Overview
You are to write and test a C/C++ program that implements a simple shell whose executable is called `myshell12` which is an extended form of `myshell10`. It allows only one command per line and supports the same built-in (internal) commands as `myshell10`:

- **chdir** `<Pathname>`: Change the current directory to `<Pathname>` which can be an absolute or relative Unix pathname. If `<Pathname>` is not given, change the current directory to the path given by the environment variable `HOME`. The current directory should be maintained in an environment variable called `PWD`.

- **environ**: Display the environment; i.e., the name-value pairs, one per line just like `printenv` does.

- **echo** `<Word>` `<Word>` ...: Display the arguments followed by a newline. Multiple spaces/tabs may be reduced to a single space.

- **quit**: Quit the shell.

Here are the other features of `myshell12`:

a) Again, multiple spaces/tabs may be reduced to a single space.

b) The command line prompt should be the pathname of the current directory followed by the three character sequence `''` (i.e., >, >, space).

c) There are four other built-in commands: `bg`, `wait`, `status` and `set`.

   - "`bg` ...": The remainder of the line should be run in the background with the word following `bg` treated as a command.

   - "`wait`": The shell should wait for all backgrounded processes to complete.

   - "`status`": Display the status of child processes. The display should show the process id of running processes and their run state. Their run state should be "RUNNING". The display should show the process id, system time, user time and run state of the `latest` terminated processes. Their run state should be "TERMINATED". The latest terminated processes are those which have terminated in the last 60 seconds (wall clock time). (See the getusase man page for how to get the system and user times.)

   - "`set`": Set a variable to a value. A variable name begins with a letter and consists of letters, digits and the underscore character. For example, `set XYZ 32` sets the variable `XYZ` to the string `32`. The value of `XYZ` is denoted by `$XYZ`; i.e., `$` denotes "the value of". The Web page will contain some simple code for symbol table manipulation.

d) A non-built-in command is assumed to be a Unix executable that can be found in a directory listed in the PATH environment variable.
Note that there is simple variable substitution, but there is no filename substitution nor command substitution. The shell should continue to run even when there are errors. However, there should be some informative message displayed when an error occurs. Built-in commands should be checked for the correct number of arguments.

**Implementation Notes**

In this implementation, you can assume that the system will be small and therefore simple data structures are appropriate (i.e., there is no need at this time for sophisticated data structures). It is up to you to determine what you will need, but remember that simplicity will be a virtue in this assignment.

**Additional Guidelines**

- Code readability is of the utmost importance. The Web page will contain a summary of coding guidelines that you should follow in spirit. I am not rigid about these guidelines, but unreadable code will be penalized.

- All actual system calls should be wrapped so that any fatal errors will cause an error message to be displayed followed by an exit. I will not examine any solutions that do not follow this rule. By convention, the wrapped system call name will be the same as the actual system call name except the first character should be capitalized (e.g., Fork is the wrapped version of fork).

**What to Submit**

The CS422S Web page contains a link to the documentation template. You should complete the template and submit it in both hardcopy AND electronic form. Submit the completed documentation template AND a listing of the source code. The electronic submission (described below) should include the completed documentation template, the source code, the Makefile, test scripts, and test output.

**Words of Caution**

Here are some observations from my years of experience with projects like this:

- **The documentation template is non-trivial.** Do not expect to complete it in less than an hour. Furthermore, working code but no documentation is almost useless. So, don’t forget to fill out the documentation template.

- Trivial bugs can consume tens of hours of time. Yes, I said tens of hours, not just hours. You need to start small, test the control structure, and incrementally add features on top of what seems like rock solid code.

- Keep different versions of your ”rock solid code versions” so that you can rollback to and recover from a stable version. This also helps if you mistakenly delete your latest source code!!

- Don’t ignore error messages and think they will disappear on their own. They don’t. They just come back and bite you when you least want to be bitten.

- Try to understand the origins of your bugs rather than always doing trial and error changes. (Some trial and error may be appropriate in small test cases.)
• If the approach you are taking seems like it will be a nightmare to implement, then don’t implement it. Find a better way or better understand the system calls you are trying to use.

• The more lines of code you have, the more chances for bugs to appear.

• Have a plan. Don’t try to do everything at once.

Electronic Submission
The end result should be that you mail to kenw@arl.wustl.edu a single shar (shell archive) file containing your files. Do NOT submit object code or executables. The following commands will create a shar file named A.shar containing the files myshell2.c, in1.txt, out1.txt, and other files and then send mail to me:

    shar README myshell2.c in1.txt out1.txt ... > A.shar
    mail kenw@arl.wustl.edu < A.shar       # mail is usually in /bin

The README file is the completed documentation template.

Late Policy
You can submit this programming assignment one week late for a 20% penalty. Note that you should submit something even if the final version still has bugs.