SISTEMI OPERATIVI E LAB.
(A.A. 18-19) – 11 SETTEMBRE 2019

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 numero intero strettamente positivo (**H),** il secondo parametro deve essere considerato un numero intero strettamente maggiore di 1 e minore di **255** (**K**), 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 cercare tutti i direttori che contengono un **numero uguale a H** di file **di lunghezza in caratteri uguale K**. Si riporti il nome assoluto di tali direttori sullo standard output. In ognuno di tali direttori trovati, si deve invocare la parte in C **due volte**: la prima volta passando come parametri i file trovati (cioè che soddisfano la condizione precedente) **di posizione dispari** e la seconda volta i file trovati **di posizione pari**.

La parte in C accetta un numero variabile **pari N** di parametri (con **N pari e** maggiore o uguale a **2**, da controllare) che rappresentano **N** nomi di file (**F1, F2. ... FN**): la lunghezza in caratteri dei file è la stessa (**minore di 255**, questo viene garantito dalla parte shell e NON deve essere controllato).
Il processo padre deve generare **N processi figli Pi** (**P0** … **PN-1**): i processi figli **Pi (con i che varia da 0 a N-1)** sono associati agli **N** file **Ff** (con f= i+1). I figli si devono considerare a coppie ordinate: ogni coppia è costituita dal processo di indice pari **Pp** (i = 0, 2, ... **NM»-**2) e dal processo di indice dispari **Pd** (i = 1, 3, ... **NM»-**1). La comunicazione in ognuna di tali coppie deve essere dal processo **Pp** al processo **Pd**; la prima azione che dovrà compiere il processo **Pd** sarà quella di creare un file (**Fcreato**) il cui nome sia la concatenazione del nome del file associato **Ff** con la stringa "**.MAGGIORE**". Ogni processo figlio esegue concorrentemente leggendo i caratteri del file ad esso associato **Ff:** dopo la lettura di ogni carattere (**Cp**) da parte del processo **Pp**, questo viene comunicato al processo **Pd**, il quale lo riceve dopo aver letto il proprio carattere (**Cd**); il processo **Pd** deve confrontare il proprio carattere **Cd** con il carattere ricevuto **Cp** e *se* **Cd** risulta strettamente maggiore di **Cp**, lo scrive sul file **Fcreato**, *altrimenti* scrive **Cp**.
Ogni processo figlio deve ritornare al padre il numero di caratteri letti dal proprio file. Il padre, dopo che i figli sono terminati, deve stampare su standard output i PID di ogni figlio con il corrispondente valore ritornato.
Il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

SISTEMI OPERATIVI E LAB.
(A.A. 18-19) – 11 SETTEMBRE 2019

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 **Z+2** (con **Z** strettamente maggiore di 1): il primo parametro deve essere considerato un numero intero strettamente positivo (**X),** il secondo parametro deve essere considerato un numero intero strettamente maggiore di 1 e minore di **255** (**Y**), mentre gli altri **Z** devono essere **nomi assoluti di directory** che identificano **Z** gerarchie (**G1, G2, …**) all'interno del file system. Il comportamento atteso dal programma, dopo il controllo dei parametri, è organizzato in **Z** fasi, una per ogni gerarchia.
Il programma, per ognuna delle **Z** fasi, deve esplorare la gerarchia **Gg** corrispondente - tramite un file comandi ricorsivo, **FCR.sh** – e deve cercare tutti i direttori che contengono un **numero uguale a X** di file **di lunghezza in caratteri uguale Y**. Si riporti il nome assoluto di tali direttori sullo standard output. In ognuno di tali direttori trovati, si deve invocare la parte in C **due volte**: la prima volta passando come parametri i file trovati (cioè che soddisfano la condizione precedente) **di posizione dispari** e la seconda volta i file trovati **di posizione pari**.

La parte in C accetta un numero variabile **pari M** di parametri (con **M pari e** maggiore o uguale a **2**, da controllare) che rappresentano **M** nomi di file (**F1, F2. ... FM**): la lunghezza in caratteri dei file è la stessa (**minore di 255**, questo viene garantito dalla parte shell e NON deve essere controllato).
Il processo padre deve generare **M processi figli Pj** (**P0** … **PM-1**): i processi figli **Pj (con j che varia da 0 a M-1)** sono associati agli **M** file **Ff** (con f= j+1). I figli si devono considerare a coppie ordinate: ogni coppia è costituita dal processo di indice pari **Pp** (i = 0, 2, ... **NM»**-2) e dal processo di indice dispari **Pd** (i = 1, 3, ... **NM»**-1). La comunicazione in ognuna di tali coppie deve essere dal processo **Pd** al processo **Pp**; la prima azione che dovrà compiere il processo **Pp** sarà quella di creare un file (**Fcreato**) il cui nome sia la concatenazione del nome del file associato **Ff** con la stringa "**.MINORE**". Ogni processo figlio esegue concorrentemente leggendo i caratteri del file ad esso associato **Ff:** dopo la lettura di ogni carattere (**Cd**) da parte del processo **Pd**, questo viene comunicato al processo **Pp**, il quale lo riceve dopo aver letto il proprio carattere (**Cp**); il processo **Pp** deve confrontare il proprio carattere **Cp** con il carattere ricevuto **Cd** e *se* **Cp** risulta strettamente minore di **Cd**, lo scrive sul file **Fcreato**, *altrimenti* scrive **Cd**.
Ogni processo figlio deve ritornare al padre il numero di caratteri letti dal proprio file. Il padre, dopo che i figli sono terminati, deve stampare su standard output i PID di ogni figlio con il corrispondente valore ritornato.
Il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

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_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) **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!