SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 15-16) – 13 LUGLIO 2016

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 **2 parametri**: il primo deve essere il nome assoluto di un direttorio che identifica una gerarchia (**G**) all'interno del file system, mentre il secondo parametro deve essere considerato un numero intero strettamente positivo e minore o uguale a 255 (**H**). Il programma deve cercare nella gerarchia **G** specificata, tutti i direttori che contengono almeno *un* file che abbia una lunghezza in byte minore o uguale a **H**. Si riporti il nome assoluto di tali direttori sullo standard output. In ogni direttorio trovato, si deve invocare la parte in C, passando come parametri i nomi dei file trovati (**F1, F2, ... FN-1**) **alternati** con la loro lunghezza in byte.

La parte in C accetta un numero variabile 2**N** di parametri (con 2**N** *maggiore o uguale a 2 e pari*[1]*)* che rappresentano **alternati** nomi di file (**F1,F2,... FN**) e lunghezza in byte del corrispondente file: infatti la lunghezza in byte dei file è fornita ed è sempre minore o uguale a 255 (questo viene garantito dalla parte shell e NON deve essere controllato).
Il processo padre deve 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= 2***i**+1). Ogni processo figlio **Pi** deve leggere i caratteri dal file secondo le indicazioni del padre (vedi dopo) fino alla fine. I processi figli **Pi** e il processo padre devono attenersi a questo **schema di comunicazione a due fasi**. *Nella prima fase*, il processo padre deve richiedere all'utente per ogni file **Fk** (cioè per ogni processo figlio **Pi**) un divisore della lunghezza in byte del corrispondente file **Fk** e, dopo averne controllato la correttezza, lo deve comunicare al corrispondente processo figlio **Pi**. *Nella seconda fase*, i processi figli **Pi** devono comunicare al padre i caratteri di **posizione multipla** del proprio divisore ricevuto dal padre[2]. Il padre deve ricevere i caratteri dai figli nel seguente ordine: prima deve ricevere dal figlio **P0** il primo carattere inviato, poi deve ricevere dal figlio **P1** il primo carattere inviato e così via fino a che deve ricevere dal figlio **PN-1** il primo carattere inviato; quindi deve procedere a ricevere i secondi caratteri inviati dai figli (se esistono) e così via. La ricezione di caratteri da parte del padre deve terminare quando ha ricevuto tutti i caratteri inviati da tutti i figli **Pi. Per ogni carattere ricevuto, il padre deve riportare sullo standard output: l'indice del figlio che gli ha inviato il carattere, il nome del file da cui è stato letto il carattere, la posizione all'interno di tale file del carattere ricevuto (usando in modo opportuno i divisori forniti dall'utente) e il carattere ricevuto.**
Al termine, ogni processo figlio **Pi** deve ritornare al padre il valore intero corrispondente al numero di caratteri inviati al padre (sicuramente minore o uguale a 255); il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

LE NOTE SONO STATE AGGIUNTE PER RENDERE PIÙ CHIARO IL TESTO (LA PRIMA È STATA DATA COME RISPOSTA A CHI DURANTE IL COMPITO LO HA CHIESTO; LA SECONDA DERIVA DALLA CORREZIONE DEI COMPITI).

---

[1] Chiaramente anche se non era scritto in esplicito (dimenticanza!) andava controllato che il numero dei parametri fosse >= 2 e pari!

[2] I figli dovevano comunicare al padre ogni carattere appena letto, altrimenti la soluzione non sarebbe stata molto concorrente!

COGNOME: NOME:
TURNO: 01 POS: 01

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_01_01_XXX** 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_01_01_XXX** 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_01_01_XXX**:
   - 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) Il tempo a disposizione per la prova è di **120 MINUTI** per lo svolgimento di tutto il compito e di **90 MINUTI** per lo svolgimento della sola parte C.
5) Non è ammesso **nessun tipo di scambio di informazioni** né verbale né elettronico, pena la invalidazione della verifica.
6) L'assenza di commenti significativi verrà penalizzata, così come la mancanza del makefile!
7) **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.**
8) NON devono essere presenti altri file con nome che termina con .sh o con .c nella directory **studente_01_01_XXX.**