

Soluzione 16 Gennaio 2003

Il comando dovrà funzionare nel modo seguente: il processo 'padre' P0 deve creare 2 processi figli: P1 e P2;

- ciascun figlio P_i (i=1,2) dovrà accedere al file f in lettura, per "campionare" dal file 1 carattere ogni N_i caratteri letti, e confrontarlo con il carattere C_i dato come argomento. Se il carattere "campionato" risulta uguale a C_i, P_i dovrà notificare in modo asincrono l'evento al padre P0.
- una volta creati i 2 figli, il padre P0 si sospende in attesa di notifiche da parte dei figli: per ogni notifica rilevata, P0 dovrà scrivere il pid del processo che l'ha trasmessa in un file di nome "notifiche" e rispondersi.

Il primo figlio che termina la lettura del file dovrà provocare la terminazione dell'intera applicazione.

Esercizio

Si scriva un programma in C che, utilizzando le *system call* di unix, preveda la seguente sintassi:

```
esame N1 N2 f C1 C2
```

dove:

esame è il nome dell'eseguibile da generare

- N1, N2 sono interi positivi
- f è il nome di un file
- C1, C2 sono singoli caratteri

Il comando dovrà funzionare nel modo seguente:

- il processo 'padre' P0 deve creare 2 processi figli: P1 e P2;

Soluzione dell'esercizio

```
#include <fcntl.h>
#include <stdio.h>
#include <signal.h>
int PID1, PID2, fd;

void gestore_F(int sig);
void gestore_T(int sig);

void figlio(char *com);

main(int argc, char *argv[])
{
    int N1, N2;
    char C1, C2;
    char S1[80], S2[80];
```

```

if (argc!=6)
{ printf("sintassi sbagliata!\n");
  exit(1);
}

```

```

N1=atoi(argv[1]);
N2=atoi(argv[2]);
C1=argv[4][0];
C2=argv[5][0];
signal(SIGUSR1, gestore_F);
signal(SIGUSR2, gestore_F);
signal(SIGQUIT, gestore_T);
fd=open("notifiche", O_WRONLY);
PID1=fork();

```

```

else if (PID2<0) exit(-1);
while(1)
  pause();
}

void gestore_F(int sig)
{
  int PID;

  printf("%d: ricevuto %d!\n", getpid(), sig);
  if (sig==SIGUSR1)
    PID=PID1;
  else PID=PID2;
  write(fd, &PID, sizeof(int));
  return;
}

```

```

if (PID1==0) /*codice figlio P1*/
{
  fd=open(argv[3], O_RDONLY);
  while (read(fd, &S1, N1)>0)
    if (S1[0]==C1)
      kill(getppid(), SIGUSR1);
  kill(getppid(), SIGQUIT);
  exit(0);
}
else if (PID1<0) exit(-1);
PID2=fork();
if (PID2==0)
{ /*codice figlio P2*/
  fd=open(argv[1], O_RDONLY);
  while (read(fd, &S2, N2)>0)
    if (S2[0]==C2)
      kill(getppid(), SIGUSR2);
  kill(getppid(), SIGQUIT);
  exit(0);
}
}

```

```

void gestore_T(int sig)
{
  printf("%d: ricevuto segnale di
  terminazione!\n", getpid());
  close(fd);
  kill(0, SIGKILL);
}
}

```