SISTEMI OPERATIVI e SISTEMI OPERATIVI E LAB.
(A.A. 23-24) – 10 LUGLIO 2024

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 un numero variabile di parametri **Q+1** (con **Q** maggiore o uguale a 2): il primo parametro deve essere considerato un numero intero strettamente positivo (**X**) e strettamente minore di **20**, mentre gli altri **Q** devono essere **nomi assoluti di direttori** che identificano **Q** gerarchie (**G1, G2, …**) all'interno del file system. Il comportamento atteso dal programma, dopo il controllo dei parametri, è organizzato in **Q** fasi, una per ogni gerarchia.
Il programma, per ognuna delle **Q** fasi, deve esplorare la gerarchia **G** corrispondente - tramite un file comandi ricorsivo, **FCR.sh** - e deve cercare nella gerarchia **G** tutte le directory che contengono **almeno un** file *leggibile* con lunghezza in linee maggiore o uguale a **X**, ma strettamente minore di **255**: si riporti il nome assoluto di tali directory sullo standard output. Al termine di tutte le **Q** fasi, si deve riportare il numero **N** di file trovati e poi si deve invocare la parte in C passando come parametri i nomi assoluti dei file trovati intervallati dal numero corrispondente alla lunghezza in linee dei file trovati (perciò i parametri saranno: **F1**, **L1**, **F2**, **L2**, ... **FN, LN**).

NOTA BENE NEI DUE FILE COMANDI SI USI OBBLIGATORIAMENTE:
- una variabile di nome **X** per contenere il primo parametro di FCP.sh;
- una variabile di nome **G** per le singole gerarchie di ognuna delle **Q** fasi;
- il nome **/tmp/nomiAssoluti** per il file temporaneo;
- una variabile di nome **F** per identificare, via via, i singoli file delle directory esplorate;
- una variabile di nome **N** per contare i file che soddisfano la specifica.

La parte in C accetta un numero variabile *pari* **2*N** di parametri maggiore o uguale a 4 (*da controllare*) che rappresentano **N** nomi assoluti di file **F1**, **F2**, ... **FN** intervallati da numeri interi strettamente positivi **L1, L2**, ... **LN** (si può supporre che i parametri di posizione pari siano numeri e si deve *solo controllare che siano strettamente positivi*). Il processo padre deve generare **N** processi figli: i processi figli **Pn** sono associati ad uno dei file **F1**, ... **FN** e al corrispondente numero **L1, L2**, ... **LN** (*in ordine*). Ognuno di tali figli deve, per prima cosa, inizializzare il seme per la generazione random di numeri (*come illustrato nel retro del foglio*), quindi deve, usando in modo opportuno la funzione mia_random() (*riportata sul retro del foglio*), individuare un intero **r** che rappresenterà il numero della linea del file ad esso associato♣ da trovare, a partire dall'inizio del file.
I processi figli **Pn devono usare uno schema di comunicazione a pipeline**: il figlio **P0** comunica con il figlio **P1** che comunica con il figlio **P2** etc. fino al figlio **PN-1** che comunica con il **padre**. Questo schema a pipeline deve prevedere l'invio in avanti di un array **tutteLinee** di grandezza **N** e in cui ogni elemento dell'array corrisponde **alla linea selezionata**♦ dal corrispondente processo figlio **Pn**. Quindi, il generico processo **Pn**, subito dopo aver trovato la linea individuata dal numero **r**, deve ricevere dal figlio precedente (a parte il processo **P0**) l'array di linee **tutteLinee** e, dopo aver inserito la propria linea nella posizione giusta dell'array, deve inviarlo al figlio successivo, con **PN-1** che manda al padre. Quindi al padre deve arrivare l'array **tutteLinee** di grandezza **N** e in cui ogni elemento dell'array corrisponde **alla linea trovata dai figli Pn**: il padre deve scrivere le linee trovate dai figli sullo standard output *corredate dalle stampe indicate nel retro del foglio*.
Al termine, ogni processo figlio **Pn** deve ritornare al padre il valore random **r** (minore di 255, garantito dalla parte SHELL) e il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

NOTA BENE NEL FILE C main.c SI USI OBBLIGATORIAMENTE:
- una variabile di nome **N** per il numero di file;
- una variabile di nome **n** per l'indice dei processi figli;
- una variabile di nome **L** per la lunghezza in linee dei file;
- una variabile di nome **linea** per la linea selezionata dai figli dal proprio file;
- una variabile di nome **tutteLinee** per l'array con tutte le linee lette correntemente dai figli.

♣ **Si può supporre che l'ultima linea di tutti i file abbia sempre il terminatore di linea. Inoltre, si consideri che la prima linea dei file abbia numero 1 e che ogni linea sia al massimo lunga 250 caratteri compreso il terminatore di linea!**

♦ **Si suggerisce di usare un typedef per definire un tipo array di 250 char per le linee che dovranno far parte dell'array di linee!**

IMPORTANTE:

SEGUIRE TUTTE LE REGOLE FORNITE PRIMA DELLO SVOLGIMENTO DELL'ESAME!

Chiamata alla funzione di libreria per inizializzare il seme:

```
#include <time.h>

srand(time(NULL));
```

Funzione che calcola un numero random compreso fra 1 e n:

```
#include <stdlib.h>

int mia_random(int n)
{
int casuale;
        casuale = rand() % n;
        casuale++;
        return casuale;
}
```

ESEMPIO DI OUTPUT (solo per la parte rilevante citata nel testo) RIFERITO AL CASO DI DUE FILE E QUINDI DUE FIGLI:

```
Il padre ha ricevuto le seguenti linee dai figli:
Linea ricevuta dal figlio di indice 0:
servo come prova per l'esame del 10 Luglio 2024 (seconda linea di F1)
Linea ricevuta dal figlio di indice 1:
Questa e' la ottava linea di F2.
```

N.B. Le parti sottolineate sono quelle scritte dal padre a corredo delle linee trovate dai figli, che sono indicate in corsivo!