[2014　網路系統程式設計　Homework 4]

◆　主旨：

1. 讓學生了解 parent process 與 child process 間的 signal 繼承關係。
2. 讓學生了解到 process 在 user/kernel space 所花費的時間。

◆　第一部分：

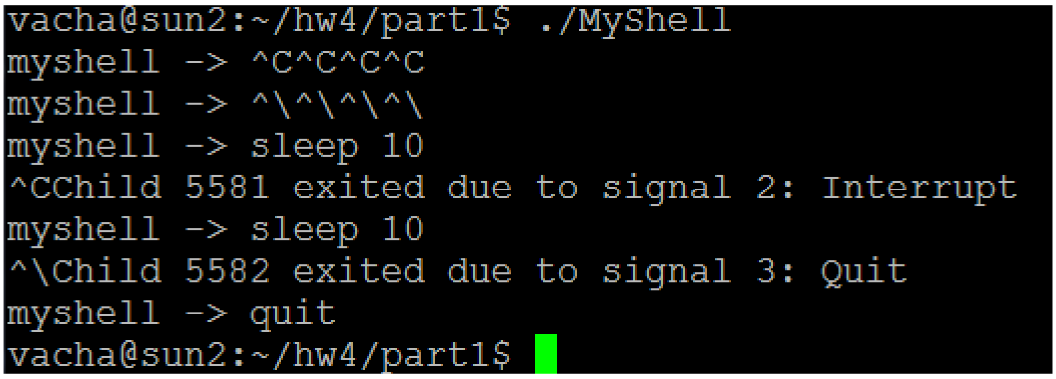
1. 請接續 Homework2 的 part2 繼續實作。

   When fork() create a child process, it clones the signal handling from the parent. If the parent is set up to ignore signals before forking, the child will inherit this behavior as well.

   Signals ignored by a parent are propagated to the child automatically. The child can then catch or ignore the signal as appropriate.

   Edit the shell.c file so that the parent ignores the interrupt (Ctrl+c) and quit (Ctrl+\) signals. Edit the run_command.c file so that the child reverts to default signal handling.

2. 執行結果如下：



```
vacha@sun2:~/hw4/part1$ ./MyShell
myshell -> ^C^C^C^C
myshell -> ^\^\^\^\
myshell -> sleep 10
^CChild 5581 exited due to signal 2: Interrupt
myshell -> sleep 10
^\Child 5582 exited due to signal 3: Quit
myshell -> quit
vacha@sun2:~/hw4/part1$
```

◆　第二部分：(此部分須包含第一部分的功能)

1. 請先了解 struct tms 此結構內各值的實際意義。

   Add two functions, set_timer() and stop_timer(), to the timer.c file. Insert calls to these functions in the run_command.c file.

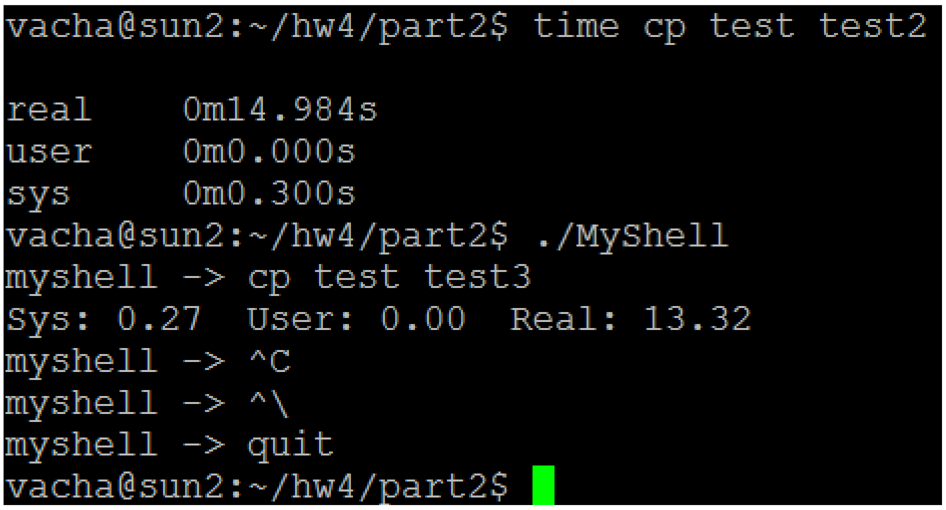
   Hint: set timer before forking; parent stops the timer after wait.

2. Files provided：

I.　timer.c

3. 執行結果如下：

以計算 cp 所花費的時間作為範例，結果近似即可。



```
vacha@sun2:~/hw4/part2$ time cp test test2

real    0m14.984s
user    0m0.000s
sys     0m0.300s
vacha@sun2:~/hw4/part2$ ./MyShell
myshell -> cp test test3
Sys: 0.27  User: 0.00  Real: 13.32
myshell -> ^C
myshell -> ^\
myshell -> quit
vacha@sun2:~/hw4/part2$
```

◆ 第三部分：

1. Develop a client-server model that communicate through named pipes and look up words in the dictionary.

   The following steps provide some information about the client and server you are going to implement.

   I. The server is told the name of a FIFO (call it the request FIFO) on which it will listen for requests. It opens the request FIFO and waits for a request. When a request comes in, it does the dictionary look up and then opens the reply FIFO (the name of which is part of the request) and sends the response back through it. The request is of type Client, defined in dict.h.

   II. The client creates a FIFO to receive a response (call it the reply FIFO), then prepares and sends a packet to the server with a word to look up and the name of reply FIFO it just created. It then waits for the reply.

   III. The server is called fifo_server and is built from lookup2.c (a module which performs the dictionary look ups) and fifo_sever.c (which interfaces with the client through the FIFOs).

   IV. The client is called fifo_client and is built from main.c (the user interface module) and lookup3.c (which interfaces with the server through the FIFOs).

2. The following provides you with some help on how to use the files provided to develop the client and server for this task.

   I. The request FIFO used to send the server a request is created before running the client or server , lookup3. c must create a new reply FIFO for each request because the FIFO's name is part of the request. Edit the lookup3.c file to create the reply FIFO.

II. Edit lookup3. c to send a request down the request FIFO and wait for a response.

III. Edit fifo_server.c to read the request FIFO, invoke the routine in-lookup2.c to do the search, and send the reply back to the client.

IV. After the files have been edited, type make, or make fifo_server and make fifo_client.

V. When y ou get the prompt, make the request FIFO, and run the fifo_server and fifo_client as shown in the sample output.

3. Files provided：

I. dict.h

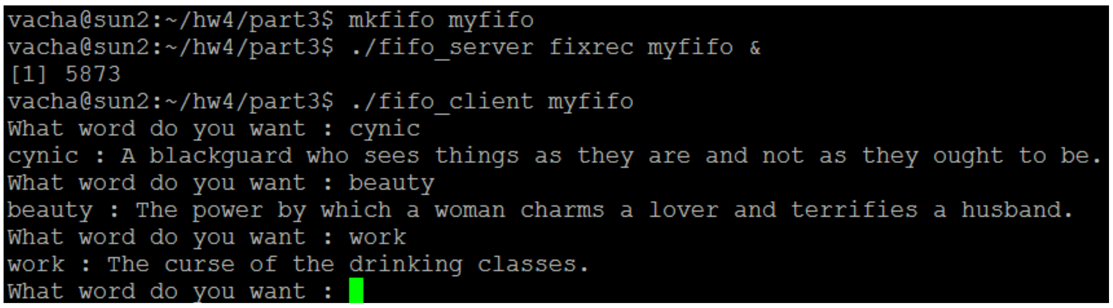
II. main.c

III. lookup3.c

IV. fifo_server.c

V. fixrec

4. 執行結果如下：



```
vacha@sun2:~/hw4/part3$ mkfifo myfifo
vacha@sun2:~/hw4/part3$ ./fifo_server fixrec myfifo &
[1] 5873
vacha@sun2:~/hw4/part3$ ./fifo_client myfifo
What word do you want : cynic
cynic : A blackguard who sees things as they are and not as they ought to be.
What word do you want : beauty
beauty : The power by which a woman charms a lover and terrifies a husband.
What word do you want : work
work : The curse of the drinking classes.
What word do you want : 
```

◆ 第四部分：

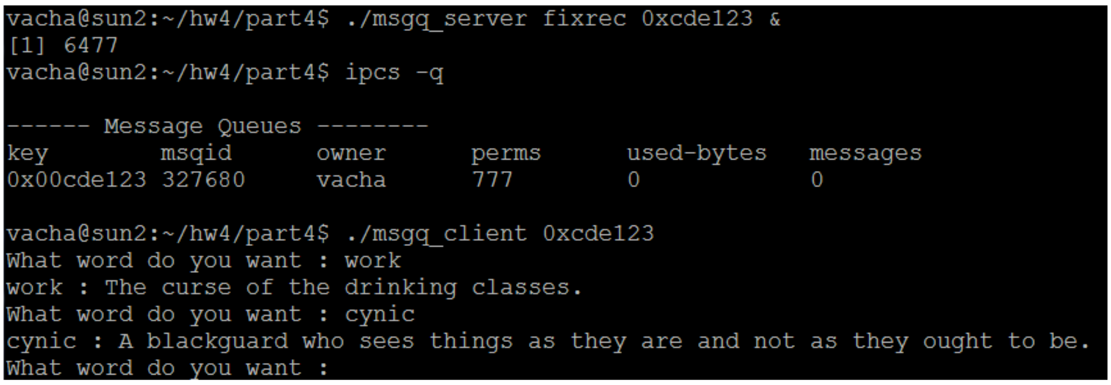
1. Develop a client and server that communicate through a message queue and look up words in the dictionary.

The following steps provide some information about the client and Server you are going to develop.

I. The server opens up a message queue on which it will listen for requests. Type 1 messages are designated to be requests; the server listens only for these. When a request comes in, the server does the dictionary look up and sends the reply. Reply messages are keyed with a type equal in value to the PID of the sender.

II. The client opens up the message queue, then prepares and sends a message containing the word to look up and the PID of the sender, to the server. It then waits for the reply.

    III. Messages from the client are of type ClientMessage. Messages from the server are of type ServerMessaqe.
    IV. The server is called msgq server and is built from lookup2.c (the module which performs the dictionary took ups) and msgq_server.c (which interfaces with the client through the message queue).
    V. The client is called msgq_client and is built from main.c (the user interface module) and lookup4.c (which interfaces with the server through the message queue).
2. The following provides you with some help on how to use the files provided to develop the client and server for this task.
    I. Edit msgq_server.c to get type 1 messages, and send messages of type N, where N is the PID of the client. Use the look up routine in lookup2.c to do the searching.
    II. Edit lookup4.c to send messages down the queue. Link with main.c.
    III. After the files have been edited, type make, or make msgq_server and make msgq_client.
    IV. When you get the prompt, run the msgq server and msgq_client as shown in the sample output.
3. File provided:
    I. dict.h
    II. main.c
    III. lookup4.c
    IV. msgq_server.c
    V. fixrec
4. 執行結果如下:



```
vacha@sun2:~/hw4/part4$ ./msgq_server fixrec 0xcde123 &
[1] 6477
vacha@sun2:~/hw4/part4$ ipcs -q

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x00cde123 327680     vacha      777        0            0

vacha@sun2:~/hw4/part4$ ./msgq_client 0xcde123
What word do you want : work
work : The curse of the drinking classes.
What word do you want : cynic
cynic : A blackguard who sees things as they are and not as they ought to be.
What word do you want :
```

◆ 限制：

1. 請在 Ubuntu 14.04 系統上使用 C 語言寫本次作業並進行測試。
2. 本作業必須上傳能編譯本作業之 Makefile，內容不拘，未寫要扣分。
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. 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。
6. 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。

◆ 作業上傳：

1. 請壓縮成 zip 或 tar 的壓縮檔，並上傳至中山網路大學，作業命名規則為“學號_SP_HW4”。

   Example: M013040001_SP_HW4.zip
2. 作業截止時間為 2014/10/21 (Tue.) 23:59，請在時間內上傳作業。