SISTEMI OPERATIVI E LAB.
(A.A. 20-21) – 16 FEBBRAIO 2022

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 **Q+1** (con **Q** maggiore o uguale a **2**): il primo parametro (**C**) deve essere considerato un singolo carattere *alfabetico minuscolo*, mentre gli altri **Q** devono essere **nomi assoluti di directory** che identificano **Q** gerarchie (**G1, G2, …**) all'interno del file system. Il comportamento atteso dal programma, dopo il controllo dei parametri, è organizzato in **Q** fasi, una per ogni gerarchia.
Il programma, per ognuna delle **Q** fasi, deve esplorare la gerarchia **G** corrispondente - tramite un file comandi ricorsivo, **FCR.sh** – e deve cercare tutte le directory che contengono almeno un file **leggibile** che contenga (nel suo contenuto) almeno una occorrenza del carattere **C**. Si riporti il nome assoluto di tali directory sullo standard output. Al termine di tutte le **Q** fasi, *se sono stati trovati almeno 2 file*, si deve invocare la parte in C, passando come parametri i nomi assoluti di tutti i file trovati globalmente e il carattere **C**.

NOTA BENE NEI DUE FILE COMANDI SI USI OBBLIGATORIAMENTE:
- una variabile di nome **C** per contenere il primo parametro di FCP.sh;
- una variabile di nome **G** per le singole gerarchie di ognuna delle **Q** fasi;
- il nome **/tmp/nomiAssoluti** per il file temporaneo;
- una variabile di nome **N** per contenere il numero dei file trovati globalmente;
- una variabile di nome **F** per identificare, via via, i singoli file delle directory esplorate in FCR.sh.

La parte in C accetta un numero variabile di parametri **N+1** con **N** maggiore o uguale a **2**: i primi **N** rappresentano nomi di file (**F1, …FN**), mentre l'ultimo parametro **C** rappresenta un singolo carattere *alfabetico minuscolo* (*da controllare*). Il processo padre deve generare **N** processi figli (**P0** … **PN-1**): i processi figli **Pi (**con **i** variabile da **0 a N-1)** sono associati agli **N** file **Fk** (con k= **i** +1). Ogni processo figlio **Pi** deve leggere i caratteri del file associato **Fk** cercando il carattere **C**. 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 **PN-1** 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) *c1*, di tipo int, che deve contenere il pid di un processo; 2) *c2*, di tipo long int, che deve contenere il numero di occorrenze del carattere **C** calcolate dal corrispondente processo. *Gli array di strutture DEVONO essere creati da ogni figlio della dimensione minima necessaria per la comunicazione sia in ricezione che in spedizione.* Quindi la comunicazione deve avvenire in particolare in questo modo: il figlio **P0** passa in avanti (cioè comunica *con una singola write*) un array di strutture **A1**, che contiene una sola struttura con *c1* uguale al proprio pid e con *c2* uguale al numero di occorrenze del carattere **C** trovate da **P0** nel file **F1**; il figlio seguente **P1**, dopo aver calcolato numero di occorrenze del carattere **C** nel file **F2**, deve leggere (*con una singola read*) l'array **A1** inviato da **P0** e quindi deve confezionare l'array **A2** che corrisponde all'array **A1** aggiungendo all'ultimo posto la struttura con i propri dati e la passa (*con una singola write*) al figlio seguente **P2**, etc. fino al figlio **PN-1**, che si comporta in modo analogo, ma passa al **padre**. Quindi, il processo padre deve allocare l'array **AN** per ricevere quanto inviato dall'ultimo figlio e cioè l'array di **N** strutture (uno per ogni processo **P0** … **PN-1)**. Il padre deve leggere (*con una singola read*) l'array **AN** e, quindi, deve riportare i dati di ognuna delle **N** strutture su standard output insieme al numero d'ordine del processo corrispondente, al nome del file associato a tale processo e al carattere **C**.
Al termine, ogni processo figlio **Pi** deve ritornare al padre il valore intero corrisponde al proprio indice d'ordine (**i**); il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

NOTA BENE NEL FILE C main.c SI USI OBBLIGATORIAMENTE:
- una variabile di nome **N** per il numero di file;
- una variabile di nome **i** per l'indice dei processi figli;
- una variabile di nome **ch** per il carattere letto dai file dai figli;
- una variabile di nome **cur** per l'array dinamico creato da ogni figlio (della dimensione minima necessaria) e dal padre.