Header: Grade: FOR /20 DES /40 EXP /30 ORI /10 / TOT _100
Lab # 3

CS 5390 Summer 2017, Date of Submission: 07/21/17
Students Names: Adeel Malik (amalik@utep.edu)
Instructor: Yadira Jacquez

Section 1: Effort: 12 hours
- Planning and preparation: 1 hours
- Experiment: 7 hours (on simulator)
- Report writing: 1 hours

Section 2: Objectives
The objective of this experiment is to understand how message queues work and how tasks can send and receive messages to queues.

Section 3: Procedures and Results
Part A:
A1. Execute the function message from command line. Record and observe the output.
Answer: Running the command message, creates a maximum capacity FIFO message queue "mqId" and creates two tasks t1 Sender and t2 Receiver. One task sends message and the other receives it and its on going…
Host shell:

```
-> message
value = 337599800 = 0x141f5d38
-> i
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
tAioIoTask aioIoTask    0x14238750  50 Pend       0x1015e446        0x0     0
tNet0      ipcomNetTask 0x1423cdf0  50 Pend       0x1015dbab   0x3d0001     0
ipcom_sysl ipcom_syslog 0x1423f4a0  50 Pend       0x1015e446        0x0     0
tNetConf   ipnet_config 0x1429f900  50 Pend       0x1015dbab        0x0     0
tAioWait   aioWaitTask  0x1421f728  51 Pend       0x1015dbab        0x0     0
t1         Sender       0x141f0c10 110 Delay      0x101650a7        0x0    28
t2         Receiver     0x141f5d38 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
```

Target console:

```
-> SENDER 337579024 MESSAGE 0:
RECEIVER 337599800: message # 0 from Sender 337579024
SENDER 337579024 MESSAGE 1:
RECEIVER 337599800: message # 1 from Sender 337579024
SENDER 337579024 MESSAGE 2:
RECEIVER 337599800: message # 2 from Sender 337579024
SENDER 337579024 MESSAGE 3:
RECEIVER 337599800: message # 3 from Sender 337579024
SENDER 337579024 MESSAGE 4:
RECEIVER 337599800: message # 4 from Sender 337579024
SENDER 337579024 MESSAGE 5:
RECEIVER 337599800: message # 5 from Sender 337579024
SENDER 337579024 MESSAGE 6:
RECEIVER 337599800: message # 6 from Sender 337579024
```

A2. Check the running tasks:

a. Suspend (and later resume) the sender and receiver tasks independently. Check the status of message queue (show mqId). Record the output and explain what happened.

Answer: Suspending t1 (Sender) stops message sending and the t2 (Receiver) goes to pending and waits for a message till sender again sends it. The printing loop pauses on the Console. Status of message queue however, remains the same.

Host shell:

```
Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell
vxsim0_vx_config@ANACONDA
-> message
value = 339829664 = 0x144163a0
-> i(t1)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x141f0b40 110 Delay      0x101650a7        0x0     1
value = 0 = 0x0
-> i(t2)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t2         Receiver     0x144163a0 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId

Message Queue Id     : 0x144119d0
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 1
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
-> ts(t1)
value = 0 = 0x0
-> i(t1)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x141f0b40 110 Suspend    0x101650a7        0x0     0
value = 0 = 0x0
-> i(t2)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t2         Receiver     0x144163a0 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId

Message Queue Id     : 0x144119d0
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 1
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
->
```

Target console:

```
-> SENDER 337579024 MESSAGE 0:
RECEIVER 337600680: message # 0 from Sender 337579024
SENDER 337579024 MESSAGE 1:
RECEIVER 337600680: message # 1 from Sender 337579024
SENDER 337579024 MESSAGE 2:
RECEIVER 337600680: message # 2 from Sender 337579024
SENDER 337579024 MESSAGE 3:
RECEIVER 337600680: message # 3 from Sender 337579024
SENDER 337579024 MESSAGE 4:
RECEIVER 337600680: message # 4 from Sender 337579024
SENDER 337579024 MESSAGE 5:
RECEIVER 337600680: message # 5 from Sender 337579024
SENDER 337579024 MESSAGE 6:
RECEIVER 337600680: message # 6 from Sender 337579024
SENDER 337579024 MESSAGE 7:
RECEIVER 337600680: message # 7 from Sender 337579024
SENDER 337579024 MESSAGE 8:
RECEIVER 337600680: message # 8 from Sender 337579024
SENDER 337579024 MESSAGE 9:
RECEIVER 337600680: message # 9 from Sender 337579024
```

Resuming t1 (sender) and suspending t2 (receiver) stops the receiver task from receiving the messages and the sender keeps on sending messages till the queue is full, in this case when the queue will have 99 messages and then t1 (sender) also gets to pending or blocked state.

Target console:

```
SENDER 337578816 MESSAGE 106:
SENDER 337578816 MESSAGE 107:
SENDER 337578816 MESSAGE 108:
SENDER 337578816 MESSAGE 109:
SENDER 337578816 MESSAGE 110:
SENDER 337578816 MESSAGE 111:
SENDER 337578816 MESSAGE 112:
SENDER 337578816 MESSAGE 113:
SENDER 337578816 MESSAGE 114:
SENDER 337578816 MESSAGE 115:
SENDER 337578816 MESSAGE 116:
SENDER 337578816 MESSAGE 117:
SENDER 337578816 MESSAGE 118:
SENDER 337578816 MESSAGE 119:
SENDER 337578816 MESSAGE 120:
```

Host Shell:

Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell

vxsim0_vx_config@ANACONDA

```
-> tr(t1)
value = 0 = 0x0
-> ts(t2)
value = 0 = 0x0
-> show mqId

Message Queue Id      : 0x144119d0
Task Queueing         : FIFO
Message Byte Len      : 50
Messages Max          : 100
Messages Queued       : 2
Receivers Blocked     : 0
Send Timeouts         : 0
Receive Timeouts      : 0
Options               : 0x0        MSG_Q_FIFO

VxWorks Events
--------------
Registered Task       : NONE
Event(s) to Send      : N/A
Options               : N/A
value = 0 = 0x0
-> i(t1)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x141f0b40 110 Delay      0x101650a7        0x0    40
value = 0 = 0x0
-> i(t2)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t2         Receiver     0x144163a0 110 Suspend    0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId
Message Queue Id      : 0x144119d0
Task Queueing         : FIFO
Message Byte Len      : 50
Messages Max          : 100
Messages Queued       : 16
Receivers Blocked     : 0
Send Timeouts         : 0
Receive Timeouts      : 0
Options               : 0x0        MSG_Q_FIFO

VxWorks Events
--------------
Registered Task       : NONE
Event(s) to Send      : N/A
Options               : N/A
value = 0 = 0x0
```

```
Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell
vxsim0_vx_config@ANACONDA
-> show mqId

Message Queue Id      : 0x144119d0
Task Queueing         : FIFO
Message Byte Len      : 50
Messages Max          : 100
Messages Queued       : 99
Receivers Blocked     : 0
Send Timeouts         : 0
Receive Timeouts      : 0
Options               : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task       : NONE
Event(s) to Send      : N/A
Options               : N/A


value = 0 = 0x0
-> i(t1)
  NAME         ENTRY         TID     PRI   STATUS        PC        ERRNO    DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x141f0b40 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
-> i(t2)
  NAME         ENTRY         TID     PRI   STATUS        PC        ERRNO    DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t2         Receiver     0x144163a0 110 Suspend    0x1015b8eb        0x0     0
value = 0 = 0x0
->
```

b. Suspend (and later resume) both tasks observing the tasks status. Check the status of message queue (show mqId). Record the output and explain what happened.

Answer: As soon as t1 (sender) is blocked t2 (receiver) goes to pending state since there is nothing to receive. When t2 (receiver) is blocked, t2 (sender) keeps on sending till the message queue is full and then sender also gets blocked. As soon as the receiver resumes, it gets all messages from the message queue and sender starts sending again. Images above and below explain the same since I have already observed the tasks along with the queue as well.

Target console:

```
Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles
vxsim0_vx_config@ANACONDA
RECEIVER 339829664: message # 135 from Sender 337578816
RECEIVER 339829664: message # 136 from Sender 337578816
RECEIVER 339829664: message # 137 from Sender 337578816
RECEIVER 339829664: message # 138 from Sender 337578816
RECEIVER 339829664: message # 139 from Sender 337578816
RECEIVER 339829664: message # 140 from Sender 337578816
SENDER 337578816 MESSAGE 141:
RECEIVER 339829664: message # 141 from Sender 337578816
SENDER 337578816 MESSAGE 142:
RECEIVER 339829664: message # 142 from Sender 337578816
SENDER 337578816 MESSAGE 143:
RECEIVER 339829664: message # 143 from Sender 337578816
```

A3. With the sender(s) task(s) suspended and the receiver(s) pending, send a message from the shell command line. Show the command you used to send the message. How have message queue changed?

Answer: When sender is suspended and receiver is pending, I sent a message using the command "**msgQSend mqId, "Hello World", 12**" and the receiver immediately receives the message and displays it. The message queue however, remains the same.

Host shell:

```
Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell
vxsim0_vx_config@ANACONDA
-> message
value = 338026856 = 0x1425e168
-> ts(t1)
value = 0 = 0x0
-> i(t1)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x1425dd20 110 Suspend    0x101650a7        0x0     0
value = 0 = 0x0
-> i(t2)
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t2         Receiver     0x1425e168 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId

Message Queue Id     : 0x1425c360
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 1
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
-> msgQSend mqId, "Hello World", 12
value = 0 = 0x0
-> show mqId

Message Queue Id     : 0x1425c360
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 1
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
```

Target console:

Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles

vxsim0_vx_config@ANACONDA

```
-> SENDER 338025760 MESSAGE 0:
RECEIVER 338026856: message # 0 from Sender 338025760
SENDER 338025760 MESSAGE 1:
RECEIVER 338026856: message # 1 from Sender 338025760
SENDER 338025760 MESSAGE 2:
RECEIVER 338026856: message # 2 from Sender 338025760
SENDER 338025760 MESSAGE 3:
RECEIVER 338026856: message # 3 from Sender 338025760
SENDER 338025760 MESSAGE 4:
RECEIVER 338026856: message # 4 from Sender 338025760
SENDER 338025760 MESSAGE 5:
RECEIVER 338026856: message # 5 from Sender 338025760
SENDER 338025760 MESSAGE 6:
RECEIVER 338026856: message # 6 from Sender 338025760
SENDER 338025760 MESSAGE 7:
RECEIVER 338026856: message # 7 from Sender 338025760

-> RECEIVER 338026856: Hello World
```

A4. Resume the Sender task and suspend the Receiver task. Show how to receive a message from the shell. Create an "urgent" message on the message queue. Repeat your command above to receive the message from the shell (HINT: you will need to create a buffer from the shell to store the received message). Explain and comment on the results?

Answer:

Executed the following commands:

message	//running the program
ts(t1)	//suspending t1 (sender)
ts(t2)	//suspending t2 (receiver)
tr(t1)	//resuming t1 (sender)
str = "This is an Urgent Message\n"	//urgent message str
msgQSend mqId, str, strlen(str)+1, 0, **1**	//sending as an urgent message (last parameter 1 is priority)
bfr = calloc (100, 1)	//creating receive buffer
msgQReceive mqId, bfr, 100, 1	//reading the queue
printf bfr	//displaying the result

Host shell:

```
-> message
value = 337604304 = 0x141f6ed0
-> ts(t1)
value = 0 = 0x0
-> ts(t2)
value = 0 = 0x0
-> tr(t1)
value = 0 = 0x0
-> str = "This is an urgent message\n"
str = 0x141ba618: value = 337356248 = 0x141ba5d8
-> msgQSend mqId, str, strlen(str)+1, 0, 1
value = 0 = 0x0
-> bfr = calloc (100, 1)
bfr = 0x141ba5f4: value = 337607960 = 0x141f7d18
-> msgQReceive mqId, bfr, 100, 1
value = 27 = 0x1b = _vx_offset_HASH_TBL_pHashTbl + 0x3
-> printf bfr
This is an urgent message
value = 26 = 0x1a = _vx_offset_HASH_TBL_pHashTbl + 0x2
```

A5. Experiment with spawning 2-3 receiver tasks and/or 2-3 new sender tasks. Analyze the output, task status and the queue status while suspending/resuming the tasks. Check the status of the message queue while experimenting as in the point above. Describe explicitly how do you do that and what did you learned.

Answer:

When we have 2 receiving tasks, both of them receive message one by one from the sender task. When we have 3 receiving tasks, they receive one by one, one after another. Queue status shows that the receivers blocked by the queue are 3.

Target console:

```
-> SENDER 339449536 MESSAGE 0:
RECEIVER 339462624: message # 0 from Sender 339449536
SENDER 339449536 MESSAGE 1:
RECEIVER 339673104: message # 1 from Sender 339449536
SENDER 339449536 MESSAGE 2:
RECEIVER 339686192: message # 2 from Sender 339449536
SENDER 339449536 MESSAGE 3:
RECEIVER 339462624: message # 3 from Sender 339449536
SENDER 339449536 MESSAGE 4:
RECEIVER 339673104: message # 4 from Sender 339449536
SENDER 339449536 MESSAGE 5:
RECEIVER 339686192: message # 5 from Sender 339449536
SENDER 339449536 MESSAGE 6:
RECEIVER 339462624: message # 6 from Sender 339449536
SENDER 339449536 MESSAGE 7:
RECEIVER 339673104: message # 7 from Sender 339449536
SENDER 339449536 MESSAGE 8:
RECEIVER 339686192: message # 8 from Sender 339449536
```

Host shell:

```
Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell
vxsim0_vx_config@ANACONDA
-> message
value = 339686192 = 0x143f3330
-> i
  NAME        ENTRY          TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x143b96c0 110 Delay      0x101650a7        0x0    27
t2         Receiver     0x143bc9e0 110 Pend       0x1015b8eb        0x0     0
t3         Receiver     0x143f0010 110 Pend       0x1015b8eb        0x0     0
t4         Receiver     0x143f3330 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
-> show(mqId)

Message Queue Id     : 0x143b7d00
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 3
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
```

Adding new receiving task

```
if((receiverId2 = taskSpawn("t3",110,0x100,2000,(FUNCPTR)Receiver,0,0,0,0,0,0,0, 0,0,0)) == ERROR)
        printf("taskSpawn taskThree failed\n");
```

Adding new sender task

```
if((senderId2 = taskSpawn("t5",110,0x100,2000,(FUNCPTR)Sender,0,0,0,0,0,0,0, 0,0,0)) == ERROR)
        printf("taskSpawn taskFive failed\n");
```

When new sender task is added, sender tasks send messages one after another, and receiving tasks also receive messages one after another. 2 Sender tasks send messages first and 2 of the receiving receive on FIFO basis. When one more sender task is added, 3 sender tasks send message one after another and then 3 receiving tasks receive messages on after another. When a sender task is suspended then the receiving tasks receive messages by turns.

Host shell:

Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell

vxsim0_vx_config@ANACONDA

```
-> message
value = 338027328 = 0x1425e340
-> show mqId

Message Queue Id     : 0x141f49f8
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 3
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A


value = 0 = 0x0
-> i
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x141f63b8 110 Delay      0x101650a7        0x0    56
t5         Sender       0x141f66c8 110 Delay      0x101650a7        0x0     0
t6         Sender       0x141f69d8 110 Delay      0x101650a7        0x0     0
t2         Receiver     0x1425dd20 110 Pend       0x1015b8eb        0x0     0
t3         Receiver     0x1425e030 110 Pend       0x1015b8eb        0x0     0
t4         Receiver     0x1425e340 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
```

Target console:

```
-> SENDER 337601464 MESSAGE 0:
SENDER 337602248 MESSAGE 0:
SENDER 337603032 MESSAGE 0:
RECEIVER 338025760: message # 0 from Sender 337601464
RECEIVER 338026544: message # 0 from Sender 337602248
RECEIVER 338027328: message # 0 from Sender 337603032
SENDER 337601464 MESSAGE 1:
SENDER 337602248 MESSAGE 1:
SENDER 337603032 MESSAGE 1:
RECEIVER 338025760: message # 1 from Sender 337601464
RECEIVER 338026544: message # 1 from Sender 337602248
RECEIVER 338027328: message # 1 from Sender 337603032
SENDER 337602248 MESSAGE 20:
SENDER 337603032 MESSAGE 20:
RECEIVER 338025760: message # 20 from Sender 337602248
RECEIVER 338026544: message # 20 from Sender 337603032
SENDER 337602248 MESSAGE 21:
SENDER 337603032 MESSAGE 21:
RECEIVER 338027328: message # 21 from Sender 337602248
RECEIVER 338025760: message # 21 from Sender 337603032
SENDER 337602248 MESSAGE 22:
SENDER 337603032 MESSAGE 22:
RECEIVER 338026544: message # 22 from Sender 337602248
RECEIVER 338027328: message # 22 from Sender 337603032
SENDER 337603032 MESSAGE 28:
RECEIVER 338026544: message # 28 from Sender 337603032
SENDER 337603032 MESSAGE 29:
RECEIVER 338027328: message # 29 from Sender 337603032
SENDER 337603032 MESSAGE 30:
RECEIVER 338025760: message # 30 from Sender 337603032
SENDER 337603032 MESSAGE 31:
RECEIVER 338026544: message # 31 from Sender 337603032
SENDER 337603032 MESSAGE 32:
RECEIVER 338027328: message # 32 from Sender 337603032
SENDER 337603032 MESSAGE 33:
RECEIVER 338025760: message # 33 from Sender 337603032
```

Note: The above results are in the case of same priority levels for each task. When the priority is changed, then the sending and receiving pattern also changes based on the priority, high priority task sends and receives task before the other tasks.

When receiving tasks are blocked, the queue status reduces number of blocked tasks. When all receiving tasks are blocked the queue starts filling towards max capacity. As soon as receiving tasks are released, they receive messages from queue.

Host shell:

Feds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles | Debug Command Shell

vxsim0_vx_config@ANACONDA

```
message
value = 340066320 = 0x14450010
-> i
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x1425ec10  90 Delay      0x101650a7        0x0    37
t4         Receiver     0x14450010  90 Pend       0x1015b8eb        0x0     0
t5         Sender       0x14433020 100 Delay      0x101650a7        0x0     0
t3         Receiver     0x1443c980 100 Pend       0x1015b8eb        0x0     0
t6         Sender       0x14436340 110 Delay      0x101650a7        0x0     0
t2         Receiver     0x14439660 110 Pend       0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId

Message Queue Id     : 0x141f63b8
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 3
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
-> ts(t1)
value = 0 = 0x0
-> ts(t2)
value = 0 = 0x0
-> ts(t3)
value = 0 = 0x0
-> show mqId

Message Queue Id     : 0x141f63b8
Task Queueing        : FIFO
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 0
Receivers Blocked    : 1
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
-> ts(t4)
value = 0 = 0x0
```

```
-> i
  NAME        ENTRY         TID    PRI   STATUS        PC       ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x1425ec10  90 Suspend    0x101650a7        0x0     0
t4         Receiver     0x14450010  90 Pend+S     0x1015b8eb        0x0     0
t5         Sender       0x14433020 100 Delay      0x101650a7        0x0    25
t3         Receiver     0x1443c980 100 Suspend    0x1015b8eb        0x0     0
t6         Sender       0x14436340 110 Delay      0x101650a7        0x0     0
t2         Receiver     0x14439660 110 Suspend    0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId

Message Queue Id    : 0x141f63b8
Task Queueing       : FIFO
Message Byte Len    : 50
Messages Max        : 100
Messages Queued     : 13
Receivers Blocked   : 0
Send Timeouts       : 0
Receive Timeouts    : 0
Options             : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task     : NONE
Event(s) to Send    : N/A
Options             : N/A
value = 0 = 0x0
-> ts(t1)
value = 0 = 0x0
-> ts(t5)
value = 0 = 0x0
-> ts(t6)
value = 0 = 0x0
```

```
-> i
  NAME        ENTRY         TID    PRI   STATUS        PC       ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
t1         Sender       0x1425ec10  90 Suspend    0x101650a7        0x0     0
t4         Receiver     0x14450010  90 Suspend    0x1015b8eb        0x0     0
t5         Sender       0x14433020 100 Suspend    0x101650a7        0x0     0
t3         Receiver     0x1443c980 100 Suspend    0x1015b8eb        0x0     0
t6         Sender       0x14436340 110 Suspend    0x101650a7        0x0     0
t2         Receiver     0x14439660 110 Suspend    0x1015b8eb        0x0     0
value = 0 = 0x0
-> show mqId

Message Queue Id    : 0x141f63b8
Task Queueing       : FIFO
Message Byte Len    : 50
Messages Max        : 100
Messages Queued     : 60
Receivers Blocked   : 0
Send Timeouts       : 0
Receive Timeouts    : 0
Options             : 0x0       MSG_Q_FIFO

VxWorks Events
--------------
Registered Task     : NONE
Event(s) to Send    : N/A
Options             : N/A
value = 0 = 0x0
```

Target console:

Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Console | Target Consoles

vxsim0_vx_config@ANACONDA

```
-> SENDER 338029584 MESSAGE 0:
RECEIVER 340066320: message # 0 from Sender 338029584
SENDER 339947552 MESSAGE 0:
RECEIVER 339986816: message # 0 from Sender 339947552
SENDER 339960640 MESSAGE 0:
RECEIVER 339973728: message # 0 from Sender 339960640
SENDER 338029584 MESSAGE 1:
RECEIVER 340066320: message # 1 from Sender 338029584
SENDER 339947552 MESSAGE 1:
RECEIVER 339986816: message # 1 from Sender 339947552
SENDER 339960640 MESSAGE 1:
RECEIVER 339973728: message # 1 from Sender 339960640
SENDER 338029584 MESSAGE 2:
RECEIVER 340066320: message # 2 from Sender 338029584
SENDER 339947552 MESSAGE 2:
RECEIVER 339986816: message # 2 from Sender 339947552
```

<u>Part B:</u>

B1. Write a new program to implement Client-Server scenario with three Clients sending messages to a Server. Have the three Clients each send messages at a different rate (20, 40, 60 and ticks). The message shall be a string with the client identifier and a message number (e.g. "clientId-#"). The Server shall respond to each client with the same string appending "received at - <timestamp>" (where the <timestamp> is the time of getting the message). Each message shall be displayed three times: (a) upon creation by the client, (b) upon receiving by the server and addition of the timestamp by the server (c) upon receiving back by the client (adding second timestamp to identify the time of receiving the message back to the client). Include the code, excerpts of output, and explain the results of executing your program.

NOTE1: Consider spawning three threads and use only one client function. You may consider using semaphores if you need any synchronization or mutual exclusion. Also think about thread priorities. Describe how the program runs to prove that it works as specified.

NOTE2: See appendix for help with the timestamp implementation.

Answer: I have added message receiving section in the else part of “Sender” method of the code provided and renamed it as client and a sending section in the “Receive” method’s else section and renamed it as server. On top of these, I have added a mutex semaphore with WAIT_FOREVER to ensure message delivery. Additionally, I have formatted the message format as requested to clearly show the message synchronization. Message task with lower ticks repeats quickly then the other tasks.

The code is as below:

```
#include <vxWorks.h>      /* Always include this as the first thing in every program */
#include <stdio.h>        /* Always include this as the first thing in every program */
#include <unistd.h>
#include <time.h>         /* we use clock_gettime */
#include <taskLib.h>      /* we use tasks */
#include <sysLib.h>       /* we use sysClk... */
```

```
#include <semLib.h>		/* we use semaphores */
#include <logLib.h>		/* we use logMsg rather than printf */
#include <msgQLib.h>		/* we use message queues */

/* function prototypes */
void Client(int);
void Server(void);

/* defines */
#define MAX_MESSAGES 100
#define MSG_SIZE 50

/* globals */
MSG_Q_ID mqId;
SEM_ID semMQ;
struct timespec timer;
int sender1, sender2, sender3, server_task;

/* Function to create semaphore and message queue */
void createSemQueue (){
	mqId = msgQCreate(MAX_MESSAGES, MSG_SIZE, MSG_Q_PRIORITY);//Max capacity priority message que
	semMQ = semBCreate(0, 1); //Binary semaphore FIFO and Full
}

/* Run program */
void message(){
	createSemQueue();

	timer.tv_sec = 0;
	clock_settime(CLOCK_REALTIME, &timer);
	//Client one at 20 ticks
	if((sender1 = taskSpawn("ClientOne", 110, 0x100, 2000, (FUNCPTR)Client, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0)) ==
ERROR)
		printf("taskSpawn ClientOne failed\n");
	//Client two at 40 ticks
	if((sender2 = taskSpawn("ClientTwo", 110, 0x100, 2000, (FUNCPTR)Client, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0)) ==
ERROR)
		printf("taskSpawn ClientTwo failed\n");
	//Client three at 60 ticks
	if((sender3 = taskSpawn("ClientThree", 110, 0x100, 2000, (FUNCPTR)Client, 60, 0, 0, 0, 0, 0, 0, 0, 0, 0)) ==
ERROR)
		printf("taskSpawn ClientThree failed\n");

	if((server_task = taskSpawn("ServerOne", 110, 0x100, 2000, (FUNCPTR)Server, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)) ==
ERROR)
		printf("taskSpawn Server failed\n");
}

/* Client 1 task that writes to the message queue */
void Client(int delayTicks){
	int i = 0;
	while (1) {
		char message[MSG_SIZE];
```

```
		semTake(semMQ, WAIT_FOREVER);

		sprintf(message, "Message#%d from Client#%d",i, taskIdSelf());
		printf("Client#%d, Message: %d:\n",taskIdSelf(), i++); /* print what is sent */

		if(msgQSend(mqId, message, MSG_SIZE, WAIT_FOREVER, MSG_PRI_NORMAL) == ERROR){
			printf("msgQSend failed on Client#%d.\n",taskIdSelf());
		} else {
			taskDelay(10);//delay to let the other tasks run and wait for incoming message

			if (msgQReceive(mqId, message, MSG_SIZE, WAIT_FOREVER) == ERROR) {
				printf("msgQReceive failed on Client#%d.\n", taskIdSelf());
			} else {
				clock_gettime(CLOCK_REALTIME, &timer);
				printf("Client#%d received Server reply at %d Seconds and reply is: %s\n",
taskIdSelf(), timer.tv_sec, message);

	printf("*****************************************************************************
****************\n\n");
			}
		}
		semGive(semMQ);
		taskDelay(delayTicks);
	}
}

/* tasks that reads from the message queue */
void Server(){
	while(1){

		char message[MSG_SIZE];
		if (msgQReceive(mqId, message, MSG_SIZE, WAIT_FOREVER) == ERROR) {
			printf("msgQReceive in Server failed\n");
		} else {
			clock_gettime(CLOCK_REALTIME, &timer);
			printf("Server received: %s at %d Seconds.\n", message, timer.tv_sec);

			if(msgQSend(mqId, message, MSG_SIZE, WAIT_FOREVER, MSG_PRI_NORMAL) == ERROR){
				printf("Server: Unable to SEND message.\n", taskIdSelf());
			}
		}
		taskDelay(20);//Delay added to let the other tasks run…
	}
}
```

Host shell:

Feeds (0) | Tasks | Terminal 1 | Properties | Build Console | Target Consoles | Debug Command Shell

<terminated> vxsim0_vx_config@ANACONDA

```
-> message
value = 337602672 = 0x141f6870
-> i
  NAME        ENTRY         TID     PRI   STATUS        PC        ERRNO   DELAY
---------- ------------ ---------- --- ---------- ---------- ---------- -----
ClientOne  Client       0x1425e588 110 Delay      0x101650a7        0x0    10
ClientTwo  Client       0x1425e9d0 110 Delay      0x101650a7        0x0     1
ClientThre Client       0x141f6560 110 Delay      0x101650a7        0x0    19
ServerOne  Server       0x141f6870 110 Delay      0x101650a7        0x0    10
value = 0 = 0x0
-> show(mqId)

Message Queue Id     : 0x1425cb48
Task Queueing        : PRIORITY
Message Byte Len     : 50
Messages Max         : 100
Messages Queued      : 1
Receivers Blocked    : 0
Send Timeouts        : 0
Receive Timeouts     : 0
Options              : 0x1       MSG_Q_PRIORITY
VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
-> show(semMQ)

Semaphore Id         : 0x1425e508
Semaphore Type       : BINARY
Task Queueing        : FIFO
Pended Tasks         : 0
State                : EMPTY
Options              : 0x0       SEM_Q_FIFO

VxWorks Events
--------------
Registered Task      : NONE
Event(s) to Send     : N/A
Options              : N/A
value = 0 = 0x0
```

Target console:

```
Feds (0) | Tasks | Terminal 1 | Properties | Build Console | Target Consoles | Debug Command Shell
vxsim0_vx_config@ANACONDA
-> Client#338026512, Message: 0:
Server received: Message#0 from Client#338026512 at 0 Seconds.
Client#338026512 received Server reply at 0 Seconds and reply is: Message#0 from Client#338026512
*****************************************************************************************************

Client#338027296, Message: 0:
Server received: Message#0 from Client#338027296 at 1 Seconds.
Client#338027296 received Server reply at 1 Seconds and reply is: Message#0 from Client#338027296
*****************************************************************************************************

Client#338028080, Message: 0:
Client#338028080 received Server reply at 1 Seconds and reply is: Message#0 from Client#338028080
*****************************************************************************************************

Client#338026512, Message: 1:
Server received: Message#1 from Client#338026512 at 1 Seconds.
Client#338026512 received Server reply at 1 Seconds and reply is: Message#1 from Client#338026512
*****************************************************************************************************

Client#338027296, Message: 1:
Server received: Message#1 from Client#338027296 at 1 Seconds.
Client#338027296 received Server reply at 1 Seconds and reply is: Message#1 from Client#338027296
*****************************************************************************************************

Client#338026512, Message: 2:
Server received: Message#2 from Client#338026512 at 2 Seconds.
Client#338026512 received Server reply at 2 Seconds and reply is: Message#2 from Client#338026512
*****************************************************************************************************

Client#338028080, Message: 1:
Server received: Message#1 from Client#338028080 at 2 Seconds.
Client#338028080 received Server reply at 2 Seconds and reply is: Message#1 from Client#338028080
*****************************************************************************************************

Client#338026512, Message: 3:
```

<u>Section 4: Observations, Comments, and Lessons Learned</u>

I have learned about message queues, how they work, how to send and receive messages from message queues. I have used a binary semaphore to ensure message delivery between tasks. However, the server acknowledgment is missing randomly in between task synchronization. Due to shortage of time, I am unable to troubleshoot this part, otherwise program works fine.