SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 16-17) – 14 FEBBRAIO 2018

IMPORTANTE:

LEGGERE LE INFORMAZIONI SUL RETRO DEL FOGLIO!!!

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

La parte in C accetta un numero variabile **N+1** di parametri (con **N** maggiore o uguale a 2**)** che rappresentano i primi **N** nomi di file (**F1, ... FN**), mentre l'ultimo rappresenta un numero intero (**H**) strettamente positivo e minore di 255 (*da controllare*) che indica la lunghezza in linee dei file: infatti, la lunghezza in linee dei file è la stessa (questo viene garantito dalla parte shell e NON deve essere controllato).
Il processo padre deve, per prima cosa, inizializzare il seme per la generazione random di numeri (come illustrato nel retro del foglio). Il processo padre deve, quindi, generare **N processi figli** (**P0** … **PN-1**): i processi figli **Pi (con i che varia da 0 a N-1)** sono associati agli **N** file **Fk** (con **k= i+1**). Ogni processo figlio **Pi** deve leggere le linee del file associato **Fk sempre** fino alla fine. I processi figli **Pi** e il processo padre devono attenersi a questo **schema di comunicazione**: per ogni linea letta, il figlio **Pi** deve comunicare al padre la lunghezza (come int) della linea corrente compreso il terminatore di linea; il padre, per ogni linea, deve calcolare il **valore minimo** di lunghezza fra quelle inviate dai figli e, in seguito, usando in modo opportuno la funzione ***mia_random()*** (riportata nel retro del foglio), deve individuare un intero (**r**) che rappresenterà un indice per le linee correnti dei figli; il padre deve comunicare (per ogni linea) indietro a tutti i figli **Pi** tale indice: ogni figlio **Pi** ricevuto, per ogni linea, l'indice **r** deve stampare su standard output il carattere della linea corrente, corrispondente a tale indice, insieme con l'indice **r** di linea inviato dal padre, il suo indice di processo **i**, il suo **PID**, il numero di linea corrente e il nome del file associato.
Al termine, ogni processo figlio **Pi** deve ritornare al padre il valore intero corrispondente al numero di caratteri scritti su standard output (sicuramente minore di 255); il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

IMPORTANTE:

1) Fare il login sui sistemi in modalità Linux usando il proprio **USERNAME** e **PASSWORD**, aprire un browser sulla pagina ftp://lica02.lab.unimo.it/README, copiare il comando presente in un terminale ed eseguirlo rispondendo alle domande proposte: sul Desktop, viene creata automaticamente una directory **studente_1_1_USERNAME** al cui interno viene creato un file denominato student_data.csv che non va eliminato; infine , dopo avere copiato i propri file da chiavetta, passare in modalità testuale.
2) I file prodotti devono essere collocati nella directory **studente_1_1_USERNAME** dato che tale directory viene zippata e salvata automaticament sul server ad intervalli di tempo regolari. **ALLA SCADENZA DEL TEMPO A DISPOSIZIONE VERRÀ ATTIVATA UNA PROCEDURA AUTOMATICA DI ESTRAZIONE, PER OGNI STUDENTE DEL TURNO, DEI FILE CONTENUTI NELLA DIRETTORY SPECIFICATA.**
3) **NOVITÀ DALL'APPELLO DI LUGLIO 2016**: per facilitare le operazioni di stampa dei compiti sono imposte le seguenti regole per nominare i file da salvare nella directory **studente_1_1_USERNAME**:
   - FCP.sh per il file che contiene lo script principale (quello di partenza) della parte SHELL;
   - FCR.sh per il file che contiene lo script ricorsivo della parte SHELL;
   - main.c per il file che contiene il programma della parte C;
   - makefile per il file che contiene le direttive per il comando make.

   Devono essere rispettati esattamente i nomi indicati altrimenti NON si procederà alla correzione del compito!
4) NON devono essere presenti altri file con nome che termina con .sh o con .c nella directory **studente_1_1_USERNAME.**
5) Il tempo a disposizione per la prova è di **120 MINUTI** per il compito completo e di **90 MINUTI** per lo svolgimento della sola parte C.
6) Non è ammesso **nessun tipo di scambio di informazioni** né verbale né elettronico, pena la invalidazione della verifica: **all'ingresso deve essere lasciato il/i cellulare/i sulla cattedra e potranno essere ripresi solo all'uscita**.
7) L'assenza di commenti significativi verrà penalizzata, così come la mancanza del makefile!
8) **AL TERMINE DELLA PROVA È INDISPENSABILE CONSEGNARE IL TESTO DEL COMPITO (ANCHE IN CASO UNO STUDENTE SI RITIRI): IN CASO CONTRARIO, NON POTRÀ ESSERE EFFETTUATA LA CORREZIONE DEL COMPITO MANCANDO IL TESTO DI RIFERIMENTO.**
9) SI RICORDA CHE IN CASO DI ESITO INSUFFICIENTE è necessario visionare il compito prima di potersi iscrivere a qualunque appello successivo!

Chiamata alla funzione di libreria per inizializzare il seme:

```
#include <time.h>

srand(time(NULL));
```

Funzione che calcola un numero random compreso fra 0 e n-1:

```
#include <stdlib.h>

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