COGNOME: TCHEUSSIEU MENDJI NOME: COLINCE T. 1 POS. 3

SISTEMI OPERATIVI e SISTEMI OPERATIVI E LAB. (A.A. 23-24) – 12 GIUGNO 2024

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+1** (con **Z** strettamente maggiore di **1**): i primi **Z** devono essere **nomi assoluti di directory** che identificano **Z** gerarchie (**G1, G2, …**) all'interno del file system, mentre l'ultimo parametro (**car**) deve essere un singolo carattere. 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 **dirAss** corrispondente - tramite un file comandi ricorsivo, **FCR.sh** – e deve cercare tutte le directory (inclusa la radice della gerarchia) che rispettano le seguenti specifiche: *a)* il nome della directory deve essere esattamente di **5** caratteri; *b)* i caratteri *dispari* (sempre del nome della directory) devono essere uguali a **car**; *c)* la directory deve contenere almeno un file *leggibile*, *non VUOTO*, la cui lunghezza in linee sia **pari**. Si riporti il nome assoluto delle *directory trovate* sullo standard output; nel caso siano soddisfatte solo le specifiche *a)* e *b)*, ma non la *c)*, si riporti sempre il nome assoluto di tali directory, ma insieme con una frase opportuna. *In ogni directory trovata* (cioè, in cui sono soddisfatte tutte e 3 le specifiche *a/b/c)*, si deve invocare la parte in C, passando come parametri i nomi relativi semplici dei file che soddisfano la specifica *c)*.

NOTA BENE NEI DUE FILE COMANDI SI USI OBBLIGATORIAMENTE:
- una variabile di nome **car** per contenere l'ultimo parametro di FCP.sh;
- una variabile di nome **dirAss** per le singole gerarchie di ognuna delle **Z** fasi;
- una variabile di nome **item** per identificare, via via, i singoli file delle directory esplorate;
- una variabile di nome **Cont** per contare i file che soddisfano la specifica *c)*.

La parte in C accetta un numero variabile di parametri **V strettamente maggiore di 0** che rappresentano **V** nomi di file (**f1, …fV**): si assuma (senza bisogno di effettuare alcun controllo) che i file abbiano tutti lunghezza in linee pari♣. Il processo padre deve generare **2*V** processi figli (**P0** … **P2*V-1**); tali processi figli costituiscono **V** coppie di processi: ogni coppia **Cj** è composta dal processo **Pj** (primo processo della coppia) e dal processo **Pj+V** (secondo processo della coppia), con **j** variabile da **0 a V-1**. Ogni coppia di processi figli **Cj** è associata ad uno dei file **itemj+1**♠. Il secondo processo della coppia deve, per prima cosa, creare (nella directory corrente) un file il cui nome risulti dalla concatenazione del nome del file associato alla coppia con la stringa .**Min** (ad esempio se il file associato è /tmp/pippo.txt, il file creato si deve chiamare /tmp/pippo.txt.Min). Tutte le coppie **Cj** di processi figli eseguono concorrentemente leggendo tutte le linee del proprio file associato: in particolare, il primo processo di ogni coppia deve identificare le linee di posizione dispari, mentre il secondo processo deve identificare le linee di posizione pari. La comunicazione fra ogni coppia è tale che il primo processo, dopo la lettura di ogni linea di posizione dispari, invia al secondo processo *DUE* informazioni: la lunghezza della linea dispari corrente (compreso il terminatore di linea♦) e la linea stessa. Mentre, il secondo processo di ogni coppia, dopo la lettura e identificazione di ogni linea di posizione pari, deve ricevere (con DUE read!) dal primo processo della coppia le DUE informazioni (lunghezza linea e linea); quindi, il secondo processo di ogni coppia deve scrivere sul file creato la propria linea di posizione pari *se* questa ha lunghezza minore di quella ricevuta o, *altrimenti*, la linea di posizione dispari ricevuta. Al termine, ogni processo di ogni coppia deve ritornare al padre il valore minimo calcolato sulle linee di loro competenza. Il padre, dopo che i figli sono terminati, deve stampare su standard output i PID di ogni figlio con il corrispondente valore ritornato.

NOTA BENE NEL FILE C main.c SI USI OBBLIGATORIAMENTE:
- una variabile di nome **V** per il numero di file;
- una variabile di nome **v** per l'indice dei processi figli;
- una variabile di nome **BUFF** per la linea corrente (pari/dispari) letta dai figli dal proprio file; *si supponga che 250 caratteri siano sufficienti per ogni linea, compreso il terminatore di linea.*
- una variabile di nome **lung** per il valore minimo della lunghezza delle linee pari/dispari dei file;
- una variabile di nome **createdfile** per il file descriptor del file creato.

♣ Si può supporre che l'ultima linea di tutti i file abbia sempre il terminatore di linea. Inoltre, si consideri che la prima linea dei file abbia numero 1 e quindi sia dispari: si veda esempio sul retro del foglio!

♠ Se **N** è 3 (j varia da 0 a 2), le coppie di processi e i file associati sono P0-P3 per F1, P1-P4 per F2 e P2-P5 per F3.

♦ Nel resto del testo tutte le volte che si parlerà di lunghezza di una linea si intenderà compreso il terminatore di linea.

COGNOME: TCHEUSSIEU MENDJI NOME: COLINCE T. 1 POS. 3

IMPORTANTE:

SEGUIRE TUTTE LE REGOLE FORNITE PRIMA DELLO SVOLGIMENTO DELL'ESAME!

ESEMPIO DI FILE PASSATI COME PARAMETRI ALLA PARTE C:

Primo file f1.txt	Secondo file f2.txt
SONO LA LINEA numero 1 di f1.txt, ci sono 2 LINEE. SONO LA SECONDA LINEA con ancora altri caratteri	SONO LA PRIMA LINEA di f2.txt e sono LUNGO 4 LINEE SONO LA LINEA nro 2 di f2.txt Ancora una linea Ed ecco LA LINEA nro 4 di questo file f2.txt!

Linee dispari del file f1.txt

```
SONO LA LINEA numero 1 di f1.txt, ci sono 2 LINEE.
```

Linee pari del file f1.txt

```
SONO LA SECONDA LINEA con ancora altri caratteri
```

Linee dispari del file f2.txt

```
SONO LA PRIMA LINEA di f2.txt e sono LUNGO 4 LINEE
Ancora una linea
```

Linee pari del file f2.txt

```
SONO LA LINEA nro 2 di f2.txt
Ed ecco LA LINEA nro 4 di questo file f2.txt!
```