Homework #3

[ECE30021/ITP30002] Operating Systems

Mission

- Solve problem 1, 2, and 3
  - Solve the problems on Ubuntu Linux (VirtualBox or UTM) using vim and gcc.
  - Problem 1 is a group assignment but Problem 2 and 3 are not.
- Submission
  - Submit a .tgz file containing hw3_1.c, hw3_2.c, and hw3_3.c on LMS.
    "tar cvfz hw3_<*student_id*>.tgz hw3_1.c hw3_2.c hw3_3.c"
    (Do not copy&past but type the above command.)
  - After compression, please check the .tgz file by decompressing in an empty directory.
    "tar xvfz hw3_<*student_id*>.tgz"
- Due date: PM 11:00:00, Apr. 3rd

Honor Code Guidelines

- “과제”
  - 과제는 교과과정의 내용을 소화하여 실질적인 활용 능력을 갖추기 위한 교육활동이다. 학생은 모든 과제를 정직하고 성실하게 수행함으로써 과제에 의도된 지식과 기술을 얻기 위해 최선을 다해야 한다.
  - 제출된 과제물은 성적 평가에 반영되므로 공식적으로 허용되지 않은 자료나 도움을 획득, 활용, 요구, 제공하는 것을 포함하여 평가의 공정성에 영향을 미치는 모든 형태의 부정행위는 단호히 거부해야 한다.
  - 수업 내용, 공지된 지식 및 정보, 또는 과제의 요구를 이해하기 위하여 동료의 도움을 받는 것은 부정행위에 포함되지 않는다. 그러나, 과제를 해결하기 위한 모든 과정은 반드시 스스로의 힘으로 수행해야 한다.
  - 담당교수가 명시적으로 허락한 경우를 제외하고 다른 사람이 작성하였거나 인터넷 등에서 획득한 과제물, 또는 프로그램 코드의 일부, 또는 전체를 이용하는 것은 부정행위에 해당한다.
  - 자신의 과제물을 타인에게 보여주거나 빌려주는 것은 공정한 평가를 방해하고, 해당 학생의 학업 성취를 저해하는 부정행위에 해당한다.
  - 팀 과제가 아닌 경우 두 명 이상이 함께 과제를 수행하여 이를 개별적으로 제출하는 것은 부정행위에 해당한다.
  - 스스로 많은 노력을 한 후에도 버그나 문제점을 파악하지 못하여 동료의 도움을 받는 경우도 단순한 문법적 오류에 그쳐야 한다. 과제가 요구하는 design, logic, algorithm의 작성에 있어서 담당교수, TA, tutor 이외에 다른 사람의 도움을 받는 것은 부정행위에 해당한다.
  - 서로 다른 학생이 제출한 제출물간 유사도가 통상적으로 발생할 수 있는 정도를 크게 넘어서는 경우, 또는 자신이 제출한 과제물에 대하여 구체적인 설명을 하지 못하는 경우에는 부정행위로 의심받거나 판정될 수 있다.

Problem 1

- Mission: peer review of hw#2 as a group
  - Sign up study group
    - Section 01: https://docs.google.com/spreadsheets/d/1fIRP_15xVKxahA6bGDeifoBvW2upYjgkyi6eU0ch4MQ/edit?usp=sharing
    - Section 02: https://docs.google.com/spreadsheets/d/1JovWOmVhmVbF2A74NX_T7uXxEZrC-oRXlgHeK9r53tQ/edit?usp=sharing
  - Have a meeting to review hw2_2 and hw2_3
    - Review the solutions of other members and share comments to each other.
    - Each member update her/his solution to hw2_3 individually applying the comments.
      - Mark the modified parts by comments as the next page.
      - If you believe your previous solution was perfect, submit it again and put a comment “My previous solution was perfect, so I didn't modify it!” in the first line.
    - Review both hw2_2 and hw2_3 but submit only the updated solution to hw2_3.

Problem 1

- Example of updated solution.

```
int main()
{
	int i = 0;

	// previous code
	/*
	<old code>

	*/

	// updated code
	<new code>


	return 0;
}
```

Problem 2

- Write a shell (command line interpreter) that runs UNIX commands in a working directory following the instructions
  - at startup, move into the working directory (the global variable working_dir).
    - if the working directory does not exist, create it by calling the CheckAndCreateDirectory() function
  - repeat until the user inputs "quit" command
    - read a command line
    - split the command line into arguments by calling the SplitCommandLine() function
    - for debug, display the split arguments
    - if the command line is not empty,
    - execute the command line using the fork(), execvp(), and wait() system calls
      // all commands should run in the working directory
- TO DO: Compete hw3_2_problem.c following the instructions
  - Complete main()
  - Write CheckAndCreateDirectory() by reusing the solution to the previous homework with slight modification
  - Write SplitCommandLine()

Problem 2

- Example

```
Welcome to myshell.          // greeting messages
All commands will be executed in "Working/".
Type "quit" to quit.
Directory Working was created.          // this message was printed
                                        // because the working directory did not exist
$ ls                // the first command from the user
argc = 1            // split arguments
argv[0] = ls
argv[1] = (null)
// the first 'ls' command does not print anything bcause the working directory is initially empty
// if the working directory already exists and contains files, the 'ls' command should display its contents
$ ls -al            // the second command from the user
argc = 2            // split arguments
argv[0] = ls
argv[1] = -al
argv[2] = (null)
total 0
// the result of 'ls -al' showing only the current and parent directories
drwxrwxrwx 1 callee callee 512 Mar 25 09:49 .
drwxrwxrwx 1 callee callee 512 Mar 25 09:49 ..
// continued on the next page
```

Problem 2

- Example (cont'd)

```
$ vi test.txt        // the third command from the user
argc = 2
argv[0] = vi
argv[1] = test.txt
argv[2] = (null)
// the shell executes vi to allow the user to edit 'test.txt'
$ ls                 // the user typed 'ls' again
argc = 1
argv[0] = ls
argv[1] = (null)
test.txt
                     // the result of 'ls'. now, it contains a file
$ quit               // command to terminate the shell
Bye!                 // a goodbye message
```

Problem 3

- Extend your solution to Problem 2 to prevent the user from moving out of the working directory
  - Upgrade SplitCommandLine() to SplitCommandLine2().
    - If the command contains an argument containing a directory, remove the directory name from the argument.
    - If an argument contains "..", replace it by "."

- Example) cp test.c ../test2.c

  argc = 3

  argv[0] = cp

  argv[1] = test.c

  argv[2] = test2.c // the directory name "../" was discarded

  argv[3] = (null)

Problem 3

- Example)

```
Welcome to myshell.
All commands will be executed in "Working/".
Type "quit" to quit.
$ ls ..                    // the user tries to see the parent directory
argc = 2
argv[0] = ls
argv[1] = .                // ".." was replaced by "."
argv[2] = (null)
test.txt // display only the current directory
$ cp test.txt ..           // the user tries to copy 'test.txt' to the parent directory
argc = 3
argv[0] = cp
argv[1] = test.txt
argv[2] = .                // ".." was replaced by "."
argv[3] = (null)
cp: 'test.txt' and './test.txt' are the same file          // the shell executed 'cp test.txt .'
$ vi ../test2.txt          // the user tries to create a file 'test2.txt' in the parent directory
argc = 2
argv[0] = vi
argv[1] = test2.txt        // "../test2.txt" was replaced by "test2.txt"
argv[2] = (null)
// the shell execute 'vi test2.txt'. the new file is created in the working directory
$ ls
argc = 1
argv[0] = ls
argv[1] = (null)
test.txt  test2.txt        // the current directory contains 'test2.txt'
$ quit
Bye!
```