SISTEMI OPERATIVI e

LABORATORIO DI SISTEMI OPERATIVI

(A.A. 13-14) – 10 SETTEMBRE 2014

IMPORTANTE:

1) Fare il login sui sistemi in modalità Linux usando il proprio **username** e **password**, attivare syncexam.sh e passare in modalità testuale.
2) I file prodotti devono essere collocati in un **sottodirettorio** (che deve essere nella directory studente_XXX) che deve essere creato e avere nome **ESAME10Set14-1-01.** FARE ATTENZIONE AL NOME DEL DIRETTORIO, in particolare alle maiuscole e ai trattini indicati. Verrà penalizzata l'assenza del direttorio con il nome indicato e/o l'assenza dei file nel direttorio specificato, al momento della copia automatica del direttorio e dei file. **ALLA SCADENZA DEL TEMPO A DISPOSIZIONE VERRÀ INFATTI ATTIVATA UNA PROCEDURA AUTOMATICA DI COPIA, PER OGNI STUDENTE DEL TURNO, DEI FILE CONTENUTI NEL DIRETTORIO SPECIFICATO.**
3) Il tempo a disposizione per la prova è di **75 MINUTI** per lo svolgimento della sola parte C e di **120 MINUTI** per lo svolgimento di tutto il compito.
4) Non è ammesso **nessun tipo di scambio di informazioni** né verbale né elettronico, pena la invalidazione della verifica.
5) **L'assenza di commenti significativi (per la parte in C, scritti in C standard) verrà penalizzata, così come la mancanza del makefile!**
6) **AL TERMINE DELLA PROVA È INDISPENSABILE CONSEGNARE IL TESTO DEL COMPITO (ANCHE IN CASO CHE UNO STUDENTE SI RITIRI): IN CASO CONTRARIO, NON POTRÀ ESSERE EFFETTUATA LA CORREZIONE DEL COMPITO MANCANDO IL TESTO DI RIFERIMENTO.**

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 **tre** parametri: il primi due parametri devono essere nomi assoluti di direttori che identificano due gerarchie (**G1** e **G2**) all'interno del file system mentre il terzo parametro deve essere considerato un numero intero strettamente positivo (**K**). Il programma deve cercare (in due fasi successive) nelle due gerarchie specificate (prima **G1** e poi **G2**) tutti i direttori che contengono almeno **un** file la cui lunghezza in linee sia maggiore o uguale a **K:** si riporti il nome assoluto di tali direttori sullo standard output. **Al termine dell'intera esplorazione ricorsiva di G1 e G2**, si deve invocare la parte in C passando come parametri i nomi assoluti di tutti i file trovati in entrambe le gerarchie **F0**, **F1**, ... **FN-1** e, come ultimo parametro, **K**.

La parte in C accetta un numero variabile pari **N+1** di parametri maggiore o uguale a 2 (*da controllare*) che rappresentano, i primi **N,** i nomi assoluti di file **F1, F2, … FN-1** *(tutti con lunghezza in linee maggiore o uguale a K, che non deve essere controllata)* mentre l'ultimo rappresenta un numero intero strettamente positivo (**K**, *da controllare*). Il processo padre deve innanzitutto chiedere all'utente di fornire un numero intero **X** strettamente positivo e minore o uguale a **K** e quindi deve generare **N processi figli** (**P0 … PN-1**): ogni processo figlio **Pi** è associato al file **Fi**. Ogni processo figlio **Pi** deve leggere, dal suo file associato, la linea **X**-esima e, calcolata (come *int*) la sua lunghezza **L** compreso il terminatore di linea, deve mandare **L** al processo padre. Il processo padre ha il compito di ricevere, rispettando l'ordine dei file, i valori interi che rappresentano le lunghezze **L** delle linee **X**-esime dei file. Il processo padre, dopo aver ordinato tali valori in senso crescente, deve comunicare ai vari figli **Pi** di riportare la linea **X**-esima sullo standard output: l'ordine di tali comunicazioni deve essere tale per cui vengano stampate prima le linee di maggiore lunghezza secondo l'ordinamento ottenuto; inoltre, prima di inoltrare tale comunicazione, il padre deve riportare sullo standard output **il nome del file** cui la linea **X**-esima si riferisce e la sua **lunghezza L**.
Al termine, ogni processo figlio **Pi** deve ritornare al padre il valore risultante dal resto della divisione intera fra **L** e 255. Il padre, dopo che i figli sono terminati, deve stampare, su standard output, i PID di ogni figlio con il corrispondente valore ritornato.

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--;
        }
}
```