[2016　網路系統程式設計　Homework 1]

◆ 規則:

1. 請在 **Ubuntu 16.04** 系統上使用 C 語言寫本次作業並進行測試。
2. 本次作業請使用附上的 Makefile 來編譯你的程式。
3. 嚴禁抄襲其他同學作業，參與者(抄襲與被抄襲)均以**零分**計算。
4. 請對你的程式碼有深入瞭解，Demo 時助教會問。
5. **逾期以零分計算，不接受補交**，有問題請事先告知，Demo 時間會另外通知。
6. 在寫作業的時候務必注意能在 **Ubuntu 16.04** 上執行你的程式，不同的 linux作業系統之下 可能 lib 或某些指令會有些許不同。
7. 對題目有問題可以寄信問助教群(sp_ta@net.nsysu.edu.tw)或是到實驗室(F5018)詢問，但不幫忙 debug。

◆ 作業上傳:

1. 請壓縮成 zip 或 tar 檔，作業命名規則為 “學號_SP_HW1.zip”，**請勿壓縮成 rar 檔或其它壓縮檔格式，否則將會扣分。**

Example: “M0530400XX_SP_HW1.zip”

2. 上傳到網路大學。
3. DeadLine： **2016 年 9 月 27 日(週二)23:59**

◆ 主旨：

本次作業著重於讓學生練習製作簡單的 shell parse

◆ 第一部分：

1. Edit the **parse.c** file to use **strtok()** and **realloc( )** to implement the **parse()** and **free_argv()** functions.

2.Files provided:

shell.h

shell.c

parse.c

3. 執行結果如下：

```
myshell -> system program
[0] : system
[1] : program
myshell -> homework one is vary easy
[0] : homework
[1] : one
[2] : is
[3] : vary
[4] : easy
```

◆ 第二部分：

1. Add code to the **builtin.c** stub to recognize the **echo, quit, exit and bye** commands. Write functions implementing these commands, and add a new line for each command to table inbuilts[] just above the line {NULL, NULL}.

2. File provided:

   Builtin.c

3. echo 執行結果如下：

   如果沒有參數-n，則印出所有字串，若有-n，則印出指定位置的字串

```
myshell -> echo -n 1 one two three
[0] : echo
[1] : -n
[2] : 1
[3] : one
[4] : two
[5] : three
one
myshell -> echo -n 2 one two three
[0] : echo
[1] : -n
[2] : 2
[3] : one
[4] : two
[5] : three
two
```

```
myshell -> echo -n 3 one two three
[0] : echo
[1] : -n
[2] : 3
[3] : one
[4] : two
[5] : three
three
myshell -> echo one two three
[0] : echo
[1] : one
[2] : two
[3] : three
one two three
```

4. quit 執行結果如下：

   exit, quit, logout and bye 這四個 command 作用皆同為結束程式

```
vacha@sun2:~/hw1$ ./myshell
myshell -> exit
[0] : exit
vacha@sun2:~/hw1$
```

◆ 第三部分：

1. Edit the **run_command.c** file so a child process is created to run the command, and the parent waits for the child process to terminate. Check for builtin commands first, create a new process only for commands which are not built in.

Use the parser from pervious labs to create from the command line the argv array passed to the child.

Hint: You can use **exec()/execvp()** function, **waitpid()** function and the **fork()** system call.

2.File provided:

run_command.c

3. 執行結果如下：

```
myshell -> ls -l
[0] : ls
[1] : -l
total 772
-rw-rw-r-- 1 vacha vacha   2450 Jul 31 15:13 builtin.c
-rw-rw-r-- 1 vacha vacha   1956 Jul 31 16:40 builtin.o
-rw-rw-r-- 1 vacha vacha    355 Jul 31 10:48 is_background.c
-rw-rw-r-- 1 vacha vacha    306 Jul 31 16:40 makefile
-rwxrwxr-x 1 vacha vacha  12387 Jul 31 16:40 myshell
-rw-rw-r-- 1 vacha vacha   2012 Jul 31 15:44 parse.c
-rw-rw-r-- 1 vacha vacha   1908 Jul 31 16:40 parse.o
-rw-rw-r-- 1 vacha vacha    971 Jul 31 16:38 run_command.c
-rw-rw-r-- 1 vacha vacha   1308 Jul 31 16:40 run_command.o
-rw-rw-r-- 1 vacha vacha    779 Jul 31 16:31 shell.c
-rw-rw-r-- 1 vacha vacha    430 Jul 31 10:58 shell.h
-rw-rw-r-- 1 vacha vacha 725248 Jul 31 16:40 shell.h.gch
-rw-rw-r-- 1 vacha vacha   1552 Jul 31 16:40 shell.o
myshell ->
```

◆ 第四部份：

1. Edit the **is_background.c** file to detect an **"&"**. Alter the **run_command.c** file so that if a task is running in the background, the parent does not wait. Do not worry about background processes becoming zombies at this point; this will be addressed later.

2.File provided:

is_background.c

3. 執行結果如下：