CS4760 **Operating Systems**
Assignment # 1 Due Date: September 10, 2018

Important: It is required that you submit on the `hoare` student OS server, so it is a good idea to do your projects there.

Unix System Calls and Library Functions

The goal of this homework is to become familiar with the environment in hoare while practicing system calls. I'll like to see the use of `perror` and `getopt` in this submission. The base of this is to do Exercise 3.8 (p. 87) in your text by Robbins/Robbins.

This exercise expands on the process chain of Program 3.1 in the book, so it would be a good idea to study that code well. The *process chain* is a vehicle to experiment with `wait(2)` and with sharing of devices. All of the processes in the chain created by Program 3.1 share `stdin`, `stdout`, and `stderr`.

This project consists of several parts:

1. Set up a Makefile and a basic framework for your project. I will have a resource linked for helping you with using Makefile. This Makefile should, when given no options, compile your project into an executable called ass1. In addition, it should, when called with Make clean, remove all generated object files. The makefile should also contain constants determining what compiler flags and what compiler to use.
2. Set up version control. If you are not familiar with one, RCS is a simple one. I will have a resource linked for you. It would be a good idea to set both of these up before trying to do any other part of this project. To test this, I would suggest starting by modifying Program 3.1 to terminate immediately after taking command line arguments (you do not want to be running the program until everything else is set up to work in Test four). For later projects, I will require that RCS or other version control that you use inserts into the file version information, so you can either work on that now or have it to look forward to.
3. Modify the code of Program 3.1 in the textbook to take several command line arguments through `getopt`. Your program should take three options, either -n x, -h, or -p. The -h option will display a help message indicating the type of input it expects and then the program should terminate. If it takes -n x, it should store that x into a variable for use for the rest of the program. If it receives -p as an option it should use `perror` to generate a test error message as described in the next task. Test and ensure that these work.
4. Test the generation of error messages for future projects. You are required to use `perror` for this. The format of the error message itself should be:

   ```
   my_prog: Error: Detailed error message
   ```

   where `my_prog` is actually the name of the executable (`argv[0]`) that you are trying to execute. This should not be hardcoded, so no matter what the executable name is, it will output the correct result. This is necessary so that later on when we are working with multiple executables it is easier to find where the problem is.
5. Follow the experiments in the textbook in Exercise 3.8. Include the answers or comments to all the questions (1-8) in a file called Answers (make sure the case is correct).

For all tasks, you should be adding to a README file. This file should contain information on how to compile and run your project, as well as any problems or comments you have on its performance. If you found something particularly hard or easy, let me know. This can be a useful area for me to get feedback from you on how I can help you or others in the future.

What to handin

I will want you to hand in an electronic copy of all the sources, `README`, Makefile(s), and results. Create your programs in a directory called *username*.1 where *username* is your login name on hoare. Once you are done with everything, *remove the executables and object files*, and issue the following commands:

```
chmod 700 username.1
```

```
cp -p -r username.1 /home/hauschild/cs4760/assignment1
```

Do not forget `Makefile` (with suffix or pattern rules), `RCS`, and `README` for the assignment. I do not like to see any extensions on `Makefile` and `README` files. Before the final submission, perform a `make clean` and keep the latest source checked out in your directory.

You do not have to hand in a hard copy of the project. Assignment is due by 11:59pm on the due date.