Esercitazione di Venerdì 25 Maggio 2018

1. (19Giu13.c) Esame del 19 Giugno 2013 (consideriamo la sola parte C): La parte in C accetta un numero variabile di parametri che rappresentano N nomi di file F1..FN: tutti i file Fi hanno uguale lunghezza in byte e contengono solo caratteri numerici (queste due proprietà sono garantite dalla parte shell e non devono essere controllate).
   Il processo padre deve generare N processi figli (P0 … PN-1): ogni processo figlio è associato ad uno dei file Fi. Ognuno di tali processi figli esegue concorrentemente, legge tutti i caratteri del file associato Fi e calcola per ogni carattere numerico letto il numero intero positivo corrispondente. I processi figli e il processo padre devono attenersi ad uno schema di comunicazione a pipeline; il figlio PN-1 comunica con il figlio PN-2 etc. fino al figlio P0 che comunica con il padre; questo schema a pipeline deve essere ripetuto per ogni carattere di ogni file e deve prevedere l'invio indietro, per ogni carattere, via via di una struttura Si che deve contenere due campi, c1 e c2, con c1 uguale al PID di un figlio e con c2 uguale ad un numero intero positivo. In particolare, l'ultimo processo PN-1 per ogni carattere numerico letto dopo aver calcolato il numero corrispondente (NumeroN-1), passa indietro (cioè comunica) una struttura SN-1 con il proprio PID e NumeroN-1; il processo precedente PN-2, dopo aver calcolato il numero (NumeroN-2) corrispondente al carattere numerico corrente, riceve da PN-1 l'informazione comunicata e passa al processo PN-3 l'informazione ricevuta da PN-1 se NumeroN-1 è maggiore o uguale di NumeroN-2 oppure la propria informazione e così via fino a che il primo processo P0, dopo aver calcolato il numero (Numero0) corrispondente al carattere numerico corrente, riceve da P1 l'informazione comunicata e passa al processo padre l'informazione ricevuta da P1 se Numero1 è maggiore o uguale di Numero0 oppure la propria informazione. Quindi, al processo padre devono arrivare tante strutture quanti sono i caratteri numerici dei file letti dai processi P0, P1, … PN-1. Il padre ha il compito di stampare su standard output tutte le informazioni ricevute dal processo P0, che rappresentano il numero corrispondente con massimo valore e il PID del processo che lo ha calcolato per ognuno dei caratteri numerici presenti nei file letti.
   Al termine, ogni processo figlio Pi deve ritornare al padre l'ultimo carattere numerico letto dal proprio file associato Fi e il padre deve stampare su standard output i PID e i valori ritornati dai figli.
2. (9Set16.c) Esame del 9 Settembre 2016 (consideriamo la sola parte C): La parte in C accetta un unico parametro che rappresenta il nome assoluto di un file (F) (senza bisogno di controlli sul fatto che sia assoluto).
   Il processo padre deve generare 26 processi figli (P0, P1, … P25) tanti quanti i caratteri dell'alfabeto inglese: tutti i processi figli Pi (con i che varia da 0 a 25) sono associati all'unico file F e ognuno dei processi figli è associato al carattere alfabetico minuscolo corrispondente (P0 è associato al carattere 'a' fino a P25 che è associato al carattere 'z'). Ogni processo figlio Pi deve leggere i caratteri del file F cercando il carattere a lui associato Ci (per i=0, C0='a', … per i=25, C25='z'). I processi figli 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 P25 che comunica con il padre. Questo schema a pipeline deve prevedere l'invio in avanti di un array di strutture dati ognuna delle quali deve contenere due campi: 1) v1, di tipo char, che deve contenere il carattere Ci; 2) v2, di tipo long int, che deve contenere il numero di occorrenze del carattere Ci, calcolate dal corrispondente processo. Ogni array di strutture utilizzato dai figli e dal padre deve avere dimensione fissa (26 elementi!). Quindi la comunicazione deve avvenire in particolare in questo modo: il figlio P0 passa in avanti (cioè comunica) un array di strutture A0 (di 26 elementi), che contiene una sola struttura significativa (nell'elemento di indice 0 dell'array A0) con v1 uguale a 'a' e con v2 uguale al numero di occorrenze del carattere 'a' trovate da P0 nel file F; il figlio seguente P1, dopo aver calcolato numero di occorrenze del carattere associato C1 nel file F, deve leggere (con una singola read) l'array A0 inviato da P0 e quindi deve confezionare l'array A1 che corrisponde all'array A0 aggiungendo nell'elemento di indice 1 la struttura con i propri dati e la passa (con una singola write) al figlio seguente P2, etc. fino al figlio P25, che si comporta in modo analogo, ma passa al padre. Quindi, al processo padre deve arrivare l'array A25. Il processo padre, dopo aver ordinato tale array A25 in senso crescente rispetto al campo v2, deve riportare i dati di ognuna delle 26 strutture su standard output insieme al pid e all'indice i del processo che ha generato tale struttura.

Al termine, ogni processo figlio Pi deve ritornare al padre l'ultimo carattere letto dal file F; il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato sia come carattere che come valore ASCII (in decimale).

SE PUÒ SERVIRE RIPORTO IL SEGUENTE CODICE dai Lucidi di Fondamenti II e Lab. - Algoritmi di ordinamento (pag. 5):

```
void bubbleSort(int v[], int dim)
{ int i; bool ordinato = false;
while (dim>1 && !ordinato)
{  ordinato = true; /* hp: è ordinato */
   for (i=0; i<dim-1; i++)
     if (v[i]>v[i+1])
     {
         scambia(&v[i],&v[i+1]);
         ordinato = false;
     }
   dim--;
}
}
```