Overview

You are to extend your Project A to remove some restrictions and add new features. The command has the following format:

    xssh [-x] [File] [Arg] ...

The difference here is that the optional File argument indicates an xssh script. The File argument can be followed by zero or more arguments that are assigned to the variables $0, $1, etc., just as in normal shells.

Here are the changes to existing features and commands:

- **xssh** should continue to run even in the face of errors.
- All builtin commands should be checked for the proper number of arguments if possible.
- The command line prompt is simply the three character sequence '>> ' (i.e., >, >, space); i.e., there is no pathname.
- Environment variables are inherited from the parent and can be more than the three we had in Project A.
- Variables can be of arbitrary length instead of just one character.
- $. is a special variable which is used to terminate a variable name. $XY$.Z is the string resulting from the concatenation of the value of the variable XY and the value of the variable Z.
- A blank line has no effect (the null command) except that it should act like any other command with respect to terminated background jobs (described later).

This version of **xssh** should support pipelines and responds to the terminal interrupt character (normally ctrl-c). Here are some specifics:

- A pipeline in the foreground has the same syntax as in other shells; i.e., 'Command | Command | ... | Command' is a pipeline. Note that the Commands can have arguments.
- A pipeline in the background is expressed as 'bg Command | ... | Command'; i.e., it combines our earlier bg command with the pipeline symbol '|'.
- The terminal interrupt character (normally ctrl-c) generates the interrupt signal (SIGINT). **xssh** should terminate the foreground process(es) and return to the command line prompt. However, it should not terminate background processes.
- The user should be notified about any background jobs that have terminated right after the command line prompt is displayed.
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. The electronic copy is due by midnight Nov. 29. The hardcopy can be submitted in class on Nov. 29 or to my office by noon Nov. 30. This submission is worth 50 points.

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 Ax.shar containing the files xssh.c, in1.txt, out1.txt, and other files and then send mail to me:

shar README Makefile xssh.c ... in1.txt out1.txt ... > Ax.shar
mail -s Ax.shar kenw@arl.wustl.edu < Ax.shar # mail is usually in /bin

The README file is the completed documentation template.

Late Policy
There is no late submission date.

Extra Credit
The maximum extra credit that you can get is 10 Points. Here are some possible extra credit topics:

- Job Control (10 Points): Respond to ctrl-z in a standard fashion, and provide support for the bg, fg and jobs commands.
- Some other topic. This has to be negotiated and should extend your understanding of a shell facility. It can not be just arbitrary work.