Fundamentos de Sistemas de Operação

MIEI 2017/2018

Laboratory session 3

Objectives

Enhance a command interpreter (*shell*) with IO redirection, background execution and signal handling.

Redirecting the standard IO

We want to add to our shell the ability to execute commands with redirected output or input to files. For that, the user can write a command line like:

command args... **>** *filename*

meaning that the standard output of the process running the command should be redirected to the given file. Similarly, a command line like:

command args... **<** *filename*

must execute the command with its standard input redirected to the given file.

Executing commands on the background

The Unix shell gives the user the ability to run commands on the background. This means the user can launch a command terminated with the character '**&**' meaning that the shell should immediately be ready to execute a new command. This is achieved when the shell process, after using fork to launch the new process, does not wait for the first command to exit but immediately prints the prompt and reads the user's next command.

Handling signals in a command interpreter

We want to modify the shell program so that it will continue to run when the user presses Ctrl-C at the keyboard. This action at the keyboard will send a SIGINT signal to the process were the default action (or handler) is to terminate the process. That is not desired for our shell so you must handle the signal in a convenient way.

You can use the examples with `signal` from the classes as a starting point.

Notice that when the shell receives the signal, the *read* operation (or *fgets*) may be interrupted without reading anything. Take that into account if needed.

Bibliography

Look at examples from classes 8 and 9.
Check section 5.4 from OSTEP book (ostep.org)
Read reference manuals (*man*) for `bash`, `signal`, `sigaction`, `pipe` and `dup/dup2`.