SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 05-06) – 31 MARZO 2006

Esercizio

Si realizzi un programma **concorrente** per UNIX che deve avere una parte in **Bourne Shell** e una parte in **C**.

La parte in Shell deve prevedere due parametri: il primo deve essere il **nome assoluto di un direttorio** che identifica una gerarchia (**G**) all'interno del file system, mentre il secondo parametro deve essere considerato una stringa **S** di tre caratteri. Il programma deve cercare nella gerarchia **G** specificata tutti i direttori che contengono almeno **un** file il cui nome abbia come terminazione **.S** e la cui lunghezza sia multiplo intero di **51**. Si riporti il nome assoluto di tali direttori sullo standard output. In ogni direttorio trovato e per ogni file trovato **Fi** che soddisfa la condizione precedente, si deve invocare la parte in C, passando come parametro il nome del file **Fi**.

La parte in C accetta un parametro che rappresenta il nome di un file **Fi binario** che si supponga costituito da soli numeri interi positivi. Il processo padre per prima cosa deve inizializzare un array **A** di quattro elementi interi con i numeri 2, 3, 5 e 7 e quindi deve generare **4 processi figli** (**P0** … **P3**): ogni processo figlio è associato ad uno degli elementi dell'array **A**. Ognuno di tali processi figli esegue concorrentemente aspettando il numero comunicato o dal padre (nel caso del primo figlio) o dal fratello precedente e quindi prova a dividere tale numero per il numero ad esso associato: solo se il numero non è divisibile, il processo figlio lo comunica al processo figlio successivo, a parte l'ultimo figlio che deve riportare il numero sullo standard output. Il processo padre deve concorrentemente leggere i numeri interi dal file **Fi** (si supponga tali numeri sempre strettamente minori di **51**) e li comunica al primo figlio.
Ogni processo figlio deve ritornare al padre il numero di numeri divisibili e quindi scartati.
Il padre, dopo che i figli sono terminati, deve stampare su standard output i PID di ogni figlio con il corrispondente valore ritornato.

```
#!/bin/sh
# Soluzione del compito di Sistemi Operativi e Laboratorio del 31
# marzo 2006
#
#  ============= file del controllo parametri ==============
# PARAMETRI:
# $1) dirass $2) S

# controllo numero dei parametri
case $# in
2);;
*) echo "Errore nel numero dei parametri"
   exit 1;;
esac

#controllo primo parametro = direttorio assoluto
case $1 in
/*)
    if test ! -d $1 -o ! -x $1
    then
     echo "Direttorio non esistente o non accessibile"
     exit 2;
    fi;;
 *)
    echo "Direttorio non assoluto"
    exit 3;;
esac

# controllo secondo parametro = numero intero strettamente positivo
# minore di 50

case $2 in
???);;
*)
     echo "Errore: $2 non e' una stringa di tre caratteri"
     exit 3;;
esac

# Salvo il numero nella variabile che poi esportero'
S=$2

#Impostazione della variabile PATH
PATH=$PATH:`pwd`
export PATH
export S

#chiamata alla parte ricorsiva
parteRicorsiva.sh $1
#!/bin/sh
```

```
# Soluzione del 31 marzo 2006
#
# ============= PARTE RICORSIVA ======================
# $1) dirass
#

# variabili di subshell
LISTA_FILE=    # contiene la lista di tutti i file del direttorio in
               # esame che soddisfano la condizione del testo

# entro nella sottogerarchia
cd $1

for i in *
do
    if test -f $i -a -r $i
    then

     # test delle condizioni del file
     dim=`wc -c <$i`
     resto=`expr $dim % 51`

     if test $resto -eq 0
     then
          case $i in
          *.$S)
               echo "Il file $i soddisfa le condizioni"
               LISTA_FILE="$LISTA_FILE $i";;
          *)
               echo "Il file $i NON soddisfa le condizioni";;
          esac
     fi

    fi
done

# Ho trovato almeno un nome di file del direttorio corrente ?
if test ! -z "$LISTA_FILE"
then
    echo "Trovato il direttorio $1"
    echo "Invoco la parte c per ciascuno dei file trovati"
    for i in "$LISTA_FILE"
    do
          parteC $i
    done
fi


# ricerca ricorsiva nei sottodirettori
for i in *
do
    if test -d $i -a -x $i
    then
     $0 `pwd`/$i
    fi
```

```
done
```

```c
#include <stdio.h>
#include <string.h>
#include <signal.h>
#include <fcntl.h>
#include <sys/stat.h>
#define PERM 0644 /* in UNIX */

typedef int pipe_t[2];

int main(int argc, char* argv[])
{
	/* ======== VARIABILI LOCALI ================ */
	int A[4]={2,3,5,7};// numeri dell'array A
	int i,j;		// indici cicli for
	int pid;		// pid per processi
	pipe_t piped[4];	// pipe descriptors
	int numero;		// numero letto dalla pipe
	int cont;		// contatore dei file divisibili per il proprio numero
				// e quindi scartati
	int status;		// variabile di stato per la wait
	int fdr;		// file descriptor per il padre
	/* ========================================== */

	/* Controllo numero dei parametri */
	if(argc<2)
	{
		printf("Errore nel numero degli argomenti: %s nomefile\n", argv[0]);
		exit(-1);
	}

	/* Creazione delle 4 pipes*/
	for(i=0; i<4; i++)
		if(pipe(piped[i])<0)
		{
			printf("Errore di creazione della pipe\n");
			exit(-2);
		}

	/* Generazione dei 4 processi figli */
	for(i=0; i<4; i++)
	{
		if((pid=fork())<0)
		{
			printf("Errore nella fork\n");
			exit(-3);
		}

		if(pid==0) /* CODICE DEL FIGLIO */
		{
			/* Prima chiudo le pipe che non uso */
			for (j=0; j<4; j++)
			{
				if(j!=i+1) close(piped[j][1]);
				if(j!=i) close(piped[j][0]);
			}

			cont=0;
			printf("Sono il figlio %d e comincio a leggere\n", i);
```

```
            /* aspetto il numero comunicato dal precedente */
            while (read(piped[i][0], &numero, sizeof(int)), numero != -1)
            {
                  printf("Sono il figlio %d e ho letto %d\n", i, numero);

                  if( (numero % A[i]) != 0)
                  /* il numero letto non è divisibile per il mio numero */
                  {
                        if(i==3) /* Sono l'ultimo figlio */
                              printf("%d\n", numero);
                        else
                              write(piped[i+1][1], &numero, sizeof(int));
                  }
                  else
                        cont++;
            }

            if(i<3) /* passo il -1 a fratello seguente, tranne per l'ultimo*/
                  write(piped[i+1][1], &numero, sizeof(int));

            printf("Sono il figlio %d e muoio con cont=%d\n", i, cont);

            /* suppongo cont <= 255*/
            exit(cont);

      } /* fine codice del figlio*/
} /* FINE FOR */

/* CODICE DEL PADRE*/

/* Chiusura di tutte le pipe tranne la prima in scrittura */
for(i=0; i<3; i++)
{
      close(piped[i][0]);
      if(i!=0) close(piped[i][1]);
}

/* apro il file passato come primo argomento */
if((fdr=open(argv[1], O_RDONLY))<0)
{
      printf("Errore nella apertura del file %s\n", argv[1]);
      exit(-4);
}

while(read(fdr, &numero, sizeof(int))>0)
{
      printf("Sono il padre ed ho letto il numero \"%d\"\n", numero);

      /* scrivo il numero letto nella pipe del primo figlio */
      write(piped[0][1], &numero, sizeof(int));

} /* fine while di lettura dal file fdr*/

/* invio il segnale al primo figlio di terminare la catena*/
numero=-1;
write(piped[0][1], &numero, sizeof(int) );
```

```
	/* Aspetto i vari figli */
	while((pid=wait(&status))>0)
	{
		printf("Il figlio con PID %d è morto con codice di ritorno %d\n", pid,
		(status >> 8)&0xFF );
	}

}
```