# SISTEMI OPERATIVI e
# LABORATORIO DI SISTEMI OPERATIVI
# (A.A. 06-07) – 16 MARZO 2007

## IMPORTANTE:

1) Fare il login sui sistemi in modalità Linux usando il proprio **username** e **password**.
2) I file prodotti devono essere collocati in un **sottodirettorio** della propria **HOME** directory che deve essere creato e avere nome **ESAME16Mar07-1-1.** FARE ATTENZIONE AL NOME DEL DIRETTORIO, in particolare alle maiuscole e ai trattini indicati. Verrà penalizzata l'assenza del direttorio con il nome indicato e/o l'assenza dei file nel direttorio specificato, al momento della copia automatica del direttorio e dei file. **ALLA SCADENZA DEL TEMPO A DISPOSIZIONE VERRÀ INFATTI ATTIVATA UNA PROCEDURA AUTOMATICA DI COPIA, PER OGNI STUDENTE DEL TURNO, DEI FILE CONTENUTI NEL DIRETTORIO SPECIFICATO.**
3) Il tempo a disposizione per la prova è di **75 MINUTI** per lo svolgimento della sola parte C.
4) Non è ammesso **nessun tipo di scambio di informazioni** né verbale né elettronico, pena la invalidazione della verifica.
5) L'assenza di commenti significativi verrà penalizzata.
6) **AL TERMINE DELLA PROVA È INDISPENSABILE CONSEGNARE IL TESTO DEL COMPITO (ANCHE IN CASO CHE UNO STUDENTE SI RITIRI): IN CASO CONTRARIO, NON POTRÀ ESSERE EFFETTUATA LA CORREZIONE DEL COMPITO MANCANDO IL TESTO DI RIFERIMENTO.**

## Esercizio

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

### Parte in Linguaggio C

La parte in C accetta un numero variabile di parametri che rappresentano nomi di file **F1...FN**. Il processo padre deve creare un file il cui nome (**FCreato**) risulti dalla concatenazione della stringa "collect." con la stringa "a" e quindi deve generare **N processi figli** (**P1** … **PN**): ogni processo figlio è associato ad uno dei file **Fi**. Ogni processo figlio deve cercare le occorrenze del carattere 'a' nel file associato **Fi**. Il processo figlio, dopo aver trovato 4 occorrenze del carattere 'a', deve comunicare al processo padre che sono state trovate tali occorrenze e chiedere al processo padre il permesso di scrivere tali occorrenze sul file **FCreato**. Il processo padre, mediante il conteggio delle autorizzazioni fornite ai figli, determina se autorizzare o meno la scrittura delle 4 occorrenze trovate da un processo figlio: in particolare, il padre autorizza le scritture fino a che, globalmente, i figli hanno scritto 400 occorrenze.
Ogni processo figlio deve ritornare al padre il numero di caratteri scritti sul file **FCreato.**
Il padre, dopo che i figli sono terminati, deve stampare su standard output i PID di ogni figlio con il corrispondente valore ritornato.

```c
/* Soluzione della 2° prova in itinere del 16 marzo 2007 */
/* Soluzione CON UNA SOLA PIPE E SEGNALI */
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#define PERM 0644 /* in UNIX */

/* Variabili globali --> inizializzate a 0 per default */
int contaScritti;       /* contatore delle occorrenze scritte da ciascun figlio*/
int fdCreato;           /* file descriptor associato ad FCreato */
/* ----------------- */


int scrivi(int sig)
{
      char car[5]="aaaa";
      contaScritti+=write(fdCreato, car, 4);
      signal(SIGUSR1, scrivi);
}

int esci(int sig)
{
      exit(contaScritti);
}

main(int argc, char **argv)
{
      /* Variabili locali */
      char FCreato[10]; /* spazio per il nome del file da creare*/
      int N;            /* numero di file passati sulla riga di comando*/
      int i;            /* indice per cicli for       */
      int *pid;         /* pid[] per il ritorno di fork()*/
      int Fi;           /* file descriptor del file assegnato a ciascun figlio*/
      char ch;          /* buffer dove mettere ciascun carattere letto da Fi*/
      int status;       /* variabile di stato della wait    */
      int pipeFP[2];    /* la pipe dove i figli scrivono i loro pid al padre*/
      int occ=0;        /* contatore delle 4 occorrenze di 'a' */
      /* -----------------*/

      /* Controllo sul numero dei parametri */
      N=argc-1;
      if(N == 0)
      {
            printf("Errore nel numero dei parametri. Uso: %s file1 file2 ...
            fileN\n",argv[0]);
            exit(-1);
      }

      strcpy(FCreato, "collect.");
      strcat(FCreato, "a");

      /* Apro il file "collect.a" --> VIENE EREDITATO APERTO DAI FIGLI
      --> I/O POINTER CONDIVISO!!!! */
      if((fdCreato = creat(FCreato, PERM))<0)
      {
            printf("Errore di creazione del file %s\n", FCreato);
```

```
	exit(-4);
}

/* Apro la pipe di collegamento coi figli */
if((pipe(pipeFP)<0))
{
	printf("Errore di apertura delle pipe\n");
	exit(-5);
}

/* Alloco l'array dei pid */
pid = (int *) malloc(N*sizeof(int));
if(!pid)
{
	printf("Errore di malloc()\n");
	exit(-5);
}

/* ciclo di creazione degli N figli */
for (i=0;i<N;i++)
{
	if((pid[i]=fork())<0)
	{
		printf("Errore nella creazione di un figlio\n");
		exit(-2);
	}
	else if(pid[i] == 0) /*codice del figlio*/
		break; /* esco dal ciclo */
} /* file ciclo for */

if(pid[i]==0) /* codice del figlio */
{
	signal(SIGUSR1, scrivi);
	signal(SIGUSR2, esci);

	printf("Sono figlio con PID %d ed apro il mio file %s\n", getpid(),
	argv[i+1]);

	/* Apro il file associato "Fi" */
	if((Fi=open(argv[i+1], O_RDONLY))<0)
	{
		printf("Errore nella apertura del file \"%s\"\n", argv[i+1]);
		exit(0); /* ritorno 0 occorrenze del carattere nro nel file */
	}

	/* Chiudo il lato delle pipe che non uso*/
	close(pipeFP[0]);


	/* Cerco le occorrenze del carattere 'a' nel file Fi */
	while(read(Fi, &ch, 1)>0)
	{	if (ch == 'a')
			occ++;

		if(occ==4)
		{
			/* comunico al padre il mio pid */
```

```
                        write(pipeFP[1], &i, sizeof(int));

                        /* mi metto in attesa di un segnale */
                        pause();


                        /* ricomincio a contare le occorrenze */
                        occ=0;
                  }
            }

            /* ritorno al padre il conteggio delle occorrenze scritte */
            exit(contaScritti);

      } /* fine figlio */

      /* codice del padre */

      /* Il padre chiude i lato della pipe che non usa */
      close(pipeFP[1]);

      /* comincio a leggere le occorrenze trovate dai figli */
      while(contaScritti < 400)
      {
            /* leggo il pid del figlio che ha trovate 4 occorrenze */
            read(pipeFP[0], &i, sizeof(int));
            contaScritti+=4;

            if(contaScritti<=400)
                  kill(pid[i], SIGUSR1);
      }

      /* invio il segnale di uscita a tutti i figli */
      for(i=0; i<N; i++)
            kill(pid[i], SIGUSR2);

      for (i=0; i<N; i++)
      {
            pid[i]=wait(&status);
            printf("Il figlio con PID %d e' ritornato con valore %d\n", pid[i],
            (status>>8)&0xFF);
      }

}
```