CS4760 | **Operating Systems**

Assignment # 1 | Due Date: February 06, 2019

Important: Please do all assignments on `hoare`

Linux System Calls and Library Functions

The goal of this homework is to become familiar with the environment in hoare while practicing system calls. We will be using `getopt` and `perror` as well as fork().

This project is just reading some integers from an input file and writing them out in reverse order into the output file. All that is required of you is to use `fork()`, parsing the options, and using `perror()`.

Your project should consist of one program, which will `fork` off versions of itself to do some file processing. It will start by using some command line arguments. You must implement at least the following command line arguments using `getopt`:

```
-h
-i inputfilename
-o outputfilename
```

The `-h` option will display all legal command line options and how it is expected to run, as well as the default behavior. If input and output filenames are not specified, the defaults should be `input.dat` and `output.dat`.

Once you have parsed the command line arguments and validated them, then you should attempt to open the input file. The input file should start with a number on a line by itself, with that number indicating the amount of times you will be required to do a task with copies of your process using `fork`. Each task list will follow in the file, which will consist of an integer on one line (representing the amount of numbers that will follow) and that many numbers. An example of this input file is below:

```
3
6
3 6 107 8 1 3
4
1 3 50 4
7
5 3 5 7 8 9 1
```

The main process (parent) should read the first line of the file. Once it has read that line, it should then go into a loop based on that number, with each iteration of the loop forking off a copy that will then process the next two lines. Once that child has finished its work (defined below), it will write some data to the output file and then terminate. At that point, as the parent detects its child has terminated, it should do another iteration of the loop until done. After all children have terminated. the parent should write the PIDs of all of its children that it launched, as well as its own PID to the output file.

When a child process starts, it should read the next line of the file, which will tell it how many numbers to read afterwards. We see in our example file that the first forked child would read a `6`. It should then read that number of integers and put them into a stack. After putting all of the numbers into a stack, the child should write its PID to the output file, followed by those numbers in reverse order. For example:

```
13278: 3 1 8 107 6 3
```

After this has been done with the example file given above, we would expect an output file to look as below (with different PIDs, of course):

```
13278: 7 3 1 8 107 6 3
```

```
13281: 4 50 3 1
13294: 1 9 8 7 5 3 5
All children were: 13278 13281 13294
```

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

```
logParse: Error: Detailed error message
```

where `logParse` is actually the name of the executable (`argv[0]`) that you are trying to execute. These error messages should be sent to `stderr` using `perror`.

It is required for this project that you use version control, a `Makefile`, and a `README`. Your `README` file should consist at a minimum of a description of how I should compile and run your project, any outstanding problems that it still has, and any problems you encountered. Your `Makefile` should use suffix-rules or pattern-rules and have an option to clean up object files.

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:

```
% cd
% chmod 755 ~
% ~sanjiv/bin/handin cs4760 1
% chmod 700 ~
```

Do not copy and paste those commands from the PDF *of the assignment. Type in the commands.*

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.