

Soluzione 11 Gennaio 2002

- Il processo P0, una volta creato il processo figlio P1, dovrà leggere e stampare sullo standard output quanto inviatogli dal processo figlio P1, secondo le seguenti modalità:
 - Ogni carattere letto diverso da Cstop e da Cecc, viene stampato da P0 sullo standard output;
 - Nel caso in cui P0 legga il carattere **Cstop**, dovrà semplicemente terminare forzatamente l'esecuzione di entrambi i processi (P0 e P1);
 - Nel caso in cui P0 legga il carattere **Cecc**, P0 dovrà interrompere l'esecuzione del figlio P1; P1 dal momento dell'interruzione in poi, passerà ad eseguire il comando **Comando**, e successivamente terminerà.

Esercizio

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

Sintassi di invocazione:

esame filein Comando Cstop Cecc

Significato degli argomenti:

- **esame**: nome dell'eseguibile generato.
- **filein**: nome di un file leggibile.
- **Comando**: nome di un file eseguibile.
- **Cstop**, **Cecc**: singoli caratteri.

Comportamento:

Il processo iniziale (**P0**) deve creare un processo figlio (**P1**).

- P1 dovrà leggere il contenuto del file **filein**, e trasferirlo integralmente al processo padre P0.

Soluzione dell'esercizio

```
#include <stdio.h>
#include <signal.h>
#include <fcntl.h>
```

```
int pp[2];
char com[20];
void trap(int num);
```

```
main(int argc, char **argv)
{ int pid0, pid1, fd, k, status;
  char filein[20], buf[40], c;
```

```

if (argc!=5)
{ printf("Sintassi errata!!- esame filein
  Comando Cstop Cecc\n");
  exit();
}
strcpy(com,argv[2]);
pipe(pp);
pid1=fork();

```

Sistemi Operativi L-A

5

```

else if (pid1>0) /* codice padre */
{ close(pp[1]);
  while((k=read(pp[0], &C, 1))>0)
  {
    printf("PADRE: %c\n", C);
    if(C==argv[3][0]) kill(0, SIGKILL);
    else if (C==argv[4][0])
    { kill(pid1, SIGUSR1);
      wait(&status);
      close(pp[0]);
      exit();
    }
    else write(1, &C, 1);
  }
  wait(&status);
  close(pp[0]);
  exit(0);
}

```

Sistemi Operativi L-A

7

```

if (pid1==0) /*codice figlio */
{ signal(SIGUSR1, trap);
  close(pp[0]);
  fd=open(argv[1], O_RDONLY);
  if (fd<0)
  { perror("open");
    exit();
  }
  while((k=read(fd, &buf, 40))>0)
  { printf("FIGLIO: ho letto %s\n", buf);
    write(pp[1],&buf, k);
  }
  close(fd);
  close(pp[1]);
  exit();
}

```

Sistemi Operativi L-A

6

```

else
{ perror("creazione!");
  exit();
}
/* fine main*/

void trap(int num)
{
  close(pp[1]);
  execlp(com, com, (char *)0);
  exit(-1);
}

```

Sistemi Operativi L-A

8