Esercizio sui segnali Unix

Testo

Si realizzi un comando in ambiente Unix, che, utilizzando le *system call* del sistema operativo, soddisfi le seguenti specifiche:

- **Sintassi di invocazione:**

  `esame com1 com2 ..comN`
- **Significato degli argomenti:**
- `esame`: nome dell'eseguibile generato;
- `com1, com2, ..comN:` nomi di comandi (file eseguibili).
-

Testo (continua)

- **Comportamento dei processi:**

Il processo iniziale (P_0) deve creare N processi figli (P_1, P_2, .. P_N).

Comportamento del generico figlio P_i (i=1,..N):

Il processo figlio $\mathbf{P_i}$, una volta creato, deve porsi in attesa di un'eventuale ***attivazione*** da parte del padre. In caso di attivazione, Pi eseguira` il comando $\mathbf{com_i}$.

Comportamento del padre P_0:

Il processo padre ($\mathbf{P_0}$), una volta creati i figli, definira` il suo comportamento in base al valore del proprio pid (ppid):

- se **ppid e` pari**, attivera` i figli con pid pari e terminera` forzatamente i figli con pid dispari;
- se **ppid e` dispari**, attivera` i figli con pid dispari e terminera` forzatamente i figli con pid pari;

Successivamente, dopo aver raccolto e stampato lo stato di terminazione di tutti i figli, il padre terminera` la propria esecuzione.

Soluzione:

```
#include <stdio.h>
#include <signal.h>
#define max 10

void gestore_att(int sig);
void figlio(char *com);

main(int argc , char *argv[])
{
  int ppid, pid[max], pf;
  int   status,  i;
   if (argc==1)
  { printf("sintassi sbagliata!\n");
    exit(1);
  }
 ppid=getpid();
```

```
for(i=0; i< argc-1; i++)
 {  pid[i]=fork();
    if (pid[i]==0)
    { figlio(argv[i+1]);
              exit(0);
        }
     else
      printf("%d: creato figlio %d\n", ppid, pid[i]);
}
sleep(2); /* solo per ritardare..*/
if ((ppid%2)==0)
   /*attivazione dei pari e uccisione dei dispari:
  SIGUSR1 */
        for(i=0; i<argc-1; i++)
                kill(pid[i], SIGUSR1);
else
for(i=0; i<argc-1; i++)
                kill(pid[i], SIGUSR2);
```



```
for (i=0; i<argc-1; i++)
{    pf=wait(&status);
     if ((char)status==0)
printf("terminato%d con stato%d\n", pf, status>>8);
     else
printf("terminato   %d   involontariamente   (segnale
  %d)\n", pf, (char)status);
}
exit(0);
} /* fine padre */
```



```
void figlio(char *com)
{       int miopid;
        miopid=getpid();

        if ((miopid%2)==0)
                signal(SIGUSR1, gestore_att);
        else
                signal(SIGUSR2, gestore_att);
        pause();
        execlp(com, com,(char *)0);
}

void gestore_att(int sig)
{               printf("%d:  sono  stato  attivato!\n",
  getpid());
        return;

}
```


Esecuzione:

```
[aciampolini@ccib48 esercizi]$ gcc -o segnali segnali.c

[aciampolini@ccib48 esercizi]$ ./segnali ps ls date
27725: creato figlio 27726
27725: creato figlio 27727
27725: creato figlio 27728
27727: sono stato attivato!
terminato 27728 involontariamente (segnale 12)
terminato 27726 involontariamente (segnale 12)
processi  processi.c  segnali  segnali.c
terminato27727 con stato0
[aciampolini@ccib48 esercizi]$
```