# Esercizi sul controllo dei processi[1]

## Esercizi

Per gli esercizi dovrete rispolverare quanto fatto nel corso di Informatica (linguaggio C). Funzioni che potrebbero essere utili sono le seguenti (riportate con il loro prototipo):

```
#include <stdio.h>
int printf(const char *format, ...)

#include <unistd.h>
unsigned int sleep (unsigned int seconds)

#include <stdlib.h>
void srand (unsigned int seed)
int rand (void)
```

**Ricordate:** Il comando `man` è vostro amico (per la sezione riguardante C: `man -s2`).

1. Scrivere un programma C `crea_processo.c` che esegue le seguenti operazioni:

   - stampare a video il proprio PID
   - crei un processo figlio e ne attenda la terminazione
   - quando il figlio termina ne stampi il PID, saluti e termini anch'esso.

   Il processo figlio dovrà:

   - stampare il proprio PID e quello del processo padre
   - dormire 3 secondi
   - salutare e terminare.

   L'output atteso è del tipo:

```
> [Padre] PID = 1101
> [Figlio] PID = 1102, PID padre = 1101
> [Figlio] Ho dormito e adesso termino
> [Padre] Il figlio PID=1102 ha finito e anche io. Ciao!
```

[1] Dispense per il corso di Sistemi di Elaborazione Informazione I, Scuola Interfacoltà di Scienze Strategiche, Università di Torino
Docente: Alessia Visconti, http://di.unito.it/~visconti

2. Quanti processi sono generati dalle seguenti porzioni di codice? Perché?

```
...
unsigned int i;
for (i=0; i<3; i++) {
        int n = fork();
        if (n == 0) {
                printf("Sono il figlio");
        }
        else {
                printf("Sono il padre");
        }
}
...
```

```
...
unsigned int i;
for (i=0; i<3; i++) {
        int n = fork();
        if (n == 0) {
                printf("Sono il figlio");
                exit(0);
        }
        else {
                printf("Sono il padre");
        }
}
...
```

3. Scrivere un programma C `crea_5_processi.c` che esegue le seguenti operazioni:

   - saluti
   - crei 5 processi figli e ne attenda la terminazione (deve attendere tutti i figli)
   - quando i figli terminano saluti e termini anch'esso.

   I processi figli dovranno:

   - salutare, indicando il proprio pid
   - dormire un numero random di secondi (incluso tra 0 e 9)
   - salutare e terminare, indicando il proprio pid e i secondi dormiti

   L'output atteso è del tipo:

```
> [Padre] Sono il padre, sto per generare i miei figli
> [Figlio] Ciao, sono PID = 103
> [Figlio] Ciao, sono PID = 104
...
> [Figlio] Sono PID = 104. Ho dormito 1 secondi e termino
> [Figlio] Sono PID = 103. Ho dormito 4 secondi e termino
...
> [Padre] Tutti i miei figli si sono riposati: termino.
```

4. Scrivere un programma C `esegui.c` che esegue le seguenti operazioni:

   - segnala il proprio pid;
   - esegue il comando Unix che permette di listare il contenuto di una directory, includendo anche i file/cartelle nascosti

   Attenzione: i *programmi* che eseguono i comandi Unix si trovano nella cartella `/bin/*`, dove `*` indica il comando da eseguire.
   Il processo deve segnalare eventuali errori nell'esecuzione del comando richiesto. L'output atteso è del tipo:

```
> Sono il processo con PID=102
> .  ..  esegui  esegui.c
```

   oppure (in caso di fallimento!):

```
> Sono il processo con PID=102
> Impossibile eseguire il comando richiesto
```