 2.9

Suggerimento:guardare le slide su bash scripting, dove si parla di Estrazione di sottostringhe da variabili.

Esercizio2: e ancora un po' piu' di for per tutti

In una propria directory, creare 10 directory avente nome

1.0 1.1 1.2 1.3 1.4 ..... 1.9

Utilizzare il comando **for** ed il comando **mv** della bash, per cambiare i nomi delle directory rispettivamente in

2.9 2.8 2.7 2.6 2.5 ..... 2.0

Notare che, ad esempio, 1.1 deve diventare 2.8 e 1.3 deve diventare 2.6

In generale, 1.X deve diventare 2.(9-X)

usare iterazioni bash e file

Soluzione Esercizio1: piu' for per tutti

```
for (( NUM=0 ; ${NUM}<10 ; NUM=${NUM}+1 )) ; do mv 1.${NUM} 2.${NUM} ; done
```

Soluzione Esercizio2: : e ancora un po' piu' di for per tutti

```
for (( NUM=0 ; ${NUM}<10 ; NUM=${NUM}+1 )) ; do ((NEWNUM=9-${NUM})) ; mv 1.${NUM} 2.${NEWNUM} ; done
```

Esercizio **41 - script cercarecente**

Scrivere uno script bash **cercarecente.sh** che comincia cercando tutti i file con estensione .h i**n tutte le sottodirectory** della directory /usr/include/linux/ escludendo i files che si trovano direttamente nella directory /usr/include/linux/

Confrontare la data di ultima modifica dei file così trovati e stampare a video il nome del file modificato più recentemente.

Soluzione Esercizio **41 - script cercarecente**

```
#!/bin/bash

FILES=`find /usr/include/linux/ -mindepth 2 -name "*.h" -print`
if [[ -z ${FILES} ]] ; then
        echo "nessun file trovato"
else
# assegno a RECENTE il primo nome di file
        for RECENTE in ${FILES} ; do
                break
        done

        for NAME in ${FILES} ; do
                if [[ ${RECENTE} -ot ${NAME} ]] ; then
                        RECENTE=${NAME}
                fi
        done
fi
echo "il file piu' recente e' ${RECENTE}"
```

usare moduli, man, gcc, Makefile

Esercizio3: correggere errori nei Makefile e nei moduli C

All'indirizzo
http://www.cs.unibo.it/~ghini/didattica/sistemioperativi/ESERCIZI/ESERCIZI_CORREGGERE_ERRORI_1.tgz

c'e' un archivio in formato tar gz contenente una directory 1 che a sua volta contiene delle sottodirectory 1.1 1.2 1.3 1.4 ..... 1.8 1.9

In ciascuna sottodirectory c'e' il necessario per creare un eseguibile, ovvero i codici sorgenti in linguaggio ANSI C (esageratamente semplici) ed un Makefile.

Purtroppo ☺ sorgenti e Makefiles possono contenere degli errori.

I sorgenti e i makefile sono talmente semplici che DOVETE essere in grado di capire cosa fanno, anche se contengono errori.

Quindi, scaricate l'archivio, decomprimetelo in una vostra directory, e poi entrate in ciascuna delle directory in ordine crescente di secondo indice, cioe' prima 1.1 poi 1.2 poi 1.3

In ciascuna directory provate a generare l'eseguibile, correggendo gli eventuali errori.

Poi provate ad eseguire l'eseguibile, correggendo eventuali errori.

NB: decomprimere l'archivio fa parte dell'esercizio.