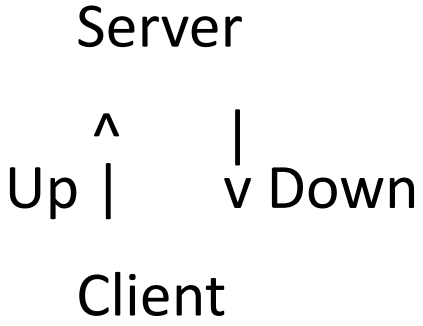
Lab 8 : A Hardcore IPC

Prof. Zichen Xu

Get Ready

- The purpose of this lab is to allow students to become comfortable with System V IPC using semaphores, message queues and shared memory.

- Two processes, client and server, communicate via two message queues "Up" and "Down".



Task 1 Client-Server Communication using Message Queues

- The client reads a message from the standard input and sends it to the server via the Up queue, then waits for the server's answer on the Down queue.
- The server is specialized in converting characters from lower case to upper case and vice versa. Therefore, if the client sends the message "lower case" via the Up message queue, the server will read the message, convert it, and send "LOWER CASE" via the Down queue.
- When the client receives the message from the server, it prints it out. You may assume the maximum size of any message is 256 bytes.

Continued

- Make sure you understand the problem. Send questions to the list. An example of how you could run two processes is below:

```
good@good:./LAB8.server &
good@good:./LAB8.client
Insert message to send to server: message
Msg processed: MESSAGE

Insert message to send to server: UPPER CASE
Msg processed: upper case
```

Task 2 The Consumer-Producer Problem using Shared Memory

- Write 2 programs, producer.c implementing a producer and consumer.c implementing a consumer, that do the following:
  - Your product will sit on a shelf: this will be a shared memory segment that contains an integer, a count of the items "on the shelf". This integer may never drop below 0 or rise above 5.
  - Your producer creates the shared memory segment and sets the value of the count to 5. It is the producer program's responsibility to stock product on the shelf, but not overstocked. The producer may add one item to the shelf at a time, and must report to STDOUT every time another item is added as well as the current shelf count.
  - Your consumer will remove one item from the shelf at a time, provided the item count has not dropped below zero. The consumer will decrement the counter and report the new value to STDOUT. Have your consumer report each trip to the shelf, in which there are no items.

Examples

- Check the attached source code for help
- Look on these examples of shared memory usage: shm_server.c creates a shared memory segment and writes "hello world" into it. shm_client.c reads and prints out the content of the shared memory segment.

```
good@good gcc -o shm_server shm_server.c
good@ good gcc -o shm_client shm_client.c

good@ good  shm_server 1234 &
[1] 27633
Try to create this segment
shared memory content: hello world

good@ good shm_client 1234
Trying shared memory 1234
shared memory: 1152
shared memory: 0x40016000
shared memory content: hello world
```