SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 15-16) – 9 SETTEMBRE 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 un numero variabile di parametri **N** (**N** maggiore o uguale a 2) che devono essere **nomi assoluti di direttori** che identificano **N** gerarchie (**G1, G2, …**) all'interno del file system. Il comportamento atteso dal programma, dopo il controllo dei parametri, è organizzato in **N** fasi, una per ogni gerarchia.
Il programma, per ognuna delle **N** fasi, deve esplorare la gerarchia **Gh** corrispondente e deve cercare tutti i direttori che contengono solo file che presentano al loro interno esclusivamente caratteri alfabetici minuscolo (N.B. possono essere presenti sottodirettori)**.** Si riporti il nome assoluto di tali direttori sullo standard output. Al termine di tutte le **N** fasi, si deve riportare sullo standard output il numero totale di file trovati globalmente che soddisfano la specifica precedente (*file trovati*) in tutte le gerarchie (**G1, G2, …**). Inoltre, per ogni *file trovato*, si deve riportare sullo standard output il suo **nome assoluto**, e quindi per ognuno di essi (*file corrente*), si deve invocare la parte in C passando come parametro il nome assoluto del *file corrente*.

La parte in C accetta un unico parametro che rappresenta il nome assoluto di un file (**F**) (senza bigogno di controlli sul fatto che sia assoluto).
Il processo padre deve generare **26 processi figli** (**P0, P1, … P25**) tanti quanti i caratteri dell'alfabeto inglese: tutti i processi figli **Pi** (con i che varia da 0 a 25) sono associati all'unico file **F** e ognuno dei processi figli è associato al carattere alfabetico ***minuscolo*** corrispondente **(P0** è associato al carattere ***'a'*** fino a **P25** che è associato al carattere ***'z'***). Ogni processo figlio **Pi** deve leggere i caratteri del file **F** cercando il carattere a lui associato **Ci** (per i=0, C0=***'a'***, … per ii=25, C25=***'z'***). I processi figli e il processo padre devono attenersi a questo **schema di comunicazione a pipeline**: il figlio **P0** comunica con il figlio **P1** che comunica con il figlio **P2** etc. fino al figlio **P25** che comunica con il **padre**. Questo schema a pipeline deve prevedere l'invio in avanti di un array di **strutture** dati ognuna delle quali deve contenere due campi: 1) *v1*, di tipo char, che deve contenere il carattere Ci; 2) *v2*, di tipo long int, che deve contenere il numero di occorrenze del carattere Ci, calcolate dal corrispondente processo. *Ogni array di strutture utilizzato dai figli e dal padre deve avere dimensione fissa (26 elementi!).* Quindi la comunicazione deve avvenire in particolare in questo modo: il figlio **P0** passa in avanti (cioè comunica) un array di strutture **A0** (di 26 elementi), che contiene una sola struttura significativa (nell'elemento di indice 0 dell'array A0) con *v1* uguale a ***'a'*** e con *v2* uguale al numero di occorrenze del carattere ***'a'*** trovate da **P0** nel file **F**; il figlio seguente **P1**, dopo aver calcolato numero di occorrenze del carattere associato **C1** nel file **F**, deve leggere (*con una singola read*) l'array **A0** inviato da **P0** e quindi deve confezionare l'array **A1** che corrisponde all'array **A0** aggiungendo nell'elemento di indice 1 la struttura con i propri dati e la passa (*con una singola write*) al figlio seguente **P2**, etc. fino al figlio **P25**, che si comporta in modo analogo, ma passa al **padre**. Quindi, al processo padre deve arrivare l'array **A25**. Il processo padre, dopo aver ordinato tale array A25 in senso crescente rispetto al campo v2, deve riportare i dati di ognuna delle **26** strutture su standard output insieme al **pid** e all'indice i del processo che ha generato tale struttura.
Al termine, ogni processo figlio **Pi** deve ritornare al padre l'ultimo carattere letto dal file **F**; il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato sia come carattere che come valore ASCII (in decimale).

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_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_1_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_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_1_1_XXX.**

SE PUÒ SERVIRE RIPORTO IL SEGUENTE CODICE dai Lucidi di Fondamenti II e Lab. - Algoritmi di ordinamento (pag. 5):

```
void bubbleSort(int v[], int dim)
{ int i; bool ordinato = false;
while (dim>1 && !ordinato)
        { ordinato = true; /* hp: è ordinato */
          for (i=0; i<dim-1; i++)
                  if (v[i]>v[i+1])
                  {
                  scambia(&v[i],&v[i+1]);
                  ordinato = false;
                  }
          dim--;
        }
}
```