System Programming (CSE4009)

Assignment #2 (Due: Dec. 18, midnight)

Tiny Shell - Overview

- A tiny shell program is given (shell.c)
- The current version of the shell program lacks some critical functionalities
  - Connecting two different commands using pipe (|)
  - Changing directory (cd)
  - Executing a binary in the background (&)
  - …
- In Assignment #2, you are asked to add the following three functionalities
  - (1) Pipe (e.g., cmd1 | cmd2)
  - (2) Change directory (e.g., cd dir1/dir2/dir3)
  - (3) Background execution (e.g., ./a.out &)
- Please do not generate other files; put all into “shell.c”

(1) Pipe

- Usage: cmd1 | cmd2
  - The result (output) of cmd1 is given to the input of cmd2
  - Finally, the output of cmd2 is printed
- Examples
  - [ps aux | wc -l] print out # of lines in the output of “ps aux”
  - [ps aux | grep wuc138] print out lines that include “wuc138” in the output of “ps aux”
- Hint (you can implement based on your idea, if you prefer to do so)
  - Create a pipe (pipe())
  - Create two child processes for two different commands (fork())
  - 1st child redirects write_des to stdout & 2nd child redirects read_des to stdin (dup2())
  - Close all des (close()) & run cmd (execvp())

(2) Change Directory

```
ganga01.cse.psu.edu 512% cd class/hw
ganga01.cse.psu.edu 513% pwd
/home/mdl/wuc138/class/hw
ganga01.cse.psu.edu 514% cd ..
ganga01.cse.psu.edu 515% pwd
/home/mdl/wuc138/class
ganga01.cse.psu.edu 516%
```

path

path

```
 chdir, fchdir - change working directory

IS
 #include <unistd.h>

 int chdir(const char *path);
 int fchdir(int fd);

ture Test Macro Requirements for glibc (see feat

 fchdir(): _BSD_SOURCE || _XOPEN_SOURCE >= 500

PTION
 chdir()  changes the current working directory
 directory specified in path.
```

- Usage: cd relative_path_to_destination_directory
  - You should be able to move towards root or leaf
- Hint (you can implement based on your idea, if you prefer to do so)
  - Do not use execvp()
  - Take advantage of “chdir()” system call

(3) Background Execution

```
ganga01.cse.psu.edu 532% ./exe &
[1] 25618
ganga01.cse.psu.edu 533%
ganga01.cse.psu.edu 533%
```

execute a binary

- Usage: ./binary arguments &
  - As soon as you execute a command in the background, the prompt should come up for the next command
- Hint (you can implement based on your idea, if you prefer to do so)
  - Create a child process (fork()) and run a command using it (execv())
  - However, you should not be blocked at "wait()"
  - Consider waitpid(-1, &status, WNOHANG);

```
wait, waitpid, waitid - wait for process to change state

IS
#include <sys/types.h>
#include <sys/wait.h>

pid_t wait(int *status);

pid_t waitpid(pid_t pid, int *status, int options);
```

```
The value of pid can be:

< -1   meaning wait for any child process whose process group ID is equal to
       the absolute value of pid.

-1     meaning wait for any child process.

0      meaning wait for any child process whose process group ID is equal to
       that of the calling process.

> 0    meaning wait for the child whose process ID is equal to the value  of
       pid.

The value of options is an OR of zero or more of the following constants:

WNOHANG     return immediately if no child has exited.
```

Submission and Others

- **Baseline source code (shell.c) is given**
  - **Add your code to given “shell.c”**
- **Submission**
  - **Due: Dec 18 (Sunday) at midnight**
  - **Do not add other files; do not change the file name (shell.c)**
  - **Upload your source file to LMS by the due**
- **Evaluation**
  - **We will execute 9 scenarios for the 3 command types (3 each)**
  - **Do not give up; try to implement something; partial credits will be given generously**
  - **Do not cheat**