COS 350: Program #5: mysh

Objectives: Working with processes: fork, exec, waitpid. Also accessing environment variables.

You will write a simplified Unix command shell. You should start with the small shell version 1 from the book. You will be adding the following functionality to the shell.
- Change the prompt to something different.
- Add a built-in command: exit [value]
    - Exit the shell returning the value if specified.
- Add a built-in command: cd [dir]
    - Change to the specified directory or, by default, to the user's home directory.
- Execute commands in the background if the command line ends with an '&' character.
    - You do not need to provide job control as the shell does, however when jobs are started in the background, the shell should print a message indicating that the job was started and its PID.
    - Your shell does need to wait for these processes at some point so that they don't become zombies. You can use waitpid() to check if any child has completed. Waitpid() provides extra options to specify which process or processes to wait for and to return immediately if no processes have completed.
    - You should check for completion of background processes after each command, and print a completion message including the PID as they complete.

Suggestions:
- You can find the starting code in the course directory: /usr/class/cos350/prog5
- Since you are starting with someone else's code, take some time to study it and avoid changing their code unless you really need to.
- When you run the book's code, they have disabled Ctrl-C and Ctrl-\ so you won't be able to use them to stop the shell. You can however use Ctrl-D which indicates the end of keyboard input stream. Their code exits on that. At some point you will also implement exit.
- Check the error values of all system calls and library functions. This will help reduce your debugging time because you will find out immediately if a system call is failing.
- Read the man pages to find out how things work.
- Use waitpid() rather than wait(). Wait() waits for any child process, whereas waitpid() can be specific. This will be important when mixing foreground and background jobs and trying to wait for the foreground job to finish.
- Test complicated situations that might expose bugs. For example try running several long running processes in the background at the same time that you are running other commands in the foreground. Improper behavior in this situation has been the most common problem with student code in the past.

What to turn in:
For each assignment you will turn in a written part in class as well as submitting your code electronically.

Written part:
- Your name.
- A discussion of any incomplete parts, known bugs, deviations from the specification, or extra features in your program
- A printout of your code. Use a2ps to save paper.
- A printout of your Makefile
- The filled in testing sheet on the last page
- A log of the following testing results. (I recommend using script to capture it.)

`script`	*start script*
`mysh`	*start your shell*
`ls`	*testing a couple commands*
`date`	
`bogus`	*a non-existent program*
`exit`	*testing exit*
`mysh`	*start your shell again*
`exit 7`	*testing exit status*
`echo $?`	*this bash command shows the exit status of the most recent command (which should be 7)*
`mysh`	*start your shell again*
`pwd`	*where are you*
`cd ..`	*change dir - relative path*
`pwd`	
`cd /usr/bin`	*change dir - absolute path*
`pwd`	
`cd bogus`	*try changing to non-existent directory*
`pwd`	
`cd`	*default – change to home dir*
`pwd`	
`sleep 10 &`	*background job, should see a started message*
`ps –l`	*should show sleep is sleeping*
(wait 10 seconds)	
`ps –l`	*might see the zombie, should see a finished message*
`ps –l`	*zombie should be gone*
`exit`	*exit your shell*
`exit`	*exit again to exit script*

Electronic submission:
- Before you submit your program, compile and test it on a **linux** machine in the lab.
- From a machine in the lab run the program "submit" to submit your files.
- Submit your source code (**mysh.c**) and any other source files, a **Makefile**, and your executable (**mysh**) to a directory named: **prog5**
- **Do not in any way combine, compress, zip, or tar your files!**

Grading: see the testing chart on the next page
- Programs not meeting the style guidelines will be returned ungraded.

COS 350 Program 5 Testing - please fill out and turn in this testing sheet with your written submission.

Name(s): _____

	Pts	Works	Mostly Works (explain)	Occasionally Works (explain)	Not implemented	Explanation
changed prompt (very easy 10 pts)	10					
correct command execution (this should still work)	10					
exit	10					
exit value	10					
cd dir	15					
cd	15					
starting jobs in background	15					
notifying that background jobs are done	15					