SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 18-17) – 31 MAGGIO 2019
TESTO TURNI 1 e 2

IMPORTANTE:

LEGGERE LE INFORMAZIONI SUL RETRO DEL FOGLIO!!!

Esercizio

Si realizzi un programma **concorrente** per UNIX che deve avere una parte in **Bourne Shell** (già svolta) e una parte in **C**.

La parte in C accetta un numero variabile **N** di parametri maggiore o uguale a 3 (*da controllare*) che rappresentano nomi assoluti di file **F1**, ... **FN**. Il processo padre deve generare **N** processi figli: i processi figli **Pi** sono associati agli **N** file **Fh** (con h= i+1). Ognuno di tali figli deve creare a sua volta un processo nipote **PPi**: ogni processo nipote **PPi** esegue concorrentemente e deve ordinare il file **Fh** secondo il normale ordine alfabetico, senza differenziare maiuscole e minuscole, usando in modo opportuno il comando ***sort*** di UNIX/Linux**.**
Ogni processo figlio **Pi** deve ricevere solo la **PRIMA** linea inviata dal suo processo nipote **PPi** e deve inviare al processo padre una **struttura** dati, che deve contenere tre campi: 1) *c1*, di tipo *int*, che deve contenere il pid del nipote; *c2*, di tipo *int*, che deve contenere la lunghezza della linea compreso il terminatore di linea; 3) *c3*, di tipo char[250][*], che deve contenere la linea corrente ricevuta dal nipote.
Il padre deve ricevere, rispettando l'ordine dei file, le singole strutture inviate dai figli e deve stampare su standard output, per ogni struttura ricevuta, ognuno dei campi insieme al nome del file cui le informazioni si riferiscono: **si faccia attenzione al fatto che è demandato al padre il compito di trasformare, in una stringa, la linea ricevuta nel campo c3 di ogni struttura!**
Al termine, ogni processo figlio **Pi** deve ritornare al padre la lunghezza della linea inviata al padre, ma non compreso il terminatore di linea e il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

[*] **Ogni linea si può supporre che abbia una lunghezza massima di 250 caratteri, compreso il terminatore di linea e, se serve, il terminatore di stringa.**

SISTEMI OPERATIVI e
LABORATORIO DI SISTEMI OPERATIVI
(A.A. 18-17) – 31 MAGGIO 2019
TESTO TURNI 1 e 2

IMPORTANTE:

LEGGERE LE INFORMAZIONI SUL RETRO DEL FOGLIO!!!

Esercizio

Si realizzi un programma **concorrente** per UNIX che deve avere una parte in **Bourne Shell** (già svolta) e una parte in **C**.

La parte in C accetta un numero variabile **M** di parametri strettamente maggiore di 2 (*da controllare*) che rappresentano nomi assoluti di file **F1**, ... **FM**. Il processo padre deve generare **M** processi figli: i processi figli **Pj** sono associati agli **M** file **Fk** (con k= j+1). Ognuno di tali figli deve creare a sua volta un processo nipote **PPj**: ogni processo nipote **PPj** esegue concorrentemente e deve ordinare il file **Fk** in ordine alfabetico inverso, usando in modo opportuno il comando ***sort*** di UNIX/Linux**.**
Ogni processo figlio **Pj** deve ricevere solo la **PRIMA** linea inviata dal suo processo nipote **PPj** e deve inviare al processo padre una **struttura** dati, che deve contenere tre campi: 1) *c1*, di tipo *int*, che deve contenere il pid del nipote; *c2*, di tipo *int*, che deve contenere la lunghezza della linea compreso il terminatore di linea; 3) *c3*, di tipo char[250][*], che deve contenere la linea corrente ricevuta dal nipote.
Il padre deve ricevere, rispettando l'ordine dei file, le singole strutture inviate dai figli e deve stampare su standard output, per ogni struttura ricevuta, ognuno dei campi insieme al nome del file cui le informazioni si riferiscono: **si faccia attenzione al fatto che è demandato al padre il compito di trasformare, in una stringa, la linea ricevuta nel campo c3 di ogni struttura!**
Al termine, ogni processo figlio **Pk** deve ritornare al padre la lunghezza della linea inviata al padre, ma non compreso il terminatore di linea e il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

[*] **Ogni linea si può supporre che abbia una lunghezza massima di 250 caratteri, compreso il terminatore di linea e, se serve, il terminatore di stringa.**

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_2_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_2_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) Per facilitare le operazioni di stampa dei compiti sono imposte le seguenti regole per nominare i file da salvare nella directory **studente_1_2_XXX**:
   - 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) NON devono essere presenti altri file con nome che termina con .sh nella directory **studente_1_2_XXX.**
5) Il tempo a disposizione per la prova è di **90 MINUTI**.
6) 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**.
7) L'assenza di commenti significativi verrà penalizzata!
8) **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.**
9) SI RICORDA CHE IN CASO DI ESITO INSUFFICIENTE è necessario visionare il compito prima di potersi iscrivere a qualunque appello successivo!