SISTEMI OPERATIVI E LAB.
(A.A. 17-18) – 12 SETTEMBRE 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 **W+2** (con **W** maggiore o uguale a 2): i primi due parametri devono essere considerati numeri interi strettamente positivi (**H** e **K**) con **H** strettamente minore di **K**, mentre gli altri **W** devono essere **nomi assoluti di directory** 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 tutti i direttori che contengono un **numero** di file leggibili (**F1, ... FN**) compreso fra **H** e **K**. Si riporti il nome assoluto di tali direttori sullo standard output. In ognuno di tali direttori trovati, si deve invocare la parte in C, passando come parametri i nomi dei file trovati (**F1, ... FN**) che soddisfano la condizione precedente.

La parte in C accetta un numero variabile **N** di parametri (con **N** maggiore o uguale a **2**, da controllare**)** che rappresentano **N** nomi di file (**F1, F2. ... FN**).
Il processo padre deve generare **N processi figli Pi** (**P0** … **PN-1**): i processi figli **Pi (con i che varia da 0 a N-1)** sono associati agli **N** file **Ff** (con f= i+1). Ognuno di tali processi figli deve creare a sua volta un **processo nipote PPi** (**PP0 … PPN-1**) associato sempre al corrispondente file **Ff**. Ogni processo figlio **Pi** e ogni nipote **PPi** esegue concorrentemente andando a cercare nel file associato **Ff** tutte le occorrenze dei caratteri **numerici** per il figlio e tutte le occorrenze dei caratteri **alfabetici minuscoli** per il nipote. Ognuno dei processi figlio e nipote deve operare una modifica del file **Ff:** in specifico, ogni *nipote* deve trasformare ogni carattere alfabetico minuscolo nel corrispondente carattere alfabetico maiuscolo, mentre ogni *figlio* deve trasformare ogni carattere numerico nel carattere spazio. Una volta terminate le trasformazioni, sia i processi figli **Pi** che i processi nipoti **PPi** devono comunicare al padre il numero (in termini di ***long int***) di trasformazioni effettuate. Il padre ha il compito di stampare su standard output, rispettando l'ordine dei file, il numero di trasformazioni ricevute da ogni figlio **Pi** e da ogni nipote **PPi**, riportando opportuni commenti esplicativi, che devono includere anche il nome del file che è stato interessato dalle trasformazioni.
Al termine, ogni processo nipote **PPi** deve ritornare al figlio **Pi** un opportuno codice ed analogamente ogni processo figlio **Pi** deve ritornare al padre un opportuno codice; il codice che ogni nipote **PPi** e ogni figlio **Pi** deve ritornare è:

- **a)** **0** se il numero di trasformazioni attuate è minore di 256;
- **b)** **1** se il numero di trasformazioni attuate è maggiore o uguale a 256, ma minore di 512;
- **c)** **2** se il numero di trasformazioni attuate è maggiore o uguale a 512, ma minore di 768;
- **d)** etc.

Sia ogni figlio **Pi** e sia il padre devono stampare su standard output il PID di ogni nipote/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_2_1_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_2_1_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_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!