SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 16-17) – 12 LUGLIO 2017

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 **W+1** (con **W** maggiore o uguale a 2): il primo parametro deve essere considerato un numero intero strettamente positivo (**L**) e strettamente minore di 100, mentre gli altri **W** devono essere **nomi assoluti di direttori** che identificano **W** gerarchie (**G1, G2, …**) all'interno del file system. Il comportamento atteso dal programma, dopo il controllo dei parametri, è organizzato in **W** fasi, una per ogni gerarchia.
Il programma, per ognuna delle **W** 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 maggiore o uguale a **L**, ma strettamente minore di 255: si riporti il nome assoluto di tali direttori sullo standard output. Al termine di tutte le **W** fasi, 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**, **X1**, **F2**, **X2**, ... **FN, XN**).

La parte in C accetta un numero variabile pari **2N** di parametri maggiore o uguale a 2 (*da controllare*) che rappresentano **N** nomi assoluti di file **F1**, ... **FN** intervallati da numeri interi strettamente positivi **X1, X2**, ... **XN** che rappresentano la lunghezza in linee dei file (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 **Pi** sono associati agli **N** file **Fh** e al numero **Xh** (con h= i+1). Ognuno di tali figli deve creare a sua volta un processo nipote **PPi**: ogni processo nipote **PPi** esegue concorrentemente e 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 che rappresenterà il numero di linee del file **Fh** da selezionare, a partire dall'inizio del file, e inviare al figlio usando in modo opportuno il comando ***head*** di UNIX/Linux**.**
Ogni processo figlio **Pi** deve ricevere tutte le linee inviate dal suo processo nipote **PPi (ogni linea si può supporre che abbia una lunghezza massima di 250 caratteri, compreso il terminatore di linea e, se serve, il terminatore di stringa)** e, per ogni linea ricevuta, deve inviare al processo padre una **struttura** dati, che deve contenere tre campi: 1) *c1*, di tipo *int*, che deve contenere il pid del nipote; *c2*, di tipo *int*, che deve contenere il numero della linea; 3) c3, di tipo char[250], che deve contenere la linea corrente ricevuta dal nipote.
Il padre deve ricevere le strutture inviate dai figli nel seguente ordine: prima deve ricevere dal figlio **P0** la prima struttura inviata, poi deve ricevere dal figlio **P1** la prima struttura inviata e così via fino a che deve ricevere dal figlio **PN-1** la prima struttura inviata; quindi deve procedere a ricevere le seconde strutture inviate dai figli (se esistono) e così via. La ricezione di strutture da parte del padre deve terminare quando ha ricevuto tutte le strutture inviate da tutti i figli **Pi**. Il padre deve stampare su standard output, per ogni struttura ricevuta, ognuno dei campi.
Al termine, ogni processo figlio **Pi** deve ritornare al padre il valore random calcolato dal proprio processo nipote **PPi** e 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_2_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_2_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_2_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_2_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 1 e n:

```
#include <stdlib.h>

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