Laboratorio di Sistemi di Elaborazione e Trasmissione dell'Informazione (SETI)

POSIX Intro #2

Lo scopo di questo laboratorio è familiarizzare con le system-call POSIX per la gestione di file e processi, abituandosi a consultare la documentazione. L'idea è "costringervi" a leggervi un po' di pagine di manuale e fare qualche prova.
Non abbiate paura di sperimentare, anzi!

Questo è un laboratorio "di riscaldamento" e il suo svolgimento NON deve essere consegnato.

Vi consigliamo di svolgere gli esercizi di questo laboratorio nell'ordine proposto, poiché dovrebbero essere in ordine di difficoltà crescente e, a volte, nel rispondere a una domanda potreste scoprire (se già non le sapete) informazioni importanti per affrontare quelle successive.

Esercizi

Scrivete, compilate e testate un programma C che...

1. Esegue il comando `ls -l`, attraverso l'uso di `fork(2)` ed `exec(3)`. Notate che `exec(3)` documenta una famiglia di funzioni e scegliete quella che vi sembra più comoda per il vostro caso.
   - Serve usare `wait(2)` in questo caso? Perché?
   - Sì, lo sappiamo che esiste `system(3)`; no, non potete usarla. In compenso, re-implementarsi `system` potrebbe essere un utile esercizio.
2. Chiede all'utente il nome di un file, usando per esempio `fgets(3)`, ed esegue `/bin/`*nome-inserito-dall'utente*
   - *Attenzione:* non deve cercare in tutto il `PATH`
   - Usate *valgrind*/*sanitizer* per controllare l'uso della memoria
     - In generale, abituatevi a testare i vostri programmi con input "strani" a piacere; per esempio, stringa vuota, stringhe contenenti caratteri non stampabili, stringhe lunghissime, ...
3. All'infinito:
   (a) stampa un prompt (per esempio, la stringa `"nano-shell $"`) sullo standard-output
   (b) chiede all'utente il nome di un file
   (c) se l'utente inserisce `exit` o *EOF* (premendo *ctrl-D* all'inizio di una nuova linea), esce con *exit status* `EXIT_SUCCESS`
   (d) esegue `/bin/`*nome-inserito-dall'utente*, dando un appropriato messaggio di errore se l'esecuzione fallisce. Per stampare il messaggio di errore, vedete `perror(3)`
   (e) aspetta la terminazione del processo figlio, vedete `wait(2)`
   - cosa succede se non utilizzate `wait`?

   Possibile migliorie: potrebbe cercare in tutto il `PATH`
4. Esegue il comando `ls -l >` *filename*; ovvero esegue il comando `ls` con argomento `-l` redirezionando il suo standard-output nel file di nome *filename*, che riceverete come primo argomento dalla linea di comando (ovvero, `argv[1]`); oltre alle system-call già usate per gli esercizi precedenti, vi serviranno `close(2)` e `dup(2)`, oppure `dup2(2)`.
5. Esegue il comando `ps aux | grep bash`, usando, oltre a quella già usate precedentemente, la system-call `pipe(2)`