Assignment 2

Kyle Pamintuan

CECS 326

Professor Ratana Ngo

12/15/17

Overview:

The purpose of this programming assignment is to help us better understand the concept of interprocess communication and basic synchronization that happens in the operating system. To do this, we will simulate them by creating a message queue program that behaves similarly to the operating system.

First, we will create 5 separate components in their own .cpp file. Two of which will be receivers that receive messages from a shared queue while the remaining three will be senders that send messages to the shared queue.

The first receiver, Receiver 1, will grab messages from the queue that contain data from Sender 251 or Sender 997. The second receiver, Receiver 2, will grab messages from the queue that contain data from Sender 257 or Sender 997.

All components will terminate based on their own requirements. These requirements may involve sending signals to each other via the queue.

**Receiver1.cpp**

#include <iostream>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <cstdlib>

#include <signal.h>

#include <unistd.h>

using namespace std;

struct buf

{

long mtype;

int data;

};

// Receiver 1 - Receives random multiples of 251 or 997 from the message queue

int main()

{

cout << "Receiver 1: PID = " << getpid() << endl;

// Create queue

int qid = msgget(ftok(".",'u'),IPC\_CREAT|0600);

buf msg;

int size = sizeof(msg)-sizeof(long);

bool term997 = false;

bool term251 = false;

while(!term251 || !term997)

{

// Receive messages from queue

msgrcv(qid, (struct msgbuf \*)&msg, size, 25, 0);

if(msg.data == 0) // Check for Sender 251's acknowledgment of termination

{

cout << "Sender 251 has terminated" << endl;

term251 = true;

}

else if(msg.data == 1) // Check for Sender 997's acknowledgment of termination

{

// Let Sender 997 know that it's acknowledgement of termination was received

msg.mtype = 26;

msg.data = 2;

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

cout << "Sender 997 has terminated" << endl;

term997 = true;

}

else if(msg.data == 101) // If we encounter Receiver 2's acknowledgement of termination, send it back.

{

msg.mtype = 25;

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

}

else if(msg.data % 251 == 0)

cout << "Received from Sender 251 : " << msg.data << endl;

else if(msg.data % 997 == 0)

cout << "Received from Sender 997 : " << msg.data << endl;

}

cout << "Receiver 1: terminating..."<<endl;

// Delete queue

msgctl(qid,IPC\_RMID,NULL);

exit(0);

}

**Receiver2.cpp**

#include <iostream>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <cstdlib>

#include <signal.h>

#include <unistd.h>

using namespace std;

struct buf

{

long mtype;

int data;

};

// Receiver 2 - Receives random multiples of 257 or 997 from the message queue

int main()

{

cout << "Receiver 2: PID = " << getpid() << endl;

// Message queue

int qid = msgget(ftok(".",'u'),0);

buf msg;

int size = sizeof(msg)-sizeof(long);

int count = 0;

// Terminate after receiving 5000 messages from the queue

while(count < 5000)

{

// Receive messages from queue with mtype = 25

msgrcv(qid, (struct msgbuf \*)&msg, size, 25, 0);

// Display messages

if(msg.data%257 == 0)

{

cout << "Received from Sender 257: " << msg.data << endl;

count++;

}

else if(msg.data%997 == 0)

{

cout << "Received from Sender 997: " << msg.data << endl;

count++;

}

}

// Send acknowledgement of termination to Sender 997 and Sender 257

msg.mtype = 26;

msg.data = 3;

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

cout << "Sent Messages Counter = " << count << endl;

cout << "Receiver 2: terminating..." << endl;

exit(0);

}

**Sender251.cpp**

#include <iostream>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <cstdlib>

#include <signal.h>

#include <unistd.h>

#include "get\_info.h"

using namespace std;

struct buf

{

long mtype;

int data;

};

// Sender 251 sends random multiples of 251 to the message queue

int main()

{

cout << "Sender 251: PID = " << getpid() << endl;

// Message queue

int qid = msgget(ftok(".",'u'),0);

buf msg;

int size = sizeof(msg)-sizeof(long);

srand(time(NULL));

bool term251 = false;

// Call get\_info method from get\_info.h file

//get\_info(qid, (struct msgbuf \*)&msg, size, 25);

msg.mtype = 25;

// Terminates by 'kill' command

//for(;;)

while(!term251)

{

// Continuosly generate numbers until we get a number divisble by 251

msg.data = rand();

while(msg.data%251 != 0)

{

msg.data = rand();

if(msg.data < 200)

{

cout << msg.data << " < 200" << endl;

// Send acknowledgement of before terminating

msg.data = 0;

msgsnd(qid, (struct msgbuf \*)&msg,size, 0);

term251 = true;

}

}

cout << "send to queue: " << msg.data << endl;

// Send message to queue

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

}

cout << "Sender 251: terminating..." << endl;

exit(0);

}

**Sender 257.cpp**

#include <iostream>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <cstdlib>

#include <signal.h>

#include <unistd.h>

using namespace std;

// Sender 257 sends random multiples of 257 to the message queue

struct buf

{

long mtype;

int data;

};

int main()

{

cout << "Sender 257: PID = " << getpid() << endl;

// Message queue

int qid = msgget(ftok(".",'u'),0);

buf msg;

int size = sizeof(msg)-sizeof(long);

srand(time(NULL));

bool term257 = false;

while(!term257)

{

// Check queue for Receiver's acknowledgement of termination

msgrcv(qid, (struct msgbuf \*)&msg, size, 26, 0);

if(msg.data == 3)

{

cout << "Receiver 2 has terminated" << endl;

term257 = true;

}

else

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

// Continuosly generate numbers until we get a number divisble by 257

msg.data = rand();

while(msg.data%257 != 0)

{

msg.data = rand();

}

cout << "send to queue: " << msg.data << endl;

// Send message to queue

msg.mtype = 25;

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

}

cout << "Sender 257: terminating..." << endl;

exit(0);

}

**Sender 997.cpp**

#include <iostream>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <stdio.h>

#include <string.h>

#include <cstdlib>

#include <signal.h>

#include <ctime>

#include <unistd.h>

using namespace std;

struct buf

{

long mtype;

int data;

};

// Sender 257 sends random multiples of 257 to the message queue

int main()

{

cout << "Sender 997: PID = " << getpid() << endl;

// Message queue

int qid = msgget(ftok(".",'u'),0);

buf msg;

int size = sizeof(msg)-sizeof(long);

srand(time(NULL));

bool termReq1 = false;

bool termReq2 = false;

int count = 0;

while (!termReq1 || !termReq2)

{

// Check queue if both receivers received our acknowledgement of termination

msgrcv(qid, (struct msgbuf \*)&msg, size, 26, 0);

if(msg.data == 2)

{

cout << "Receiver 1 received my acknowledment" << endl;

count++;

}

else if (msg.data == 3)

{

cout << "Receiver 2 received my acknowledment" << endl;

count++;

}

else

msgsnd(qid, (struct msgbuf \*)&msg, size, 0);

// If both receivers have received our acknowledgement of termination, we can now terminate

if (count >= 2)

termReq2 = true;

// Continuosly generate numbers until we get a number divisble by 257

msg.data = rand();

while (msg.data%997 != 0)

{

msg.data = rand();

// Terminate if random number is less than 100

if (msg.data < 100)

{

// Send acknowledgement of before terminating

msg.mtype = 26;

msg.data = 1;

msgsnd(qid, (struct msgbuf \*)&msg,size, 0); // One for Receiver 1

msgsnd(qid, (struct msgbuf \*)&msg,size, 0); // Another for Receiver 2

termReq1 = true;

}

}

cout << "send to queue: " << msg.data << endl;

// Send message to queue

msg.mtype = 25;

msgsnd(qid, (struct msgbuf \*)&msg,size, 0);

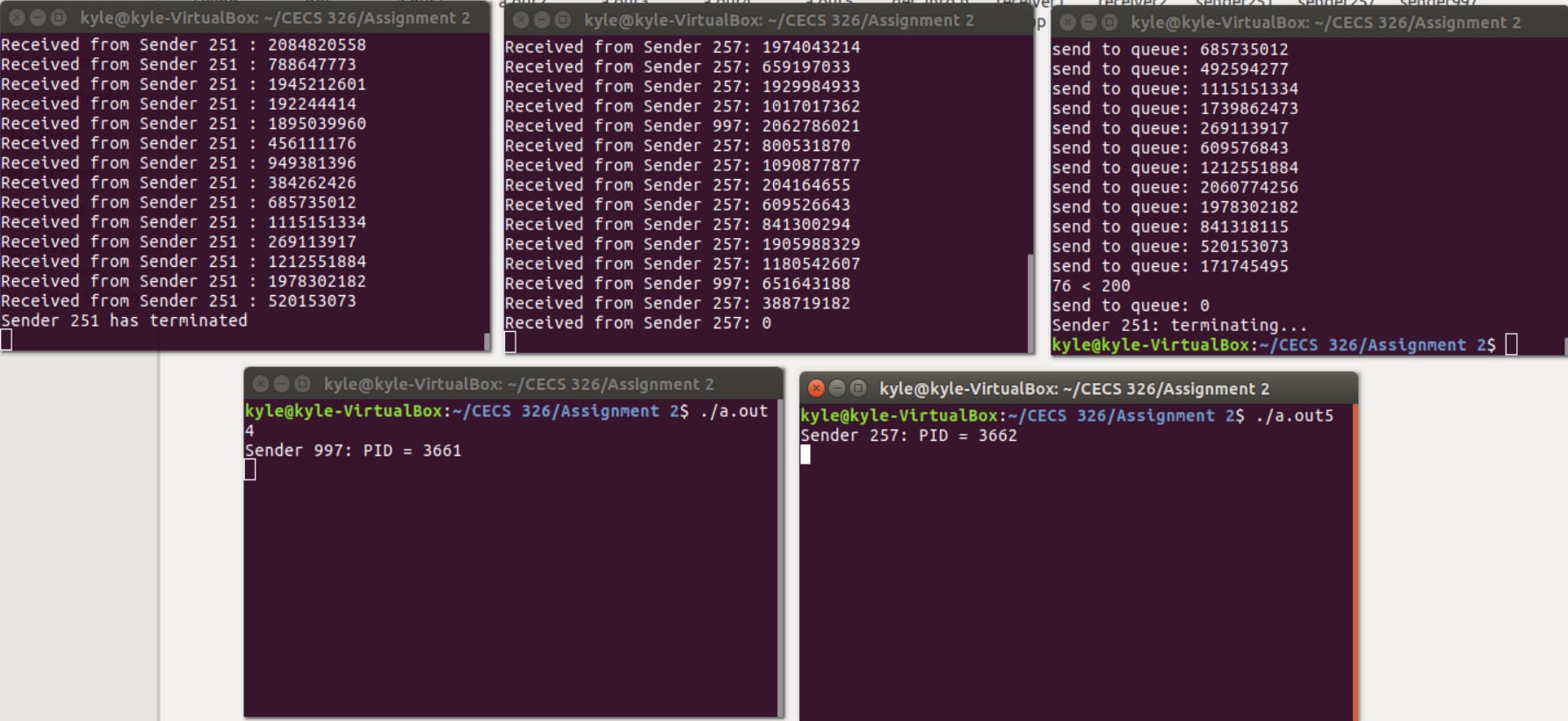
}

cout << "Sender 997: terminating..." << endl;

exit(0);

}

**Output:**

****

As you can see, there are some bugs in my program because Sender 997’s messages were not being created and outputted for some reason. And since the termination of Receiver 1 and 2 requires data from Sender 997, both Receivers never successfully terminated.

I’ve done my best to try and fix these bugs but I’ve run out of time. Originally, I thought that our deadline to turn in late assignments is one week after the due date. But I it wasn’t until the last week of the semester that I realized that we were actually able to turn in late assignments up until the day of the final. I made the mistake of reading the syllabus wrong. Had I realized this sooner, I would’ve definitely been able to fix those bugs.

Anyway, I hope I can receive some credit for this assignment as I worked so many hours to complete it. Thanks for a great semester professor! Happy Holidays!!!!

-Kyle