SISTEMI OPERATIVI E LAB.
(A.A. 14-15) – 5 GIUGNO 2015

IMPORTANTE:

1) Fare il login sui sistemi in modalità Linux usando il proprio **username** e **password**, aprire un browser sulla pagina ftp://lica02.lab.unimo.it/README e copiare il comando presente e passare in modalità testuale: sul Desktop, viene creata automaticamente una directory studente_XXX al cui interno viene creato un file denominato student_data.csv che non va eliminato.
2) I file prodotti devono essere collocati nella directory studente_XXX dato che tale directory viene zippata e salvata automaticament sul server ad intervalli di tempo regolari. **ALLA SCADENZA DEL TEMPO A DISPOSIZIONE VERRÀ ATTIVATA UNA PROCEDURA AUTOMATICA DI ESTRAZIONE, PER OGNI STUDENTE DEL TURNO, DEI FILE CONTENUTI NELLA DIRETTORY SPECIFICATA.**
3) Il tempo a disposizione per la prova è di **90 MINUTI** per lo svolgimento della sola parte C.
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 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 (gia svolta) e una parte in C.

La parte in C accetta un numero *variabile* di parametri (**maggiore o uguale a 2,** *da controllare*) che rappresentano **M** nomi assoluti di file **F1...FM**.
Il processo padre deve generare **M processi figli** (**P0 … PM-1**): ogni processo figlio è associato al corrispondente file **Fj**. Ognuno di tali processi figli deve creare a sua volta un **processo nipote** (**PP0 … PPM-1**): ogni processo nipote **PPj** esegue concorrentemente e deve, usando in modo opportuno il comando *tail* di UNIX/Linux, leggere l'ultima linea del file associato **Fi**.
Ogni processo figlio **Pj** deve **calcolare** la lunghezza, in termini di **valore intero** (**lunghezza**), della linea scritta (**escluso il terminatore di linea**) sullo standard output dal comando *tail* eseguito dal processo nipote PPj; quindi ogni figlio **Pj** deve comunicare tale **lunghezza** al padre. Il padre ha il compito di ricevere, rispettando *l'ordine inverso* dei file, il valore **lunghezza** inviato da ogni figlio **Pj** che deve essere riportato sullo standard output insieme all'indice del processo figlio e al nome del file cui tale lunghezza si riferisce.
Al termine, ogni processo figlio **Pj** deve ritornare al padre il valore di ritorno del proprio processo nipote **PPi** e il padre deve stampare su standard output il PID di ogni figlio e il valore ritornato.