# 2014 System Programming Assignment 3

Announce: Mon, Dec 15, 2014
Due: 12:00 Mon, Dec 29, 2014

## I. Problem Description

In this assignment, you need to implement a document transferring system which contains sender and receiver. Sender sends several documents to receiver and receiver deals with the documents. After receiver finishes the document, it should send the document back to sender. We use some signals to indicate the documents of this assignment.

There are three type of documents: ordinary document, urgent document and very urgent document. There are different priorities of the three documents. The priority order of document is: very urgent > urgent > ordinary. That is, a document with higher priority should be dealt with first when the receiver receives it. If the priority of the processing document is greater than or equal to the priority of the arriving document, then the arriving document should be blocked until the processing document finished.

EX1: if an ordinary document d2 arrives at the time the receiver deals with an ordinary document d1, the receiver should finish d1 first.

EX2: if a very urgent document d2 arrives at the time the receiver deals with an urgent document d1, the receiver should stop to deal with a very urgent document d2.

EX3: if an urgent document or an ordinary document d2 arrives at the time the receiver deals with a very urgent document d1, the receiver should finish a very urgent document d1 first.

Three type of document also have different time to process:
ordinary document: 1.0 second
urgent document: 0.5 second
very urgent document: 0.2 second

*Note: you can use for loop and usleep() to simulate the time consuming, but you should notice that usleep() may be interrupted by signal.

The sender should receive the finished document from receiver within a specific time.
ordinary document: no limit
urgent document: 1.0 second
very urgent document: 0.3 second

EX: If sender sent a very urgent document at 8:01:53.1, the document should be sent back before 8:01:53.4

## II. Structure of program and Format of data

Two programs are required: “sender.c” and “receiver.c”

### ● receiver.c (**$./receiver [test_data]**)

In the program, you should open a file named with “receiver_log” which will be specified later. It is used to write some information during the execution of the program.

And the receiver should fork() one child and run exec() to execute the sender program. Receiver should use pipe() to receive the ordinary document, and redirect sender’s stdin/out to pipe.

The **[test_data]** is a file name. And the content of the file is some information of testing data. It is also the argument of the sender program.

When receiver receives and starts dealing with the document, it needs to write “receive [priority] [serial_number]” to file “receiver_log”. When receiver finishes it, it needs to write “finish [priority] [serial_number]” to the file.

When receiver finishes the document, it should send a specific signal to sender and continue to deal with the next document. (There may be some documents blocked. If no document blocked, the receiver needs to wait until next document arriving.)

From receiver to sender:
The “SIGINT” signal indicates an ordinary document is finished.
The “SIGUSR1” signal indicates an urgent document is finished.
The “SIGUSR2” signal indicates a very urgent document is finished.

When receiver read “EOF” from pipe, it should write “terminate” to file “receiver_log” and terminate the program itself.

When receiver receives the signal “SIGINT” from keyboard, it should kill sender and write “terminate” to file “receiver_log” and terminate the program itself.

### ● sender.c (**$./sender [test_data]**)

In the program, you should open a file named with “sender_log” which will specify later. It is used to write some information during the execution of the program.

You should read **[test_data]** and send the document at the specific time in the **[test_data]**.

When sender sends the document, it needs to write “send [priority] [serial_number]” to file “sender_log”. When sender gets the finished document (a specific signal) sent back by receiver, it needs to write “finish [priority] [serial_number]” to file.

Note that a document should receive within a specific time. If receiver don't send back the document within the specific time, the sender should write "timeout [priority] [serial_number]" to file "sender_log" and terminate the program itself.

From sender to receiver:
Send "ordinary\n" through pipe to indicate an ordinary document is sent.
The "SIGUSR1" signal indicates an urgent document is sent.
The "SIGUSR2" signal indicates a very urgent document is sent.

*Note1: In our test data, we assure that if sender sent an urgent document, then it would not send another urgent document before receiving the previous urgent document. (for avoiding signal missing). And so is the case of very urgent documents.

## ● the format (content) of **[test_data]**

The **[test_data]** file contains several lines. Each line has two numbers, the first number is the priority of document (0: ordinary, 1: urgent, 2: very urgent) and the second number is the sending time of the document. The unit of time is 0.1 second.  The time is in ascending order in the testing data.

## ● the format of "receiver_log"

Each line should write in the format:

**[action] [priority] [serial_number]**

The **[action]** will be "receive" or "finish". When receiver receives a document, the **[action]** should be "receive". When receivers finishes a document, the **[action]** should be "finish".

The **[priority]** will be 0, 1, or 2 which indicates the ordinary, urgent or very urgent document respectively.

The **[serial_number]** is the serial number of document. Each type of document has its own serial number.

## ● the format of "sender_log"

Each line should be in the format:

**[action] [priority] [serial_number]**

The **[action]** will be "send" or "finish" or "timeout". When sender sends a document, the **[action]** should be "send". When sender receives a finished document from receiver, the **[action]** should be "finish". When a document isn't sent back in specific time, the **[action]** should be "timeout".

The **[priority]** will be 0, 1, or 2 which indicate the ordinary, urgent and very urgent document respectively.

The **[serial_number]** is the serial number of document. Each type of document has its own serial number.

## III. Sample execution

**test_data:**

```
0 0
1 8
2 12
2 21
1 22
```

**receiver_log:**

```
receive 0 1
receive 1 1
receive 2 1
finish 2 1
finish 1 1
finish 0 1
receive 2 2
finish 2 2
receive 1 2
finish 1 2
terminate
```

**sender_log:**

```
send 0 1
send 1 1
send 2 1
finish 2 1
finish 1 1
finish 0 1
send 2 2
send 1 2
finish 2 2
finish 1 2
```

## IV. Tasks and scoring

There are 6 subtasks in this assignment. By finishing all subtasks you earn the full 8 points.

(1) Generate the two execution files. (1 point)
(2) Simple test data (only ordinary documents) (1 point)
(3) Complicated test data (ordinary documents and urgent documents) (1 point)
(4) More complicated test data (three types of document) (2 points)
(5) Implement the sender.c (2 points)
(6) Deal with the “SIGINT” signal (keyboard to receiver) (1 point)

## V. Submission

Your assignment should be submitted to the course website by the due. Or you will receive penalty. At least three files should be included:

**1. receiver.c**
**2. Makefile (as well as other *.c)**
**3. README.txt**
**4. sender.c (option (2 points in this assignment))**

Since we will directly run your Makefile, therefore you can modify the names for .c files, but Makefile should compile your source into one or two executable files named **sender(option)** and **receiver**. In README.txt, please briefly state how to compile your program, and anything you have done besides the basic functionality. These files should be put inside a folder named with your student ID and you should compress the folder into a **.tar.gz** before submission. Please do
not use .rar or other file type.

The commands below will do the trick. Suppose your student ID is b02902000:

```
$ mkdir b02902000
$ cp Makefile README.txt *.c b02902000
$ tar –zcvf SPHW2_b02902000.tar.gz b02902000
$ rm –r b02902000
```

Please do NOT add executable files to the compressed file. Error in the submission file (such as files not in a directory named with your student ID, compiled binary not named **sender(option)** and **receiver**, and so on) may cause deduction of your credits.

## VI. Punishments

● Plagiarism
Plagiarism is strictly prohibited.

● Late punishment
Your credits of this assignment will be deducted 5% for each day of delay submission. (That is, you will lose all your credits on this assignment after 20 days delaying.) Even though your credits will be deducted for delaying, a late submission will still be much better than absence.