KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

Information and Computer Science Department

ICS431: Operating System

Project One (Term 062)

Due Date: April 21st 2007

Project Submission:

You are requested to submit both hardcopy and softcopy. Hardcopy should contain cover page, your source code, and sample outputs. Softcopy (source code only) will be submitted through WebCT.

Project Rules:

1. Submitting your project after the deadline will result in ZERO mark.
2. Any cheating in the project will result to ZERO mark.
3. THERE WILL BE NO DIFFERENTIATION BETWEEN THE PERSON WHO DOES IT AND THE PERSON WHO LETS OTHERS TO DO IT. Both will have the same penalty.

Project Question:

You are going to implement a simple command line shell. The shell application takes no arguments itself. It reads "commands" from the standard input.

Part 1: The shell must read commands, parse input and fork off programs using **execvp/execlp**

Part 2: You must add "*job control*" to the shell. This means you have to recognize the command "**jobs**" and print out the list of remaining background jobs. You must also recognize the "**kill**" command to send a "kill signal" to a background job.

Command:

A "command" is a line composed of words separated by any amount of "whitespace" (spaces, tabs) and ended by a single newline. The following logic is applied for each input line:

1. If the input line is empty (after removing whitespace), the next input line is fetched
2. If any previous background child process has finished execution, a message of the format "**Child process XXXXX exits**" is printed, where "**XXXXX**" is replaced by the process id.
3. An input line input no more than 8096 bytes is read (the **fgets** library routine can be used)
4. The line is broken into "tokens". Tokens are separated by whitespace. The library routines **strspn, strtok** and **strcspn** are useful for this. You must accommodate arbitrary whitespace (*e.g.* "ls ___-lt___/tmp" where "_" is a space). White space is spaces or tabs.
5. If the last token on the line is "**&**", this denotes a "background command".

The program to execute in each command is specified by the first token of input.

Part 1:

1. Fork a child process
2. Use the **execvp/execvl** system call to cause the indicated program specified in the command to be executed. If the command is a "background command", do not pass the final "**&**" to the program. If the **execvp/execvl** returns an error, you must report that the command can not be found and the child process should **exit()**.

3. If the command is a background command, continue execution. If the command is ***not*** a background comand, use the **waitpid** system call to wait for the process that was just forked to finish. Its possible, that **waitpid** returns because of a previous command that was started in the background and has since then finished. Make sure you print its status and then wait again for current foreground command.

Part 2:

In this Part, you must implement two "pseudo commands".

The first pesudo-command is "**jobs**"; this lists the set of background tasks that have not finished. Keeping and displaying this information simply involves an array of structures that keep information about all background tasks that were started. As background tasks finish execution, their entries should be removed from the array. You can assume that we will never have more than 100 background tasks.

The second pseudo command is "**kill**", which takes an integer as a second argument. The integer specifies a background job / process to be killed. This command should report an error if the specified process is ***is not*** that of a background process. If it is a valid process id, the shell should send a "kill signal" to that process using the **kill** system command.