System Programming HW3
(Shell Implementation)

Out Nov. 3, 2020
Due Nov. 19, 2020

- **System**: LINUX, CYGWIN, or WSL
- **Language**: C or C++
- **Submission**: An archive file (uploaded to **GEL**)

  What to submit? A report file and source codes.

  - Report: A text file (e.g., "readme.txt")

    Report should contain (Student name, id) +

    (a) How to compile and how to execute? LINUX, CYGWIN, or WSL?

    (b) What functionality was implemented?

    Show execution example with screen capture.

    What was correctly implemented? and what was not?

    (c) How was it implemented? (explain briefly)

    (d) Conclusion

  - Source codes

    - Create at least 4 source files (.c or .cpp) and add "Makefile"

  - Use zip command to compress all related files (codes, makefile, & the report) into one .zip file.
  - Upload the .zip file to **GEL**.

- **Execution**

  - After compilation with make, "minish" should be created.
  - Execute "minish" in a LINUX, CYGWIN, or WSL shell prompt.
  - "minish" should print a shell prompt line and wait for a user input.
  - When user input is received, "minish" interpret and process the input.
  - While "minish" is running, it should not be killed abnormally.

- **Functionality to be implemented**

(1) [50 points] Basic execution of commands.

  - "minish" should receive a user input in a command line and execute the command.

    After the execution, "minish" should wait for another input.

  - To implement this, you should use fork & exec system call.

    You must not use the function "system"

(2) [10 points] Signals

- “minish” should not be killed unless a user commands to quit the “minish”.

  Ctrl-C should kill currently running command. However, Ctrl-C should not kill “minish”.

  To exit “minish”, the built-in command “`quit`” should be implemented.

(3) [10 points] Prompt

- “minish” should print current directory path in a prompt.

  you may use the function getcwd

  `(example) /home/unix/sp$ _`

(4) [15 points] redirection

- you may use the function dup2

`(example) /home/unix/sp$ ls -l > a.txt`

After execution, the result of "ls -l" is written to the file a.txt

(5) [15 points] pipe

- you may use the function dup2

```
(example) /home/unix/sp$ cat code.c | wc
```

After execution, the standard output of "cat code.c" is put into the standard input of "wc"

● **Advanced Functionality to be implemented**

(6) [25 Bonus points] background process

- When a background job is done, “minish” should display that.

```
(example) /home/unix/sp$ sleep 2 &
          /home/unix/sp$
          [1]+ Done          sleep 2
          /home/unix/sp$ _
```

- You may use the function waitpid(pid, status, options)
- The built-in command "`jobs`" should also be implemented just like UNIX "jobs"

  If a job is finished, "Done" should be displayed once.

  If a job is running, "Running" should be displayed.

```
(example) /home/unix/sp$ sleep 5 &
          /home/unix/sp$ sleep 3 &
          /home/unix/sp$ sleep 7 &
          /home/unix/sp$ jobs
          [1] Done                 sleep 5
          [2] Done                 sleep 3
          [3] Running              sleep 7
```

- The build-in command "`kjob`" should be implemented to kill a job. Format: `kjob <job_num>`

  Use a function "`kill(pid, SIGKILL)`".

```
(example) /home/unix/sp$ jobs
          [1] Done                sleep 5
          [2] Done                sleep 3
          [3] Running             sleep 7
          /home/unix/sp$ kjob 3
          [3] Terminated          sleep 7
```

- For more, visit https://www.gnu.org/software/libc/manual/html_node/Implementing-a-Shell.html

(7) [10 bonus points] Implementation of other built-in commands

- cd : changes directories. By itself, it changes to the directory specified in the HOME environment variable. It can take one argument, the path of the directory to change to.

```
(example) /home/unix$ cd sp
          /home/unix/sp$ _
```

- path : prints or changes the path variable. By itself, it prints out the value of the PATH environment variable. It can also take one argument, the new value of the PATH environment variable.
- status : prints out the exit status of the previous command.

(8) [15 bonus points] Wildcard processing (*, ?)

- *: zero or more arbitrary characters
- ?: one arbitrary character

```
(example) /home/unix/sp$ ls *.c
          a.c b.c code.c
```

* you may assume that each component of a command is separated by one or more spaces.

```
(example) prog > outfile (O)
          prog>outfile (X)
```