Process practicle work

Daniel Hagimont

Daniel.Hagimont@irit.fr

USTH

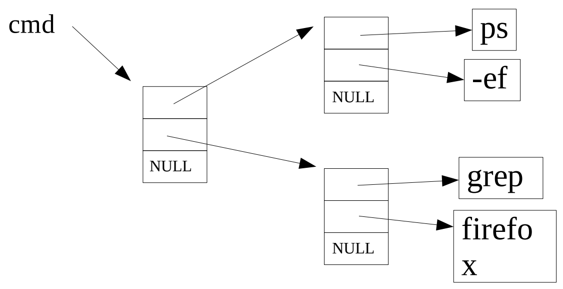
Objectives

The objective of this exercise is to use Unix process management APIs and Unix pipes for implementing a simple shell (such as bash). This shell should allow the execution of binaries (such as ls, pwd, grep or your shell itself) and it should also allow to cascade such executions with pipes.

Reading the shell input

You are given a module which implement the reading and analysis of the shell input. This module (readcmd.c/readcmd.h) provides a **readcommand** function which returns a pointer to a table of table of strings. This represents a sequence of commands where each command's output is linked to the input of the next command by a pipe. A command is an array of strings (*char* **), whose last item is a null pointer. A sequence is an array of commands (*char* ***), whose last item is a null pointer.

Here is the structure returned by **readcommand** for a "ps -ef | grep firefox" command:



Then, cmd[0] and cmd[1] can be directly passed as parameters to execvp(cmd[x][0], cmd[x]) ;

Instructions

You have to implement a shell program which will basically rely on the *readcmd* module, and the fork, execvp, pipe and dup2 system calls.

You are given template of program (shell-template.c) which implements the general loop of the program (and the "exit" command). Compile it and test it.

You have to implement :
- the execution of a simple command (e.g. a binary in /bin/)
- the execution of piped commands (e.g. ps -ef | grep firefox | wc -l) … do it for only 2 piped commands, it will be enough.