LSV / ENS de Cachan
Architecture et Système

2014/2015
Feuille de TP n° 8

TP8

The course homepage is here:

`http://www.lsv.ens-cachan.fr/~schwoon/enseignement/systemes/ws1415/`.

You will find the slides and demonstration programs from the course and some other files for the exercise there.

Details of shell commands and C functions can be obtained by using the `man` command.

1 Communicating with signals

Write a program that first forks into a parent and a child. After this, the parent reads its standard input; whenever it sees 0 or 1, it transmits SIGUSR1 resp. SIGUSR2 to the child. The child should then output the inputs of the user in correct order.

The homepage provides a program skeleton (`pingpong.c`), which you can complete. (The parent terminates when the user types Ctrl-D, which makes `getchar()` receive EOF.)

Complete the program and test it carefully. For instance, enter multiple characters on one line, or feed input directly from the shell (`echo 1010101 | ./pingpong`). If you observe that the child does not answer with the correct sequence, try to correct your program.

Here are some potentially useful functions for accomplishing the task: getchar(3), kill(2), fork(2), signal(2), sigaction(2), pause(2), sigprocmask(2), sigsuspend(2)

2 Running programs in the background

The shell allows to run programs "in the background" by appending `&` to the command. For instance, `xcalc &` will start a calculator and return control to the shell.

Also, typing Ctrl-Z in the console will stop a current (foreground) process. The shell commands `fg` / `bg` will continue them either in the foreground or the background.

1. Create two instances of the X calculator from the same shell, by running them in the background. Then kill them from the command line.
2. Do as in (a), but by stopping/continuing them in the background.
3. Open a shell window and start `xcalc` in the background. Then close the shell (i) by typing Ctrl-D; (ii) clicking on the "closing" icon. What do you observe?
4. What happened in (c), and how can one prevent a program running in the background from being closed when the shell exits? Download the program `mywin.c` and protect it against this effect.

3 Programming a shell

Let us extend the shell that we started last week this time.

1. Change signal handling so that your shell is not interrupted when the user presses Ctrl-C. However, any process started from the shell should still be interruptible by Ctrl-C (at least unless that process takes its own measures to prevent it, of course).
2. (advanced exercice) Allow your shell to send programs into the background. This requires several steps:
   - Extend the parser to allow such a syntax.
   - Set/reset the foreground process group when launching a shell command. (Find out how using Internet search.)
   - Optionally, also handle Ctrl-Z, fg, bg, . . .