

Homework 2  
Operating Systems, V22.0202  
Fall 2007, Professor Yap

Due: Mon Oct 1

- Please read questions carefully. When in doubt, please ask.
- The written homework is to be submitted in hardcopy during class, but the programming part sent to us in a single file (as detailed below) by midnite.

---

**Question 1** (10 Points)

To understand Unix, you need to know its special characters. For each of the following, describe one situation where they are used. They are all related to processes.

CHARACTER	NAME
ctrl-D	control-D
ctrl-Z	control-Z
	pipe
<, >	redirections
&	ampersand

◇

**Question 2** (28 Points)

For each of the following shell commands, (1) describe the meaning of the command, and (2) describe a situation where it is useful. Note that you can string several shell commands into one command if you separate them with a semicolon (;).

HINTS: We suggest you actually test these commands on your system. In Unix systems, you can get an online manual for a particular COMMAND by typing `man COMMAND`. For instance,

> `man ls`

will tell you all the options available for `ls`. Also try `man man`.

1. `ls -a`
2. `ls -tl /bin`
3. `ls /bin/m*`
4. `ls -sF`
5. `ps -s`
6. `ps -sW`
7. `echo "Reading This File"`
8. `which ls`
9. `file hw1.c hw1.o hw1.exe`
10. `wc hw2.cMakefile`
11. `pushd "/cygdrive/c/Program Files"; ls`  
NOTE: this will not work if you are not in Cygwin. Instead use `pushd /usr/bin; ls`.
12. `popd`
13. `alias ls="ls -sF --color=tty"`
14. `ln -s "/cygdrive/c/Program Files/Vim/vim71/gvim.exe" .`

◇

**Question 3** (5 Points Each)

- (i) Exercise 2.5, page 73 of Silberschatz. (File Management)
- (ii) Exercise 3.1, page 116 of Silberschatz. (scheduling)
- (ii) Exercise 3.2, page 116 of Silberschatz. (context-switching)

◇

**Question 4** (70 points, Programming Part) Write a toy shell program called `tsh.c` that executes unix commands. It should know how to process command lines containing the special tokens "&" and ";", but not "|".

We will provide the basic routine called `parseCommand` and also the skeleton of `tsh`. HINT: we suggest that you first implement "&" to make sure it works correctly before implementing ";".

Extra Credit Features:

- (1) Allow the user can change the prompt string.
- (2) Can you make "cd" work properly?

Enclosed is a tar file called "hw2.tar" that contains the files

`Makefile-hw2`, `parseCommand.c`, `parseCommand.h`, `tsh.c`

When you have unpacked them, we suggest you rename `Makefile-hw2` to `Makefile` (why?) so that you can easily use it. You are encourage to modify and add to this Makefile to suit your needs.

You should not modify the `parseCommand` files, but just use them in `tsh.c`. We have provided the bare skeleton in `tsh.c`, which you need to embellish in order to complete the assignment in this question.

We want you to do separate compilation of `parseCommand.c` and `tsh.c` (see the Makefile to see how it is done).

**SUBMISSION REQUIREMENTS:** Up to 20 percent of this question may be deducted for non-compliance. These rules apply for all programming assignments (with the appropriate modifications).

- Make sure that your program is well-documented (use long and short comments as appropriate)
- Include a README file (you can copy it from previous homework!). As usual, the file should contain
  - (1) your name,
  - (2) NYU ID,
  - (3) email,
  - (4) phone Contact,
  - (5) the operating system environment for your work,
  - (6) acknowledgement of sources (otherwise it would be plagiarism),
  - (7) any collaboration with others (if none, please say so),
  - (8) any other information you like us to know,
  - (9) and include the following statement: *"This submission represents my own work."*
- To submit, you should send us one single tar file. Create a tar file called `h2.tar` containing `README`, `Makefile`, `parseCommand.c`, `parseCommand.h`, `tsh.c` and any other necessary files you need. (Type "tar cvf h2.tar README Makefile ..." to create this tar file. Better still, make this task a target in your Makefile!)
- Send `h2.tar` to the grader (but cc to me). After we untar your file, we expect to type "make" to compile `tsh.c`, and to type "make test" to run the compiled program.
- If you need to resend for any reason, you must again send a single complete tar file. E.g., you forgot to attach a file `foo` the first time, DO NOT simply send us the file `foo`. You must re-package everything and re-send us the new tar file.

## HINTS:

- If you find that the output from your shell is not printed in the sequencing order you expect, try this: after each `printf` command, immediately issue the command `fflush(stdout)`. This is because the standard output is buffered, and `fflush` will force the buffer to be cleared.
- The most useful `exec*` version for this problem is `execvp`. You can use the arrays `args` returned by the `parseCommand` function as follows: `execvp(args[0], args)`.
- For testing purposes, it is helpful to know that you can directly construct your own arguments for `execvp` as follows:

```
main() {  
    char *prog = "ls";  
    char *argv[] = { "ls", "/etc", ".", NULL };  
    ...  
    execvp(prog,argv);  
}
```

◇