SISTEMI OPERATIVI E LAB.
(A.A. 22-23) – 14 FEBBRAIO 2024

IMPORTANTE:

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

Esercizio

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

TESTO PARTE SHELL: ATTENZIONE LEGGERE ANCHE LA NOTA SEGUENTE AL TESTO!

La parte in Shell deve prevedere un numero variabile di parametri **N+2** (con **N** maggiore o uguale a **2**): il primo parametro deve essere il **nome assoluto di una directory** che identifica una gerarchia all'interno del file system (**G**), il secondo parametro deve essere considerato un numero **intero** (**X**) **strettamente positivo**, mentre gli altri **N** devono essere considerati semplici stringhe (**S1, S2, …**) costituite da almeno due caratteri ciascuna (*da controllare!*). Il comportamento atteso dal programma, dopo il controllo dei parametri, è organizzato in una singola fase.

Il programma deve esplorare la gerarchia **G** - tramite un file comandi ricorsivo, **FCR.sh** – e deve cercare tutte le directory che contengono *almeno* un file leggibile con lunghezza in linee maggiore o uguale a **X**: si riporti il nome assoluto di tali directory sullo standard output. In ogni directory trovata e *per ogni file* **F** che rispetta la specifica precedente, si deve invocare la parte in C passando come parametri il nome del file corrente **F** e tutte le stringhe passate come parametri (**S1, S2, …**).

NOTA BENE NEI DUE FILE COMANDI SI USI OBBLIGATORIAMENTE:

- una variabile di nome **G** per contenere il primo parametro di FCP.sh;
- una variabile di nome **X** per contenere il secondo parametro di FCP.sh;
- una variabile di nome **F** per identificare, via via, i singoli file delle directory e poi quelli trovati.

OSSERVAZIONE: se per provare la parte shell, si commenta la chiamata alla parte C, ricordarsi di togliere il commento prima della consegna!

TESTO PARTE C: ATTENZIONE LEGGERE ANCHE LA NOTA SEGUENTE AL TESTO!

La parte in C accetta un numero variabile di parametri **N+1** (con **N** maggiore o uguale a **2**): il primo rappresenta il nome di un file (**F**), mentre gli altri **N** parametri rappresentano delle semplici stringhe (**S1, …, SN**).

Il processo padre deve generare un numero di **processi figli** pari a **N**: ogni processo figlio **Pn** è associato ad una delle stringhe **S1, …, SN** (*in ordine*). Ognuno di tali figli deve creare a sua volta un processo nipote **PPn**: ogni processo nipote **PPn** è associato alla stessa stringa del figlio che lo ha creato ed esegue concorrentemente; il compito del processo nipote **PPn** è quello di cercare nel file **F** la stringa associata al figlio, usando in modo opportuno il comando ***grep*** di UNIX/Linux. Ogni processo figlio **Pn** deve recuperare il valore tornato dal proprio nipote **PPn** e, sulla base del valore ricevuto, deve confezionare la propria struttura dati, come indicato nel seguito.

I processi figli **Pn** 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 **cur** di **strutture** dati di tipo **Strut**; ogni struttura deve contenere due campi: 1) *c1*, di tipo char[12], che deve contenere la stringa "TROVATA" o "NON TROVATA"; 2) *c2*, di tipo int, che deve contenere il pid del processo nipote **PPn**. *Gli array di strutture DEVONO essere creati da ogni figlio della dimensione minima necessaria per la comunicazione sia in ricezione che in spedizione* (come visto in altri testi di esame). Quindi, al processo padre deve arrivare l'array **cur** di **N** strutture (uno per ogni processo **P0** … **PN-1)**. Il padre deve riportare i dati di ognuna delle **N** strutture su standard output insieme al numero d'ordine **n** del processo figlio/nipote corrispondente, alla stringa **Sn** corrispondente e al nome del file **F**.

Al termine, ogni processo figlio **Pn** deve ritornare al padre il valore ritornato dal proprio processo nipote e il padre deve stampare su standard output i 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 processi figli;
- una variabile di nome **n** per l'indice dei processi figli;
- un tipo di nome **Strut** per le strutture dati;
- una variabile di nome **cur** (i cui elementi devono essere di tipo **Strut)** per l'array usato sia dai figli che dal padre.