Esercizio 2

Scrivere uno script bash con i seguenti argomenti (nell'ordine dato):

1. intervallo di campionamento c;
2. lista di comandi C da lanciare con rispettivi argomenti, terminati da una doppia virgola (`,,`); l'ultimo comando va terminato con una tripla virgola (`,,,`);
3. lista di nomi di file.

Se uno dei suddetti input manca, l'output dovrà essere semplicemente la scritta `Uso:` s `sampling commands files` su standard error (dove s è il nome dello script), e lo script dovrà terminare con exit status 15.

Lo script dovrà lanciare in background e monitorare i comandi in C, ridirigendo sia lo standard output che lo standard error. A tal proposito, la lista di nomi di file data come ultimo argomento dovrà essere del tipo $f_1, f_2, \ldots, f_n, g_1, \ldots, g_n$, dove n è il numero di comandi in C. Sia $C = C_1, \ldots, C_n$; allora lo standard output del comando C_i va rediretto in f_i e lo standard error in g_i. Una mancata concordanza tra numero di comandi e numero di file dovrà portare lo script a terminare con exit status 30, visualizzando su standard error `Uso:` s `sampling commands files` (dove s è il nome dello script). Qualora uno dei comandi in C non esista, non va lanciato. Una volta lanciati tutti i comandi, occorre scrivere sul file descriptor 3 i PID dei processi lanciati (tutti sulla prima riga, separati dal carattere _), *prima* di proseguire con la computazione descritta qui sotto. Lo script, se non riscontra errori, deve rimanere in esecuzione finché non trova un file regolare `done.txt` sulla current working directory. Il monitoraggio di questa condizione deve avvenire ogni c secondi. A quel punto, deve scrivere sul file descriptor 3 `File done.txt trovato`, seguito dalla scrittura su standard output della foresta di processi di C, per poi terminare con exit status 0. La foresta di processi va scritta come una sequenza di righe, ognuna delle quali contiene due PID separati da uno spazio: il primo di un processo padre, il secondo di un processo figlio. Le radici della foresta devono corrispondere a tutti e soli i processi di C. Le righe vanno ordinate (numericamente) prima sul primo PID, e poi sul secondo. Se invece i processi terminano tutti prima che sia possibile monitorarli per scrivere la foresta, occorre scrivere su standard output `Tutti i processi sono terminati` e uscire con exit status 1.

Attenzione: non è permesso usare Python, Java, Perl o GCC. Lo script non deve scrivere nulla sullo standard error, a meno che non si tratti di un errore nelle opzioni da riga di comando come descritto sopra. Non deve mai scrivere nulla sullo standard output, tranne che nei casi descritti sopra. Per ogni test definito nella valutazione, lo script dovrà ritornare la soluzione dopo al più 10 minuti, e lanciare i comandi passatigli entro al più 3 secondi.

Esempi

Da dentro la directory `grader.1`, dare il comando `tar xfzp all.tgz input_output.2 && cd input_output.2`. Ci sono 6 esempi di come lo script `2.sh` può essere lanciato, salvati in file con nomi `inp_out.`i`.sh` (con $i \in \{1, \ldots, 6\}$). Per ciascuno di questi script, la directory `inp.`i contiene uno script `main.sh` e degli altri file necessari per lanciare `2.sh` e controllare che funzioni correttamente. La directory `check/out.`i contiene i file con l'output atteso; lo standard output atteso sarà nel file `check/inp_out.`i`.sh.out`, mentre lo standard error atteso sarà nel file `check/inp_out.`i`.sh.err`. Ovviamente, l'output della directory `check` difficilmente coinciderà con l'output corretto (i PID cambiano...), ma può essere usata come riferimento per la formattazione dell'output stesso.