SISTEMI OPERATIVI E LAB.
(A.A. 20-21) – 14 LUGLIO 2021

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 **Q+2** (con **Q** maggiore o uguale a **2**): i primi **Q** devono essere **nomi assoluti di directory** che identificano **Q** gerarchie (**G1, G2, …**) all'interno del file system, mentre gli ultimi due parametri devono essere considerati numeri interi strettamente positivi (**H** e **M**). 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 un **numero strettamente minore di H ma maggiore o uguale a 2** di file (**F1, F2, ...**) il cui nome sia costituito da esattamente **2** caratteri e con lunghezza in linee esattamente uguale a **M**. Si riporti il nome assoluto di tali directory sullo standard output. In ognuna di tali directory trovate, si deve invocare la parte in C, passando come parametri i nomi dei file trovati (**F1, F2, ...**) che soddisfano la condizione precedente e il numero **M**.

NOTA BENE NEI DUE FILE COMANDI SI USI OBBLIGATORIAMENTE:
- una variabile di nome **H** per contenere il penultimo parametro di FCP.sh;
- una variabile di nome **M** per contenere l'ultimo parametro di FCP.sh;
- una variabile di nome **G** per le singole gerarchie di ognuna delle **Q** fasi;
- una variabile di nome **F** per identificare, via via, i singoli file delle directory esplorate;
- una variabile di nome **cont** per contare i file che soddisfano la specifica.

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 maggiore o uguale a 2** che rappresentano i primi N nomi di file (**F1, …FN**), mentre l'ultimo rappresenta un numero intero strettamente positivo (**nroLinee**) (da controllare) che indica la lunghezza in linee dei file. Il processo padre deve, per prima cosa, creare nella directory *corrente* un file **fcreato** con nome corrispondente al proprio COGNOME (tutto scritto in maiuscolo, in caso di più cognomi se ne usi solo uno, inserendo un opportuno commento).
Il processo padre deve generare un numero di **processi figli** pari a **N**: ogni processo figlio **Pn** è associato ad uno dei file **F1, …FN** (*in ordine*); la lunghezza in linee di tutti i file è uguale a **nroLinee** e non deve essere controllata.
Ognuno di tali processi figli **Pn** esegue concorrentemente e legge tutte le **linee** del proprio file associato: le linee lette devono essere scritte nel file **fcreato** seguendo le indicazioni fornite nel seguito.
I processi figli **Pn devono usare uno 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, per ognuna delle **nroLinee** linee dei file, di un array di grandezza **N** e in cui ogni elemento dell'array corrisponde **alla linea corrente** (*supposta lunga 250 caratteri compreso il terminatore di linea*[1*]) letta dal corrispondente processo figlio **Pn.** Quindi, il generico processo **Pn**, dopo aver letto la linea corrente, deve ricevere dal figlio precedente (a parte il processo **P0**) l'array di linee e, dopo aver inserito la linea corrente nella posizione giusta dell'array di linee, deve inviare l'array di linee al figlio successivo, con **PN-1** che manda al padre. Quindi al padre deve arrivare, per ognuna delle **nroLinee** linee un array di grandezza **N** e in cui ogni elemento dell'array corrisponde **alla linea corrente letta dai figli Pn**: il padre deve scrivere le linee correnti sul file **fcreato**.
Al termine dell'esecuzione, ogni figlio **Pn** ritorna al padre la lunghezza dell'ultima linea letta dal proprio file **compreso** il terminatore di linea (sicuramente minore di 255); il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.

ESEMPIO DI FILE I CUI NOMI VENGONO PASSATI COME PARAMETRI

Primo file (di lung. 2 linee)	Secondo file (di lung. 2 linee)
SONO LA LINEA numero 1, ci sono 2 LINEE. SONO LA SECONDA LINEA; e altri car: 0 1 2	SONO LA PRIMA LINEA e sono LUNGO 2 LINEE SONO LA LINEA nro 2 di f2.txt.

***CONTENUTO FINALE DEL FILE* fcreato**
```
SONO LA LINEA numero 1, ci sono 2 LINEE.
SONO LA PRIMA LINEA e sono LUNGO 2 LINEE
SONO LA SECONDA LINEA; e altri car: 0 1 2
SONO LA LINEA nro 2 di f2.txt.
```

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 **nroLinee** per la lunghezza in linee dei file;
- una variabile di nome **n** per l'indice dei processi figli;
- una variabile di nome **linea** per la linea letta correntemente dai figli dal proprio file;
- una variabile di nome **tutteLinee** per l'array con tutte le linee lette correntemente dai figli.

[*] Si suggerisce di usare un typedef per definire un tipo array di 250 char!