# Laboratorio di Sistemi Operativi
# 20 Gennaio 2021
### Compito

Si risponda ai seguenti quesiti, giustificando le risposte.

1. (6 punti) Si scriva uno script della shell `check_file_tree.sh` che prenda come argomento sulla linea di comando un percorso, controlli che corrisponda ad una directory e attraversi ricorsivamente il file system a partire da essa, stampando alla fine il numero di file e directory incontrati.

   Esempio:

   ```
   $ ./check_file_tree.sh /home/ivan
   Numero di file: 65856
   Numero di directory: 5318
   ```

   Esempio di soluzione:

   ```
   1  if ! test $# -eq 1
   2  then
   3          echo "Utilizzo $0 pathname"
   4      exit 1
   5  fi
   6
   7  num_file=0
   8  num_dir=0
   9
   10 if test -e $1 -a -d $1
   11 then
   12     lista=‘find $1 -print 2>/dev/null‘
   13     for i in $lista
   14     do
   15       if test -f $i
   16       then
   17         num_file=$[$num_file+1]
   18       fi
   19       if test -d $i
   20       then
   21         num_dir=$[$num_dir+1]
   22       fi
   23     done
   24 fi
   25
   26 echo "Numero di file: $num_file"
   27 echo "Numero di directory: $num_dir"
   28
   29 exit 0
   ```

2. (6 punti) Si scriva uno script della shell `check_login.sh` che prenda sulla linea di comando una lista arbitraria di stringhe e controlli se ognuna di esse sia un nome di login valido. In caso positivo deve stampare lo user ID corrispondente, mentre in caso negativo deve stampare un messaggio che indichi che non è un nome di login valido.

   Esempio:

   ```
   $ ./check_login.sh ivan gdm system
   ivan ha id 1000
   gdm ha id 121
   system non e’ un nome di login valido
   ```

**Suggerimento:** si ricorra al file `/etc/passwd` dove ogni linea corrisponde ad un account del sistema ed è una successione di campi separati dai due punti (`:`). Il nome di login è rappresentato dal primo campo, mentre il terzo campo è lo user ID. Ad esempio:

```
pippo:x:1017:1019:Pippo,,,:/home/pippo:/bin/bash
```

Esempio di soluzione:

```
n=$#
for((i=1; i<=$n; i++))
do
  if grep "^$1:" /etc/passwd 2>/dev/null >/dev/null
  then
    id=‘cat /etc/passwd | grep "^$1:" 2>/dev/null | cut -d’:’ -f3‘
    echo $1 ha id $id
  else
        echo "$1 non e’ un nome di login valido"
  fi
  shift
done
```

3. (6 punti) Si considerino le seguenti dichiarazioni in C:

```
int *a=(int*)malloc(10*sizeof(int));
int b[10];
```

Si dica, per ognuno dei seguenti comandi, se è lecito oppure no (motivando la risposta):

1. a[5]=67;
2. *(a+5)=67;
3. b[5]=67;
4. *(b+5)=67;
5. a=b;
6. b=a;

Risposte:

1. a[5]=67; è corretto: in generale, a[i] è sinonimo di *(a+i) e a è un puntatore ad interi che punta ad un'area di memoria allocata dinamicamente per contenere 10 interi; pertanto a[5] punterà all'indirizzo di memoria del sesto intero;
2. *(a+5)=67; è corretto: a è un puntatore ad interi che punta ad un'area di memoria allocata dinamicamente per contenere 10 interi; pertanto a+5 punterà all'indirizzo di memoria del sesto intero;
3. b[5]=67; è corretto in quanto b è il nome di un vettore di 10 interi;
4. *(b+5)=67; è corretto: in generale, *(b+i) è sinonimo di b[i];
5. a=b; è corretto: a è un puntatore a interi a cui viene assegnato l'indirizzo del primo elemento (di tipo intero) del vettore b;
6. b=a; non è corretto: il nome di un array non può apparire a sinistra di un comando di assegnamento in questo modo (è possibile assegnare soltanto i singoli elementi del vettore).

4. (4 punti) Si dica qual è l'output generato dal seguente programma C:

```c
#include <stdio.h>

int main() {
  int i,j;
  for(i=0; i<3; i++) {
    for(j=0; j<i; j++)
      printf(" ");
    for(j=0; j<(3-i)*2-1; j++)
      printf("+");
    printf("\n");
  }

  return 0;
}
```

```
+++++
 +++
  +
```

5. (10 punti) Si scriva un programma C che chieda all'utente quanti numeri interi pseudo-casuali vuole generare. Dopodiché il programma alloca memoria sufficiente per un array che possa contenere tali numeri ed inizia a generare un numero pseudo-casuale ogni 5 secondi. Alla fine stampa tutti i numeri generati ed il massimo fra questi e termina. Tuttavia, prima di giungere a terminazione, se l'utente preme Ctrl-C (ovvero, preme contemporaneamente i tasti Ctrl e C), il programma stampa il numero di interi generati fino a quel momento ed il massimo fra questi.

   **Suggerimento:** si ricorda che, per generare dei numeri pseudo-casuali, è sufficiente utilizzare la funzione di libreria `random()` come segue:

```c
#include <stdlib.h>
#include <time.h>
...
srandom(time(NULL)); // per inizializzare il seme con la data/ora attuale, assicurandosi di generare sequenze diverse ad ogni esecuzione
...
long int r;
r=random(); // genera un numero pseudo-casuale con valore compreso tra 0 e RAND_MAX e lo assegna alla variabile r
```

Esempio di soluzione:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <signal.h>
#include <unistd.h>

int compute_max=0;

void catchint(int signo) {
  compute_max=1;
```

```
}

int main() {
  int last_index, current_index, n;
  long int *buf, max;
  current_index=last_index=0;
  buf=NULL;
  max=-1;
  srandom(time(NULL));
  signal(SIGINT, catchint);
  printf("Inserisci il numero di interi da generare: ");
  scanf("%d",&n);
  buf=malloc(n*sizeof(long int));
  while(current_index<n) {
    if(compute_max) {
          for(int i=last_index;i<current_index;i++)
            if(max<buf[i])
              max=buf[i];
          printf("Numeri casuali generati finora: %d\n - massimo
             numero generato finora : %ld\n",current_index,max);
          last_index=current_index;
          compute_max=0;
    }
    else {
      buf[current_index]=random();
      current_index++;
    }
    sleep(5);
  }

  for(int i=0; i<n; i++) {
        if(i>=last_index) {
                if(max<buf[i])
                        max=buf[i];
        }
        printf("buf[%d]=%ld\n",i,buf[i]);
  }

  printf("Il massimo è: %ld\n",max);

  free(buf);
  return 0;
}
```