Università di Pisa
Dipartimento di Informatica
Corso di Laurea in Informatica

Progetto di Laboratorio di Sistemi Operativi

Sessione straordinaria autunnale a.a. 2013-14

Introduzione

Lo studente dovrà realizzare una *shell* alternativa per un sistema UNIX con le caratteristiche indicate nel seguito. Lo scopo di questa shell, che chiameremo *debosh* (per **de**prived **b**asic **o**perating **sh**ell) è quello di fornire un ambiente ristretto in cui gli utenti possono eseguire soltanto un insieme di comandi dato, su un insieme di file dato, mantenendo però la capacità di comporre sequenze degli stessi. I comandi da eseguire in questo ambiente ristretto saranno tipicamente destinati a processare, in pipeline, grandi quantità di dati (la cui natura non ci interessa).

Avvio della shell

La debosh viene avviata con due argomenti, che saranno rispettivamente il pathname della directory in cui risiedono i comandi eseguibili, e quello della directory in cui risiedono i file dati. Non deve essere consentito all'utente della debosh di fare alcun accesso a comandi e file fuori dai due path indicati. Se è presente un terzo parametro, che deve essere il pathname di un file di testo, la debosh dovrà leggere i comandi da eseguire dal file indicato,anziché con le modalità indicate sotto in “Interazione con l'utente”.

Il linguaggio di comandi

Un *comando base* è identificato da un nome (che coincide con il nome dell'eseguibile che lo implementa), e da una serie di argomenti (alcuni dei quali saranno tipicamente nomi di file). Un comando base legge i suoi dati di ingresso da stdin e/o da file citati come argomenti, e scrive i suoi risultati su stdout e/o da file citati come argomenti.

Un *comando complesso* è una sequenza di comandi base, che vengono eseguiti implicitamente in pipeline (lo stdout di un comando base è collegato allo stdin del comando base successivo).

Interazione con l'utente

La debosh richiede all'utente di inserire un comando complesso, in cui ogni comando base è fornito su una linea separata. Quando l'utente immette una linea costituita dal solo carattere “.”, questo viene interpretato come una indicazione che l'immissione del comando complesso è terminata, e si può passare all'esecuzione (vedi sotto).

Durante l'inserimento di una riga, la debosh deve fornire una funzione di *autocompletamento* così strutturata: se l'utente preme il tasto *Tab* durante la digitazione della prima parola di una riga, la debosh deve cercare il primo match (se c'è) fra i comandi eseguibili a cui ha accesso, e completare il nome del comando. Se invece *Tab* viene premuto durante la digitazione di parole successive, la debosh deve esibire lo stesso comportamento, ma relativamente ai file a cui ha accesso. I dettagli dell'autocompletamento sono a discrezione dello studente, che può ispirarsi al comportamento di altre shell. Quando l'utente preme il tasto *Enter*, la riga si intende acquisita (e forma un comando base). Ai soli fini dell'esame, si richiede di **non** utilizzare la libreria readline().

Suggerimento: l'implementazione dell'autocompletamento richiede che il terminale sia impostato in modalità CBREAK o RAW. Ciò può essere ottenuto chiamando la funzione cbreak() della libreria curses o ncurses, oppure (senza aggiungere una libreria) utilizzando una chiamata a termios() con gli opportuni parametri che imposti il terminale in modalità non-canonica. Si veda il capitolo 4.5 di “Advanced UNIX Programming” (specialmente la sezione 4.5.9) per i dettagli. Sarà necessario gestire direttamente almeno il carattere backspace, mentre è opzionale la gestione dei tasti cursore per l'editing.

Esecuzione

Una volta completato l'inserimento di un comando complesso, l'intera pipeline viene messa in esecuzione, lanciando i parallelo tanti processi distinti quanti sono necessari per l'esecuzione, e avendo cura di collegare tutti i comandi base in pipeline. In caso di errore nell'esecuzione di un comando, l'intera pipeline viene abortita. Si ricordi che gli eseguibili **devono** provenire solo dalla directory designata in fase di avvio della debosh.
Durante l'esecuzione la debosh deve monitorare la quantità di dati che passano fra un comando e l'altro, e stampare ad intervalli di 250ms sul suo stderr una riga con il formato:

$$\mathrm{cmd}_1 \mid \mathrm{data}_1 \mid \mathrm{cmd}_2 \mid \mathrm{data}_2 \mid \ldots \mid \mathrm{data}_n \mid \mathrm{cmd}_n$$

in cui ogni cmd_i è il solo nome dell'*i*-esimo comando base (senza argomenti), e data_i è la quantità di dati che sono fluiti, fino a quel momento, fra il cmd_i e il cmd_{i+1}. La quantità di dati va espressa in notazione “umana” (o meglio: da informatici), utilizzando i suffissi kb, Mb, Gb, Tb come appropriato.

Suggerimento: per stampare ripetutamente la linea in questione, si può terminare la stampa con il carattere '\r' che corrisponde a un ritorno a-capo senza avanzamento di riga (codice ASCII CR).

Makefile

Il progetto dovrà includere un makefile avente, fra gli altri, i target *all* (per generare l'eseguibile della debosh), *clean* (per ripulire la directory di lavoro dai file generati), e *test*. Quest'ultimo deve eseguire un test della debosh, indicando come directory di comandi /usr/bin e come directory dei dati la directory $HOME/debosh (in cui si può assumere siano presenti alcuni file testuali di grandi dimensioni con nomi “file1.txt”, “file2.txt”, “file3.txt”, che non è necessario includere nell'archivio consegnato). Lo script di test può generare altri file all'interno di tale directory, prima di eseguire la debosh, secondo quanto necessario alle pipeline di test che possono essere create a discrezione dello studente (usando i comandi presenti nella directory /usr/bin di una tipica distribuzione Linux). È indispensabile che l'esecuzione dei comandi nelle pipeline di test fornite duri alcuni secondi, in modo da poter osservare il corretto funzionamento del reporting in fase di esecuzione come descritto sopra.

Note finali

Ci si attende che tutto il codice sviluppato sia, per quanto possibile, conforme POSIX. Eventuali eccezioni vanno documentate nella relazione di accompagnamento.
La consegna dovrà avvenire, entro il termine pubblicato, attraverso l'upload sul sito Moodle del corso di un archivio con nome *nome_cognome*.tar.gz contenente tutto il necessario. Scompattando l'archivio in una directory vuota, dovrà essere possibile eseguire i comandi *make* (con target di default) per costruire tutti gli eseguibili, e *make test* per eseguire il test e vederne i risultati. L'archivio dovrà anche contenere una breve relazione (3-5 pagine) in formato PDF in cui illustrate il vostro progetto.