SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 14-15) – 22 GENNAIO 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 deve essere considerato un numero intero strettamente positivo maggiore o uguale a 2 (**N**). Il programma deve cercare nella gerarchia **G** specificata tutti i direttori che contengono esattamente **2N** file leggibili e nessun sotto-direttorio. 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 **2N** file che soddisfano la condizione (**F1, F2, ... F2N**).

La parte in C accetta un numero *variabile* e *pari* **2N** di parametri maggiore o uguale a 4 (*da controllare che il numero di parametri sia pari e sia* ≥ 4) che rappresentano nomi di file **F1**, **F2**, ... **FN e FN+1, FN+2**, ... **F2N**. 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 ai primi **N** file **Fj** (con j= i+1). Ogni figlio **Fi** deve generare a sua volta un processo nipote **PPi**: i processi nipoti sono associato ai secondi **N** file **Fk** (con k=i+1+N). I processi figli **Pi** e nipoti **PPi** eseguono concorrentemente. In particolare, ogni figlio **Pi** deve leggere dal proprio file associato **Fj** un carattere alla volta e, non appena trova il primo carattere alfabetico maiuscolo (**AM**), deve comunicarlo al proprio processo nipote **PPi**. Il processo figlio **Pi** deve poi proseguire calcolando il numero di occorrenze **Pocc** (in termini di long int) di **AM** nel proprio file associato **Fj**. Ogni processo nipote **PPi**, una volta ricevuto il carattere **AM** dal processo figlio **Pi**, deve calcolare a sua volta il numero di occorrenze **Nocc** (in termini di long int) di **AM** nel proprio file associato **Fk**; successivamente, ogni processo nipote **PPi** deve comunicare al proprio processo figlio **Pi** il valore di **Nocc**; il processo figlio **Pi**, dopo avere calcolato **Pocc**, riceve l'informazione **Nocc** inviata dal proprio nipote **PPi** e la confronta con la propria **Pocc**. Quindi, ogni processo figlio **Pi** deve comunicare al padre una struttura dati che deve contenere quattro campi: 1) *chr*, di tipo char, che deve contenere il carattere **AM**; 2) *occ*, di tipo long int, che deve contenere il **massimo** fra **Pocc** e **Nocc**; 3) *proc*, di tipo char, che deve contenere 'F' per figlio o 'N' per nipote a seconda di quale dei due processi ha trovato il massimo numero di occorrenze *occ*; 4) *pid*, di tipo int, che deve contenere il PID del processo che ha calcolato *occ*. Il padre deve ricevere le **N** strutture inviate dagli **N** figli **Pi** secondo l'ordine di creazione dei figli e ha il compito di stampare su standard output in modo significativo tutti i campi delle strutture ricevute aggiungendo l'indicazione del file cui si riferiscono (usando in modo opportuno l'informazione 'F' e 'N').
Al termine, i processi nipoti **PPi** e i processi figli **Pi** devono ritornare (al processo corretto) il valore di successo o insuccesso riferito alla lettura dell'ultimo carattere del file associato, facendo riferimento alla normale semantica UNIX; il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato da ogni figlio.

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_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_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) 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.
4) 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**.
5) L'assenza di commenti significativi verrà penalizzata, così come la mancanza del makefile!
6) **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.**
7) SI RICORDA CHE IN CASO DI ESITO INSUFFICIENTE è necessario visionare il compito prima di potersi iscrivere a qualunque appello successivo!