CISC 3350 Homework Assignment - 4

Ignoring the SIGHUP and SIGQUIT signals

This assignment involves the writing of a simplified version of the shell utility nohup called my_nohup. The utility is used to prevent processes from being killed by the SIGHUP or SIGQUIT signals when the user logs off from a session.

First read the man pages on nohup to understand how the utility works. Also read the man pages on the function isatty().

Your code will carry out the following actions:

1. If the user supplies no command line arguments (other than the name of the executable) write an informative message to stderr (STDERR_FILENO) and exit with an error value.
2. Use the function isatty(int filedesc) to determine if the standard output file descriptor is associated with a terminal file. If it is, your code needs to redirect standard output to the file nohup.out (which may already exist, then you append to it) nohup.out needs to be created or opened for writing. If you need to create the file, give it permissions such the owner has read/write permission. When this section of code is completed, if isatty() returned true, the file descriptior originally assigned to STDOUT_FILENO should be associated with nohup.out. If you are unable to open/create nohup.out, write an informative error message and exit with an error value.
3. If standard error, STDERR_FILENO, is associated with a terminal device, you also need to reassociate file descriptor 2 with nohup.out.
4. Ignore SIGHUP and SIGQUIT. You must do this using sigaction() and the appropriate macros. If you are unable to ignore these signals, write an informative message to stderr and exit with an error value.
5. The new program is the first command line argument to my_nohup. exec*() this program and its arguments, *e.g.*
   ```
   $ my_nohup testsim 5 10
   ```

   In this example, my_nohup would use exec*() to execute the program testsim with the command line arguments 5 and 10.

If the exec*() fails, write out an informative message using either perror() or strerror(). Exit with an error value.

Note: the standard output of the new program will be written to nohup.out.

An example showing how to redirect stdout to a file:

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

int main(int argc, const char *argv[])
{
   int out = open("nohup.out", O_RDWR|O_CREAT|O_APPEND, 0600);
   if (out == -1) {
     perror("opening cout.log");
     return 1;
   }

   if (dup2(out, fileno(stdout)) == -1) { // copy fd of out into
the fd of stdout (==1)
     perror("cannot redirect stdout");
     return 1;
   }

   close(out); // not used anymore

   // from here on, all standard output (within this process),
will be redirected to nohup.out

   return 0;
}
```

How to test?

I found that logging off by exiting from or closing a terminal always ends `nohup` on my Ubuntu VM. So I tested it in this way instead:

- Getting two terminals running with the current folder being where you have your binaries for both `my_nohup` and `testsim`
- In terminal 1, run "`$ ./my_nohup testsim 5 10`"
- Immediately in the other terminal (terminal 2), type "`$ ps -u`" and you should see that `testsim` is running. Take note of its pid
- In terminal 2, type either or both "`$ kill -HUP pid`" and "`$ kill -QUIT pid`" (the pid being the one you just got). These explicitly send SIGHUP and/or SIGQUIT to the process
- Still in terminal 2, issue "`$ ps -u`" and you will see that `testsim` is still running (since SIGHUP and SIGQUIT are both ignored)
- You can look at the `nohup.out` file later to see the record of uninterupted run of `testsim`
- Of course, if you run `testsim` without being preceded by `my_nohup`, either SIGHUP or SIGQUIT will end the process

What to submit

- Please put all your functions in a file called my_nohup.c and e-mail it to me.
- Any C library function can be used.
- Grading will be based on if your program compiles and generally works in: handling all the redirection for stdout and stderr, ignoring the relevant signals, executing the command properly, with proper error-checking and having no undesirable side effects.
- Please don't copy code.