Preview:

Note that you should submit the solution to this homework by the coming Tuesday, 
not Thursday. All of the problems in this homework will prepare you further for Project A.

Project A is listed as a reading assignment because we will discuss the assignment on Tuesday. Problem 3 begins to address the design of a part of the shell in Project A and is a vehicle for focussing our discussion.

The reading (Sections 2.3.1-2.3.3) is preparation for the next set of topics on synchronization and deadlock. Try to identify the main concepts and their relationship to each other. Don’t worry so much about understanding all of the details right now. On the surface, they may seem simple and obvious, but in fact, are quite deep and intellectually challenging.

**Problem 1** (0 Points)

This problem considers the Forward Forking (Method 1) of creating a pipeline where each new process creates the next process in the pipeline. In the description below, we use the notation in Homework 4, Problem 3 where $x[0]$ is the interactive shell. Suppose that the shell is required only to handle the case of only one pipeline and no stdin/stdout redirection; i.e., "$C[0] | C[1]".

Furthermore, suppose that the pipeline evaluation function has the following interface:

```c
void eval_pipe( char **argv[] );
```

where `argv[i]` is a pointer to the ith pipeline command stored in argv format; i.e., `argv[i][0]` is the command, `argv[i][1]` is the first argument, etc. of the ith command.

a) The diagram for Method 1 should indicate that the interactive shell $x[0]$ should wait for the first command $C[0]$ to complete but not the other commands in the pipeline. Why doesn’t the interactive shell in Method 1 wait on all the child processes as in Method 3 where the interactive shell creates the entire pipeline?

b) In Method 1, can $x[0]$ create all pipes as in Method 3? Explain.

c) Is it simpler for $x[0]$ to create all pipes or for $x[i]$ to create pipe $i - 1$? Explain.

d) Draw a diagram of the file descriptor table for each of the processes executing commands $C[0]$ and $C[1]$.

e) Give the C/C++ code for evaluating the pipeline assuming that there is always one pipe and all commands are external commands.
Problem 2 (0 Points)
The following is a sequence of commands (with line numbers) entered in rapid succession to the bash shell:

1  cat /etc/passwd | sleep 60 &
2  ps -o pid,ppid,pgrp,tpgid,cmd
3  ctrl-c          # ctrl key and c key together
4  fg
5  ctrl-c

a) What relationships will be shown by the output from line 2 regarding the process hierarchy, the process group(s), and the controlling terminal?

b) What will be the effect of line 3?

c) What will be the effect of line 4?

d) What will be the effect of line 5?

Problem 3 (6 Points)
Consider the example `xssh` command:

```
ls -l .. | grep Jan | sort
```

Answer the following questions regarding this example command which I will denote by `C`. Keep in mind that a classmate should be able to understand your solution after a few minutes of reading. As part of our Tuesday class, we will attempt to select a good solution ... and maybe even a beautiful solution.

a) Draw a picture of the important relationships right after `xssh` has launched the processes required to evaluate `C`. List the key features and concepts shown in the diagram.

b) If a classmate looked at an implementation of your concepts in Part a, what are the main things that he/she should see?

c) After `C` is read from stdin, it is usually stored as a string. It would be useful to transform `C` into a representation that would make it easy for evaluation routines to process. Draw a picture of such a data structure and explain why the data structure would make it easy for evaluation routines. You can use an abstract data structure rather than a concrete data structure if you wish. I use the term abstract data structure to mean a data structure that is devoid of implementation language details.