SISTEMI OPERATIVI E LAB.
(A.A. 13-14) – 16 LUGLIO 2014

IMPORTANTE:

1) Fare il login sui sistemi in modalità Linux usando il proprio **username** e **password**, attivare syncexam.sh e passare in modalità testuale.
2) I file prodotti devono essere collocati in un **sottodirettorio** (che deve essere nella directory studente_XXX) che deve essere creato e avere nome **ESAME16Lug14-1-01.** FARE ATTENZIONE AL NOME DEL DIRETTORIO, in particolare alle maiuscole e ai trattini indicati. Verrà penalizzata l'assenza del direttorio con il nome indicato e/o l'assenza dei file nel direttorio specificato, al momento della copia automatica del direttorio e dei file. **ALLA SCADENZA DEL TEMPO A DISPOSIZIONE VERRÀ INFATTI ATTIVATA UNA PROCEDURA AUTOMATICA DI COPIA, PER OGNI STUDENTE DEL TURNO, DEI FILE CONTENUTI NEL DIRETTORIO SPECIFICATO.**
3) Il tempo a disposizione per la prova è di **75 MINUTI** per lo svolgimento della sola parte C e di **120 MINUTI** per lo svolgimento di tutto il compito.
4) Non è ammesso **nessun tipo di scambio di informazioni** né verbale né elettronico, pena la invalidazione della verifica.
5) **L'assenza di commenti significativi (per la parte in C, scritti in C standard) 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.**

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 due parametri: il primo 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 **X** strettamente positivo. Il programma deve cercare nella gerarchia **G** specificata tutti i direttori che contengono almeno **un** file leggibile con lunghezza in linee pari **X** e il cui contenuto del file sia tale per cui tutte le sue linee iniziano con un carattere alfabetico minuscolo. Si riporti il nome assoluto di tali direttori sullo standard output e quindi i nomi di tutti i file trovati (**F1, F2, … FN**). Quindi, chiesta conferma all'utente, in ogni direttorio trovato si deve invocare la parte in C, passando come parametri i **nomi dei file trovati** (**F1, F2, … FN**) e la loro lunghezza in linee **X**.

La parte in C accetta un numero *variabile* di parametri **N+1** (**maggiore o uguale a 2,** *da controllare*) che rappresentano i primi **N** i nomi assoluti di file **F1...FN** e l'ultimo la lunghezza in linee dei file (**X**, *da non controllare*). Il processo padre deve generare **N processi figli** (**P0 … PN-1**): ogni processo figlio è associato al corrispondente file **Fi**. Ognuno di tali processi figli esegue concorrentemente, leggendo tutte le **X** linee del file associato **Fi**: **per ogni linea letta,** il figlio **Pi** deve selezionare il primo carattere e quindi comunicare questa informazione. In particolare, i processi figli e il processo padre devono attenersi ad uno ***schema di comunicazione a pipeline***; il figlio **P0** comunica con il figlio **P1** etc. fino al figlio **PN-1** che comunica con il padre; questo schema a pipeline deve essere ripetuto per ogni linea di ogni file e deve prevedere l'invio in avanti di un array di caratteri (**primiCar**) di grandezza pari al numero di ~~linee dei~~ file; il primo figlio **P0**, dopo aver completato la lettura della linea corrente letta dal file associato **F1**, inserisce in **primiCar** al primo posto il primo carattere della linea corrente e quindi invia **primiCar** al figlio **P1**; quindi ogni figlio **Pi** (a parte P0) deve ricevere **primiCar** e quindi deve inserire, dopo aver completato la lettura della linea corrente letta dal file associato **Fi**, in **primiCar** al posto corretto il primo carattere della linea corrente e quindi invia avanti **primiCar** al figlio successivo, fino a **PN-1** che la invia al processo padre. Quindi, al processo padre devono arrivare tanti array **primiCar** quante sono le linee dei file letti dai processi figli **P0 … PN-1**. Il padre ha il compito di stampare su standard output tutte le informazioni ricevute dal processo **PN-1**: in particolare, deve essere stampato -per ogni linea- un contatore di linea e quindi i primi caratteri letti da ogni processo figlio riportando anche -per ogni carattere- l'indice del processo che ha effettuato la lettura e il nome del file da cui si è letto tale carattere.
Al termine, ogni processo figlio **Pi** deve ritornare al padre l'ultimo carattere selezionato dal proprio file associato **Fi** e il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.