SISTEMI OPERATIVI e SISTEMI OPERATIVI E LAB.
(A.A. 24-25) – 10 SETTEMBRE 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 (**C**) deve essere un singolo carattere, 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 (inclusa la radice della gerarchia) che rispettano le seguenti specifiche: il nome della directory deve essere esattamente di **3** caratteri e i caratteri *dispari* devono essere uguali a **C** e deve contenere almeno un file *leggibile*, *non VUOTO*, la cui lunghezza in caratteri sia **pari**. Sullo standard output, si riporti il nome assoluto delle *directory esplorate sia di quelle che rispettano le specifiche sopra riportate* (**poi riferite come directory trovate**) e *sia di quelle che non rispettano le specifiche,* con l'indicazione della specifica NON rispettata (si veda l'ESEMPIO DI OUTPUT riportato sul retro del foglio). **In ogni directory trovata**, si deve invocare la parte in C, passando come parametri i nomi relativi semplici dei file che soddisfano le precedenti specifiche.

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, da usare in FCP.sh;
- una variabile di nome **F** per identificare, via via, i singoli file delle directory esplorate, da usare in FCR.sh;
- una variabile di nome **cont** per contare/identificare i file che soddisfano la specifica, da usare in FCR.sh.

La parte in C accetta un numero variabile di parametri **N maggiore o uguale a 1** che rappresentano **N** nomi di file (**F1, …FN**): si assuma (senza bisogno di effettuare alcun controllo) che i file abbiano tutti lunghezza in caratteri **pari**. Il processo padre deve generare **2*N** processi figli (**P0** … **P2*N-1**); tali processi figli costituiscono **N** *coppie di processi*: ogni coppia **Ci** è composta dal processo **Pi** (primo processo della coppia) e dal processo **Pi+N** (secondo processo della coppia), con **i** variabile da **0 a N-1**. Ogni coppia di processi figli **Ci** è associata ad uno dei file **Fi+1**[♠]. Il primo 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 pippo.txt, il file creato si deve chiamare pippo.txt.min). Tutte le coppie **Ci** di processi figli eseguono concorrentemente leggendo tutti caratteri del proprio file associato: in particolare, il primo processo di ogni coppia deve identificare i caratteri di posizione pari, mentre il secondo processo deve identificare i caratteri di posizione dispari[♦]. La comunicazione fra ogni coppia è tale che il secondo processo di ogni coppia, dopo la lettura e l'identificazione di ogni carattere di posizione dispari lo deve inviare al primo processo della coppia; Mentre, il primo processo di ogni coppia, dopo la lettura e l'identificazione di ogni carattere di posizione pari, deve ricevere dal secondo processo della coppia il carattere inviato; quindi, il primo processo di ogni coppia deve scrivere sul file creato il proprio carattere di posizione pari *se* questo è minore di quello ricevuto o, *altrimenti*, il carattere di posizione dispari ricevuto. Al termine, ogni processo di ogni coppia deve ritornare al padre il valore minimo calcolato sui caratteri di propria competenza. Il padre, dopo che i figli sono terminati, deve stampare su standard output il PID di ogni figlio con il corrispondente 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 **Car** per il carattere corrente (pari/dispari) letto dai figli dal proprio file;
- una variabile di nome **fcreato** per il file descriptor del file creato.

♠ Se **N** è 3 (i 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.

♦ Si consideri che la prima posizione dei caratteri di un file corrisponda alla posizione 0L, in analogia con la semantica della primitiva lseek!

IMPORTANTE:

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

ESEMPIO DI OUTPUT PER PARTE SHELL, dove i nomi assoluti delle directory sono stati parzialmente omessi (sono stati completati con la scritta OMISSIS e un numero!); le stampe per le directory che soddisfano le specifiche sono state sottolineate:

```
DIRECTORY TROVATA CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS1
LA DIRECTORY CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS1/OMISSIS2 NON HA IL NOME GIUSTO
DIRECTORY TROVATA CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS1/OMISSIS2/OMISSIS3
DIRECTORY TROVATA CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS1/OMISSIS4
LA DIRECTORY CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS5 NON CONTIENE ALCUN FILE LEGGIBILE DI
    DIMENSIONE NON NULLA O CON LUNGHEZZA IN CARATTERI PARI!
LA DIRECTORY CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS5/OMISSIS6 NON HA IL NOME GIUSTO
LA DIRECTORY CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS5/OMISSIS7 NON HA IL NOME GIUSTO
DIRECTORY TROVATA CON NOME ASSOLUTO /home/soELab/10Set25/OMISSIS5/OMISSIS7/OMISSIS8
```