COGNOME: DI CANDIA NOME: ANGELO LORENZO T. 2 POS. 46

SISTEMI OPERATIVI/SISTEMI OPERATIVI E LAB. (A.A. 24-25) – 5 GIUGNO 2025

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** (già svolta) e una parte in **C**.

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

La parte in C accetta esattamente **tre** parametri: il primo deve essere considerato un numero intero strettamente positivo (**M**, da controllare), il secondo rappresenta il nome di un file (**FILE**), mentre il terzo deve essere una stringa. Per quanto sarà chiaro leggendo il resto della specifica, il significato dei parametri è il seguente:

- **M** rappresenta la lunghezza in linee di **FILE**, senza bisogno di effettuare alcun controllo su questo;
- **FILE** contiene in ognuna delle sue linee il nome di un file; non ha importanza che siano nomi assoluti, relativi alla directory corrente o relativi semplici e quindi non c'è bisogno di effettuare alcun controllo (in ogni modo il padre riporterà, nel caso, un errore sulla base di quanto segnalato dai nipoti, tramite i figli: si veda ultima frase della specifica e retro del foglio); quindi, il formato di questo file è il tipico formato dei file temporanei che possono essere necessari per risolvere la parte in SHELL;
- il terzo parametro deve essere una stringa compatibile con quanto devono eseguire i processi nipoti e si può supporre che sia data in forma corretta e quindi non c'è bisogno di effettuare alcun controllo (in ogni modo il padre riporterà, nel caso, un errore sulla base di quanto segnalato dai nipoti, tramite i figli: si veda sempre ultima frase della specifica e retro del foglio).

Il processo padre deve, *per prima cosa*, aprire in *sola* lettura il file **FILE**. Quindi, il processo padre deve generare un numero di **processi figli** pari a **M**: ogni processo figlio **Pm** è associato ad una delle linee del file **FILE** (*in ordine*); quindi, il figlio **P0** è associato alla prima linea del file **FILE**, il figlio **P1** alla seconda linea del file **FILE** e così via fino al figlio **PM-1** che è associato all'ultima linea del file **FILE**.

Ognuno di tali processi figli **Pm** esegue concorrentemente e *legge la singola linea ad esso associata*: infatti, i processi figli **devono sincronizzarsi a vicenda** in modo che le letture dal file **FILE** avvengano **in ordine** dal primo processo all'ultimo in modo che ogni processo legga appunto solo la propria linea associata. Si usi per i processi figli **Pm uno schema di sincronizzazione a pipeline, che però risulta un po' degenere per due ragioni: a) nella pipeline non c'è il padre; b) l'ultimo figlio non deve mandare a nessuno**: il generico processo **Pm** (a parte il processo **P0**), dopo aver ricevuto l'ok dal figlio precedente, legge la propria linea e poi manda l'ok al figlio successivo (a parte il processo **PM-1** che appunto essendo l'ultimo non manda a nessuno!).

Ogni processo figlio **Pm**, dopo la lettura della propria linea associata, deve creare a sua volta un processo *nipote* **PPm;** i processi figli e nipoti eseguono concorrentemente.

Il compito di ogni processo nipote **PPm** è quello di cambiare, secondo quanto indicato nella stringa passata come terzo parametro, i diritti al file il cui nome è stato *ricavato* dal figlio nella propria linea associata, usando in modo opportuno il comando ***chmod*** di UNIX/Linux**.**

Al termine, ogni processo figlio **Pm** torna al padre il risultato del comando ***chmod*** eseguito dai nipoti e il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato. Si inserisca un opportuno controllo nel caso questo valore non rappresenti il valore di successo del comando ***chmod***!

NOTA BENE NEL FILE C main.c SI USI OBBLIGATORIAMENTE:

- una variabile di nome **M** per il primo parametro che corrisponderà alla lunghezza in linee di **FILE** e, quindi, al numero di processi figli;
- una variabile di nome **m** per l'indice dei processi figli;
- una variabile di nome **openfile** per il file descriptor usato dal padre per la *apertura in lettura* di **FILE**; **N.B. FILE**, che dovrà essere creato da ogni studente/studentessa per provare il funzionamento del programma eseguibile, dovrà avere un contenuto che sia coerente con i nomi dei file su cui dovranno andare ad operare i nipoti;
- una costante di nome **LSIZE** per la lunghezza massima delle linee (compreso il terminatore di linea); **N.B.** Per provare la soluzione ogni studente/studentessa deve chiaramente scegliere un valore per tale costante che sia congruente con quanto verrà scritto nel **FILE** per provare il funzionamento del programma eseguibile!
- una variabile di nome **LINE** per l'array di **LSIZE** caratteri usato per leggere, da parte dei figli, la propria linea.

COGNOME: DI CANDIA NOME: ANGELO LORENZO T. 2 POS. 46

IMPORTANTE:

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

ESEMPI DI OUTPUT, supponendo che il file passato come secondo parametro abbia 7 linee e che quindi il primo parametro sia il numero 7 (tutte le stampe riportate sono effettuate SOLO dal processo padre):

1) *Caso corretto*

```
DEBUG-Sono il processo padre con pid 24571 e creero' 7 processi figli
Il nipote del figlio con pid=24572 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24574 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24573 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24577 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24576 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24578 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24579 ha ritornato 0, quindi tutto OK!
```

2) *Caso non corretto, una linea del file passato come secondo parametro contiene un nome di file che NON esiste*

```
DEBUG-Sono il processo padre con pid 24598 e creero' 7 processi figli
Il nipote del figlio con pid=24599 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24600 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24602 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24601 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24604 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24606 ha ritornato 0, quindi tutto OK!
Il nipote del figlio con pid=24605 ha ritornato 0, quindi tutto OK!
```

3) *Caso non corretto, il terzo parametro non è una stringa accettabile per ciò che devono svolgere i processi nipoti*

```
DEBUG-Sono il processo padre con pid 24633 e creero' 7 processi figli
Il nipote del figlio con pid=24636 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24634 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24639 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24638 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24635 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24640 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
Il nipote del figlio con pid=24641 ha ritornato 1 e quindi ha fallito l'esecuzione
del comando chmod oppure se 255 il nipote o il figlio ha avuto dei problemi
```