CCPS590 Lab 7 – Message Queues in Linux

Preamble

In this lab you will experiment with message queues in Linux.

Lab Description

1) We will use System V message queues in this lab. Functions needed are:

`msgsnd()` - send a message
`msgrcv()` - receive a message
`msgget()` - connect to (make) a message queue

To get an idea of how to use them, refer to MessageQueues.html included with this lab.

2) Copy programs kirk.c and spock.c into your file space, and compile them into executables **kirk** and **spock**. Open 2 shell windows.

In the first window:
- Run **kirk**
- Enter several lines, but DO NOT do CTRL-D yet! Then...

In the second window:
- Run **spock**

Notice **spock** immediately reads all the pending messages from **kirk**. Now make kirk give some more orders before he ends (CTRL-D).

3) When **kirk** ended, why did **spock** also end?

4) Try starting **spock** first. What "output" does it give? Why?

5) Start **kirk** in one window, then **spock** in another. Make **kirk** give some orders. Now try to end **spock** with CTRL-D. It won't end. Why? End both by sending **kirk** CTRL-D.

6) Copy kirk.c and spock.c to kirkKey.c and spockKey.c. Modify kirkKey.c and spockKey.c so that they each use an integer key given as a command-line argument. I.e., they do not use **`ftok()`** to get a key; instead the key is given in the string **`argv[1]`**. (Your program terminates immediately if no command line argument is given, or if the command line argument is not an integer.)

7) Any message queues you make will continue to exist until they are explicitly deleted. How does kirk.c and/or spock.c delete the message queue?

You can ask the system what message queues you have open by running command:

```
ipcs
```

You can manually delete a message queue using the command…

```
ipcrm -q msqid
```

…where **`msqid`** is the message queue ID number shown in the output of the **`ipcs`** command. If you ever CTRL-C one of your **kirk** processes, you probably have old message queues hanging around, and should delete them.

8) Make several shell windows. Try running a kirkKey in one window, and in each of three or so different windows, run a spockKey. Enter several kirk commands and observe which spocks end up receiving the messages. What can this tell you about the process scheduling?

Submission

For this lab submit your files kirkKey.c and spockKey.c. At the top of kirkKey.c, include a block comment containing written answers to questions 3, 4, 5, 7, and 8. Your code must compile and run out of the box even with the written answers included.

Labs are to be submitted ***individually***! Make sure your code and written answers are formatted cleanly and are easy to read.