SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 16-17) – 14 GIUGNO 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 tre parametri: il primo parametro deve essere il **nome assoluto di un direttorio** che identifica una gerarchia (**G**) all'interno del file system, il secondo parametro deve essere considerato un numero intero strettamente positivo (**K**), mentre il terzo parametro deve essere considerato un singolo carattere (**Cx**). Il programma deve cercare nella gerarchia **G** specificata tutti i direttori che contengono almeno **un** file leggibile e scrivibile **Fi** la cui lunghezza in byte sia uguale a **K *e*** che contenga (nel contenuto) almeno una occorrenza del carattere **Cx**. Si riporti il nome assoluto di tali direttori sullo standard output. Al termine dell'intera esplorazione ricorsiva di **G**, si deve invocare la parte C passando come parametri tutti i ***nomi assoluti*** dei file **Fi** trovati e **Cx**.

La parte in C accetta un numero variabile **N+1** di parametri (con **N** maggiore o uguale a 1, da controllare**)** che rappresentano i primi **N** nomi assoluti di file (**F1, F2, ... FN**) mentre l'ultimo rappresenta un singolo carattere (**Cx**) (da controllare): si può ipotizzare che la lunghezza di tutti i file sia uguale (senza verificarlo). Il processo padre deve generare **N processi figli** (**P0, P1, … 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 i caratteri del file associato **FK** cercando le occorrenze del carattere **Cx**, sostituendole eventualmente con i caratteri inviati dal padre. Ogni figlio **Pi**, per ogni occorrenza trovata, deve comunicare al padre la posizione (in termini di *long int*) di tale occorrenza a partire dall'inizio del file[1]. Il padre deve ricevere le posizioni (come *long int*) inviate dai figli nel seguente ordine: prima deve ricevere dal figlio **P0** la prima posizione inviata, poi deve ricevere dal figlio **P1** la prima posizione inviata e così via fino a che deve ricevere dal figlio **PN-1** la prima posizione inviata; quindi deve procedere a ricevere le seconde posizioni inviate dai figli (se esistono) e così via. La ricezione di posizioni da parte del padre deve terminare quando ha ricevuto tutte le posizioni inviate da tutti i figli **Pi**. Per ogni posizione ricevuta, il padre deve riportare sullo standard output: l'indice del figlio che gli ha inviato la posizione, il nome del file in cui è stata trovata l'occorrenza del carattere **Cx** e (naturalmente) la posizione ricevuta. Quindi, per ogni posizione ricevuta, il padre deve chiedere all'utente il carattere con cui deve essere sostituita la specifica occorrenza; nel caso l'utente inserisca una linea vuota, questo deve essere interpretato dal padre come indicazione di non sostiture l'occorrenza corrente. Il padre, per ogni posizione, deve comunicare al figlio corrispondente o il carattere da sostituire oppure se può proseguire con la ricerca di altre occorrenze del carattere **Cx**.
Al termine, ogni processo figlio **Pi** deve ritornare al padre il numero di sostituzioni effettuate nel proprio file (supposto strettamente minore di 255); il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

[1] Si precisi, come commento, nel codice se il primo carattere del file viene considerato in posizione 0 o in posizione 1.

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!