SISTEMI OPERATIVI E LAB.
(A.A. 22-23) – 12 LUGLIO 2023

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

Esercizio

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

TESTO PARTE SHELL: ATTENZIONE LEGGERE ANCHE LA NOTA SEGUENTE AL TESTO!

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** (**X**) **strettamente positivo e strettamente minore di 200**, mentre gli altri **Q** devono essere **nomi assoluti di directory** 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 contare **globalmente** tutti i file *leggibili* **F** *aventi estensione* **'.txt'** la cui lunghezza in linee sia strettamente minore di **X; contestualmente**[*] **si deve anche calcolare la lunghezza (in linee) massima L dei file trovati**. Al termine di tutte le **Q** fasi, si deve riportare sullo standard output il numero totale di tali file trovati globalmente e la lunghezza (in linee) massima **L**. Quindi, solo nel caso siano stati trovati *almeno* due file, si deve invocare la parte in C, passando come parametri **L** e i nomi dei file trovati (**F1, F2, ...**).

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;
- la stringa **/tmp/nomiAssoluti** per la parte iniziale dei nomi dei file temporanei;
- una variabile di nome **L** per la lunghezza in linee massima;
- una variabile di nome **F** per identificare, via via, i singoli file delle directory esplorate.

OSSERVAZIONE: se per provare la parte shell, si commenta la chiamata alla parte C, ricordarsi di togliere il commento prima della consegna!

TESTO PARTE C: ATTENZIONE LEGGERE ANCHE LA NOTA SEGUENTE AL TESTO!

La parte in C accetta un numero variabile di parametri **N+1** (con **N** maggiore o uguale a **2**): il primo parametro rappresenta un numero intero strettamente positivo (**L**), mentre gli altri **N** rappresentano nomi di file (**F1, …FN**): si assuma (senza bisogno di effettuare alcun controllo) che il formato di tali file sia tale che abbiano lunghezza in linee minore o uguale a **L**.
Il processo padre deve, per prima cosa, inizializzare il seme per la generazione random di numeri (come illustrato nel retro del foglio), quindi, deve generare **N processi figli**: ogni processo figlio **Pn** è associato ad uno dei file **F1, …FN** (*in ordine*). Ognuno di tali processi figli **Pn** esegue concorrentemente e, per prima cosa, deve calcolare la lunghezza in linee del proprio file associato. Quindi, i processi figli **Pn** e il processo padre devono attenersi a questo 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 una singola struttura dati, che deve contenere due campi: 1) *c1*, di tipo *int*, che deve contenere il valore massimo di linee calcolato dal processo **Pn**; 2) *c2*, di tipo *int*, che deve contenere l'indice d'ordine (**n**) del processo che ha calcolato il massimo (come visto in altri testi di esame) e perciò, al processo padre deve arrivare una singola struttura: il padre deve riportare i dati di tale struttura su standard output insieme al nome del file per cui sono stati calcolati (inserendo opportune spiegazioni per l'utente).
Esaurita questa fase a pipeline, ogni processo figlio **Pn** deve aspettare un numero di linea inviato dal padre (secondo quanto di seguito specificato) e, nel caso tale numero sia ammissibile per la lunghezza in linee del proprio file associato, deve stampare su standard output: il suo indice d'ordine, il suo PID, il numero di linea inviato dal padre, il nome del proprio file associato e quindi la linea selezionata; in caso contrario, dovrà stampare una indicazione di NON ammissibilità. Il padre, dopo aver verificato che il numero **L** sia uguale al campo *c1* della struttura ricevuta, deve applicare su tale valore la funzione ***mia_random()*** (riportata nel retro del foglio) per individuare un intero (**r**) che rappresenterà un numero di linea che il padre deve comunicare **a tutti** i figli **Pn**: tale intero (**r**), se ammissibile, produrrà una stampa sullo standard output da parte dei figli, come spiegato precedentemente.
Al termine dell'esecuzione, ogni figlio **Pn** ritorna al padre il valore **0** se il numero ricevuto era ammissibile per il proprio file associato (*semantica di successo di UNIX*), altrimenti il valore **1**; 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 processi figli;
- una variabile di nome **L** per il numero passato come primo parametro;
- una variabile di nome **n** per l'indice dei processi figli;
- una variabile di nome **linea per l'array di 250 caratteri** per le linee lette dai figli dal proprio file; *si supponga che 250 caratteri siano sufficienti per ogni linea, compreso il terminatore di linea/stringa.*
- una variabile di nome **r** per il valore random calcolato dal padre.

[*] Questo vuole dire che il calcolo della lunghezza massima va effettuato mentre si stanno cercando i file!

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:

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