2013.2 LINUX System Programming Assignment

MiniShell Implementation

(Due : Oct. 16th, 11:59pm)

- **System** : LINUX or CYGWIN
- **Language** : C or C++
- **Submission** : File Upload (source codes+report file, eClass)
- **What to submit ?**
  - Report : HWP or PDF file upload in eClass

    Report should contain (Student name , id) +
    - (i) How to compile and how to execute? LINUX or CYGWIN?
    - (iii) "무슨 기능이 구현되었는지?" 와 "구현된 기능이 잘 작동하는지?" 기술
      what was correctly implemented? and what was not?
    - (iv) How was it implemented? (explain briefly)
    - (v) 각 기능에 대한 실행 예 (with screen capture)
    - (vi) Conclusion
  - Source code
    - Create at least three .c or .cpp source files, and add "`Makefile`"
    - Make a directory whose name should be exactly your student_id# and put all source files and Makefile into the directory.
    - use `tar` and `gzip` command to compress the above directory and generate .tar.gz file

      (example) tar cvf 20113051.tar dir_name

      gzip 20113051.tar (.gz file은 gzip -d 로 압축을 풀 수 있음)
    - Submit the `.tar.gz` file to eClass page
- **Execution**
  - After compilation with make, MiniShell.exe should be created.
  - Execute MiniShell in a LINUX or CYGWIN shell prompt.
  - MiniShell1 should print a shell prompt line and wait for a user input.
  - When user input is received, MiniShell interpret and process the input.
  - While MiniShell is running, it should not be killed abnormally.

- **Functionality to be implemented**

(1) [30points] Basic execution of commands.

- MiniShell should receive a user input in a command line and execute the command. After the execution, MiniShell should wait for another input.
- To implement this, you should use `fork & exec` system call.

  You must not use the function **"system"**
- support multiple commands using ";"
- For wrong input command, your shell should display appropriate error messages.

(example) `/home/bongbong/os> ls -l ; whoami ; cat a.txt`

(2) [5points] Prompt : MiniShell should print current directory path in a prompt

- you may use the function `getcwd`

(example) `/home/bongbong/os> _`

(3) [10points] Implementation of other built-in commands

- cd : change current directory , ~ : home directory processing

(example)
```
/home/bongbong> cd os
/home/bongbong/os> _
/home/bongbong/os> cd ~bongbong
/home/bongbong> _
```
- exit : quit MiniShell

- alias : (example) /home/bongbong> alias dir ls -alF
- echo : print

(4) [20points] Wildcard processing (*,?)

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

(example) /home/bongbong> ls *.c
a.c b.c code.c

(5) [10points] Shell variables and Environment Variables

- handling environment variables

(example) /home/bongbong> echo $USER
bongbong
/home/bongbong> setenv USER bssohn
/home/bongbong> echo $USER
bssohn

- handling shell variables

(example) /home/bongbong> set x = hello
(example) /home/bongbong> echo $x
hello

(6) [10points] Double Quotes, Single Quotes, and Backslash

- " " : double quotes remove the special interpretation of most metacharacters (e.g. <,>,|,*,?,...)
The exceptions are the dollar sign ($) in front of a variable name.

(example) /home/bongbong> find . -name "*.c" -print

- ' ' : single quotes operate like double quotes, but their effect is stronger.
Any enclosed metacharaters are treated as iteral characters.

(example) /home/bongbong> set x = hello
/home/bongbong> echo "< > $x *.c &"
< > hello *.c &
/home/bongbong> echo '< > $x "y" ? &'
< > $x "y" ? &

- ₩ (backslash) : converts special character into literal character

(example) /home/bongbong> find . -name ₩*.c -print

※meta-character : UNIX 쉘에서 특수한 의미를 갖는 문자 (예: $, *,?,[],~,!,;,&,|,<,>,&,% 등)

(7) [15points] Report Quality

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

(example) ls -l;cat a.txt (X) ls -l ; cat a.txt