ECS518U Operating Systems

Lab 5: A Simple Shell and extension with capability to run external Linux shell commands or programs

This exercise is assessed (Must be assessed the latest by your Week 7 lab but hopefully you will get it done before ☺).

# Testing, Demonstration and Assessment

You are expected to:

* Answer the questions in the Answer sheet and be ready to explain your answers.
* Be ready to answer questions about the sample programs we have made available.
* Be ready to demonstrate your own shell working.
* Be ready to answer questions about your code. **The code is expected to be clearly commented and properly formatted.**

**PART A**

## Your Shell: Features Completed

Complete a table similar to the following showing the status of each of the required commands. You should describe how you resolved any ambiguities you found in the implementation of each command. This is mostly for your own record of what was done in the implementation of the Shell.

|  |  |  |
| --- | --- | --- |
| **Command** | **Status**  **(Complete / Partial / Not Complete)** | **Notes**  (If you found any ambiguity in the specification for a command, say how you dealt with it. If something is left not implemented, make a note. Else say None) |
| files | Complete | None |
| info | Complete | None |
| copy | Complete | None |
| delete | Complete | None |
| where | Complete | None |
| up | Complete | None |
| down | Complete | None |
| exit | Complete | None |

**Question: Linux Shells: How do they execute commands/programs?**

Describe what happens in terms of processes when you run a command/program in a Linux shell (the answer can easily fit into 2-3 sentences). Copy and paste output of appropriate ps and/or pstree commands that explain what happens in terms of processes.

You must relate this to the lecture material on process control that was covered during the Week 3 lecture, so we obviously expect something more than, for example, (“the program that corresponds to the command is executed”).

**Ideas:** Open one or more terminals, execute some command or program and try to find a way to examine what happens in terms of processes. You can use commands such as ps (look at what command line options give you information that helps you, e.g. ps –ef, ps -Oppid), pstree, etc. to see what processes are created. Try to figure out how the command or program you ran relates to the shell program. You can also research online but you must relate all this to lecture material on process control (e.g. fork) that we covered in Week 3.

**Answer and pasted output:**

If the executable is not in the path, the command will fail.



If it is in the path, the shell will launch a child process that executes the command.

In this example, python was launched from pty0 and ps was executed in pty1

Text

Description automatically generated

**PART B**

**(These questions are in addition to code that meets the requirements for Part B – answering these questions with no implementation will not give you full credit for this Part of the lab)**

## Step 1: Behaviour of ‘myRunShell0.py’ as supplied (i.e. without any changes from you)

Answer the following questions about the behaviour of my\_run\_shell\_0.py

|  |  |
| --- | --- |
| **Question** | **Answer** |
| **What happens** when the ‘my\_run\_shell\_0.py’ script is used to run an external program (e.g. a Linux shell command)? | The program is executed, and the shell is terminated |
| **What requirement** does this behaviour not implement? | Display exit code of the external program |
| **Why does this behaviour occur?** (Refer to the documentation of os.execv()) | Execv() replaces the current process. When execv() is invoked, the shell is no longer there. When execv() finishes, the process terminates. |

## Step 2: Use of os.fork()

Answer the following questions about the use of os.fork():

|  |  |
| --- | --- |
| **Question** | **Answer** |
| **What happens** when the os.fork() function is executed? | A copy of the python process is created. In the child process, fork() returns 0. In the parent process, fork() returns the child pid. Both processes run concurrently. |
| **What Python function** is used to make the parent process wait for the child process to complete? Why is this good practice? | Os.wait()  If we don’t wait for the child process, the parent process might exit before the child. The child will then become a zombie process |