SISTEMI OPERATIVI e SISTEMI OPERATIVI E LAB.
(A.A. 23-24) – 19 FEBBRAIO 2025

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 deve essere considerato un numero intero strettamente maggiore di 1 e strettamente minore di **255** (**X)**, 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 che abbia un numero di linee esattamente uguale a **X**: si riporti il nome assoluto di tali directory sullo standard output. Al termine di tutte le **Q** fasi, si deve riportare il numero totale **N** di file trovati: se tale numero **è minore di 2** deve essere data un'indicazione di errore e si deve uscire con errore; altrimenti, si deve invocare la parte in C, passando come parametri i **nomi** dei *file trovati* (**F1, ... FN**).

NOTA BENE NEI DUE FILE COMANDI SI USI OBBLIGATORIAMENTE:
- una variabile di nome **X** 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 **F** per identificare, via via, i singoli file delle directory esplorate;
- una variabile di nome **N** per il numero totale di file che soddisfano la specifica (*file trovati*).

La parte in C accetta un numero variabile **N** di parametri (con **N** maggiore o uguale a **2**, *da controllare*) che rappresentano **N** nomi di file (**F1, ... FN**): si assuma (*senza bisogno di effettuare alcun controllo*) che i file abbiano tutti la stessa lunghezza in linee.
Il processo padre deve generare **N processi figli** (**P0, … PN-1**): i processi figli **Pn** sono associati ad uno dei file **F1, ... FN**. Ogni processo figlio **Pn** deve leggere tutte le linee del proprio file associato; dopo la lettura di *ogni* linea, il processo figlio **Pn** deve comunicare al padre l'ultimo carattere della linea corrente (*escluso il terminatore di linea*[°]) e deve ricevere dal padre l'indicazione di stampare o meno su standard output delle informazioni (*vedi dopo* *). Il padre deve ricevere, rispettando l'ordine dei file, da ogni figlio via via i caratteri che rappresentano l'ultimo carattere della linea corrente. Quindi, al processo padre devono arrivare per ogni linea dei file **F1, ... FN**, via via, **N caratteri**: ad esempio, per la prima linea, il padre riceve l'ultimo carattere della prima linea inviato dal figlio **P0**, etc. e l'ultimo carattere della prima linea inviato dal figlio **PN-1**; *così via per ogni linea*. Per ogni insieme di **N caratteri** ricevuti, il padre deve determinare il valore **massimo** e, SOLO AL PROCESSO FIGLIO CHE HA INVIATO TALE VALORE, deve indicare *(*)* di stampare su standard output **l'indice d'ordine del processo, il suo pid, il carattere identificato come massimo e quindi la linea corrente**, mentre a tutti gli altri processi figli deve indicare di non stampare[♦].
Al termine, ogni processo figlio **Pn** deve ritornare al padre il numero di linee che ha scritto sullo standard output (*sicuramente minore di 255, garantito dalla parte SHELL*) 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 file;
- una variabile di nome **n** per l'indice dei processi figli;
- una variabile di nome **linea** *per l'array di 250 caratteri* usato per leggere, da parte dei figli, via via le singole linee; *si supponga che 250 caratteri siano sufficienti per ogni linea, compreso il terminatore di linea e il terminatore di stringa.*

[°] Si può supporre che in ogni linea dei file associati ai processi figli ci sia almeno un carattere, oltre il terminatore di linea e quindi che non ci siano linee 'bianche'!

[♦] Per questo tipo di interazione, volendo, si possono usare i segnali.