CS4760 **Operating Systems**
Assignment # 1 Due Date: September 22, 2020

Important: Please do all assignments on `hoare`

Unix System Calls and Library Functions

The goal of this homework is to become familiar with the environment in hoare while practising system calls. I'll like to see the use of `perror` and `getopt` in this submission.

Do Exercise 3.9: **Process Fans** (p. 88) in your text by Robbins/Robbins.

The exercise expands on the fan structure of Program 3.2 in the book through the development of a simple batch processing facility, called `runsim`. The program being developed in this exercise is a precursor to building a license manager.

Program 3.2 creates a fan of n processes by calling `fork(2)` in a loop. So, you will need to study that code well. The *process fan* is a vehicle to experiment with `wait` and with sharing of devices.

With the use of `perror`, I'll like some meaningful error messages. The format for error messages should be:

```
proc_fan: Error: Detailed error message
```

where `proc_fan` is actually the name of the executable (`argv[0]`) and should be appropriately modified if the name of executable is changed without recompilation.

I'll like the process to be executed by the following command line:

```
proc_fan -n 20
```

where `-n` is the option to indicate the max number of processes generated.

I'll like some meaningful error messages. The format for error messages should be:

```
proc_fan: Error: Detailed error message
```

where `proc_fan` is actually the name of the executable (`argv[0]`) that you are trying to execute.

Suggested Implementation Steps

1. Set up your version control, along with your Makefile and a source file
2. Write code to parse options and receive command parameters using `getopt(3)`
3. Write a simple program that your code will "fan", called testsim in the assignment. This can simply echo out the command line arguments given it.
4. Now start implementing `proc_fan` by having it take one line of input and forking off this process and execing it, waiting for it to finish
5. At this point start implementing the limits and forking of additional processes as in steps 5-6 of the exercise.

Grading

1. *Overall submission: 25 pts* Program compiles and is able to solve the assigned problem.
2. *Command line parsing: 10 pts* Program is able to parse the command line appropriately, assigning defaults as needed.

3. *Use of perror* Program outputs appropriate error messages, making use of perror.
4. *Makefile: 5 pts* A makefile is used and it supports the ability to clean up all executable files to allow a full recompilation.
5. *README: 5 pts* Must address any special things you did, or if you missed anything. If your program segfaults when run as a normal thing, this should be mentioned.
6. *Conformance to specifications: 30 pts* Program works correctly and meets all specifications.
7. *Code readability: 10 pts* The code must be readable, with appropriate comments. Author and date should be identified.
8. *Version control: 10 pts* You should be using version control, which should also be referenced in the readme

What to handin

Create your programs in a directory called *username*.1 where *username* is your user name on hoare. Once you are done with developing and debugging, *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), your versioning files, and `README` for the assignment. If you do not use version control, you will lose 10 points. I want to see the log of how the program files are modified. Therefore, you should use some logging mechanism and let me know about it in your `README`. You must check in the files at least once a day while you are working on them. Omission of a `Makefile` (with suffix rules) will result in a loss of another 10 points, while `README` will cost you 5 points. 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.