SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 18-19) – 12 FEBBRAIO 2020

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): il primo parametro deve essere considerato un intero strettamente positivo ***dispari*** (**K1**), il secondo parametro deve essere considerato un numero intero strettamente positivo ***pari*** (K2), 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 trovare **globalmente** tutti i file che saranno cercati secondo quanto di seguito specificato. Il file comandi ricorsivo **FCR.sh** deve cercare in ogni gerarchia **Gg** tutti i direttori che contengono *esattamente* due file che abbiano dimensione in caratteri rispettivamente uguale a **K1** e uguale a **K2**: in altre parole, si devono cercare tutti i direttori che contengono uno e un solo file di dimensione **K1 e** uno e un solo file di dimensione **K2**. Si riporti il nome assoluto di tali direttori sullo standard output. Al termine di TUTTE le **W** fasi, si deve invocare la parte in C, passando come parametri i nomi assoluti dei file ***trovati*** (**F1**, **F2**, ... **FN**) avendo l'accortezza che i file siano in sequenza strettamente alternati fra file di dimensione **K2 (quindi in particolare F1 deve avere tale dimensione)** e file di dimensione **K1 (quindi in particolare F2 deve avere tale dimensione)**[*]**.**

La parte in C accetta un numero variabile **N** di parametri (con **N** maggiore o uguale a **2** e **pari**, da controllare**)** che rappresentano nomi di file **F1, F2. ... FN**. Il processo padre deve generare **N** processi figli (P0 ... PN-1): i processi figli **Pi** sono associati ai file **Fi+1** (con *i* che varia da 0 a N): i processi figli devono essere distinti, in base al loro numero d'ordine, in *processi pari* e in *processi dispari*. Ogni processo figlio deve leggere i caratteri del proprio file **Fi+1** e deve effettuare **due conteggi**, usando due variabili, entrambe di tipo *long int*:

a) ogni processo figlio **pari** deve contare quanti caratteri di posizione **pari**[°] nel file associato hanno **codice ASCII pari** (*primo conteggio*) e quanti hanno **codice ASCII dispari** (*secondo conteggio*);
b) ogni processo figlio **dispari** deve contare quanti caratteri di posizione **dispari** nel file associato hanno **codice ASCII dispari** (*primo conteggio*) e quanti hanno **codice ASCII pari** (*secondo conteggio*);

Completati i due conteggi, ogni processo figlio **Pi** deve comunicare al processo padre una **struttura** che deve contenere due campi, *c1* e *c2*, con *c1* uguale al primo conteggio ottenuto e con *c2* uguale al secondo conteggio ottenuto. Il processo padre deve ricevere, **rispettando l'ordine dei file**, *prima* tutte le strutture inviate dai figli pari e *poi* tutte le strutture inviate dai figli dispari e deve stampare su standard output sia il numero d'ordine dei processi figli, che i due campi *c1* e *c2* delle strutture ricevute.
Al termine, ogni processo figlio **Pi** deve ritornare al padre il **valore 0** (zero) se il primo conteggio è strettamente maggiore del secondo, altrimenti il **valore 1** (uno); il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

PER UN ESEMPIO DI ESECUZIONE SI CONSULTI IL RETRO DEL FOGLIO!

[*] **NOTA BENE**: Il numero di file trovati e quindi passati alla parte C sarà sempre un numero pari!
[°] **IMPORTANTE**: Si consideri per calcolare la posizione che il primo carattere di un file abbia posizione 0!

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.
   ATTENZIONE: DOPO AVER EVENTUALMENTE USATO LA/LE CHIAVETTA/E, QUESTA/E DEVE/ONO ESSERE PORTATA/E ALLA CATTEDRA E MESSA/E DI FIANCO AL PROPRIO CELLULARE!
2) I file prodotti devono essere collocati nella directory **studente_2_1_XXX** dato che tale directory viene zippata e salvata automaticamentw 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) Per facilitare le operazioni di stampa dei compiti sono imposte le seguenti regole per nominare i file da salvare nella directory **studente_2_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) **ATTENZIONE: IN PARTICOLARE PER LA PARTE C NON VERRANNO CORRETTE SOLUZIONI CHE PRESENTANO ERRORI RIPORTATI DAL COMANDO gcc ALL'INVOCAZIONE DEL COMANDO make!!!!**
4) NON devono essere presenti altri file con nome che termina con .sh o con .c nella directory **studente_1_1_USERNAME.**
6) 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.
7) 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**.
8) L'assenza di commenti significativi verrà penalizzata, così come la mancanza del makefile!
9) **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.**
10) SI RICORDA CHE IN CASO DI ESITO INSUFFICIENTE è necessario visionare il compito prima di potersi iscrivere a qualunque appello successivo!

ESEMPIO DI ESECUZIONE: Supponendo di avere due file di nome rispettivamente **a** (di lunghezza 8 caratteri, compreso il carattere terminatore di linea) e **z** (di lunghezza 7 caratteri, compreso il carattere terminatore di linea) con il seguente contenuto
soELab@lica04$ cat a
zaazaaz
soELab@lica04$ cat z
aaazzz

e ricordando che il carattere 'a' ha codice ASCII dispari, mentre il carattere 'z' ha codice ASCII pari, l'invocazione del main con parametri **a** e **z** (esattamente in questo ordine!) produce il seguente risultato:
Figlio con indice 0 ha calcolato: c1=2 e c2=2
Figlio con indice 1 ha calcolato: c1=1 e c2=2