**Programming Project: Interprocess Communication and Synchronization**

Mark Tan

30 October 2017

CECS 326 MW

damtan1996@gmail.com

This program is an event tracking system in which three senders and two receivers communicate with each other using one message queue. The senders send events in the form of randomly generated integers that are divisible by a marker value such as 251, 997 and 257. The receivers get events from the senders, output the message and send an acknowledgement message back to the original sender. All five components are intended to terminate on a given condition, and execution must begin with the 251 & 997 receiver because that is where the message queue is allocated.

**251 Sender**

// Mark Tan

// 251sender.cpp

// This sender generates a random number divisible by 251 and sends it to a receiver. To terminate this program, use

// the kill function.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <cstring>

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include "getinfo.h"

#include <cstdlib>

using namespace std;

int ref\_num = 251; //defines the message source

int ref\_num\_a = 500; //defines the message destination

int main() {

struct buf {

long mtype;

int source;

int event\_num;

char message[50];

};

buf msg;

int event, rand\_num;

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

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

do{

//generates random number divisible by 251

do{

rand\_num = rand();

event = rand\_num;

}while(rand\_num % ref\_num != 0);

//creates and sends message to queue

cout << getpid() << ": sends event" << endl;

msg.event\_num = event;

msg.mtype = ref\_num\_a;

msg.source = ref\_num;

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

cout << endl;

}while(event != 0);

//sends termination message to receiver

strcpy(msg.message, "terminate");

get\_info(qid, (struct msgbuf \*)&msg, size, ref\_num\_a);

}

**997 Sender**

// Mark Tan

// 997sender.cpp

// This sender generates a random number divisible by 997 and sends it to a receiver.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <cstring>

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include <cstdlib>

using namespace std;

int ref\_num = 997; //defines the message source

int ref\_num\_a = 500; //defines one message destination

int ref\_num\_b = 750; //defines another message destination

int main() {

struct buf {

long mtype;

int source;

int event\_num;

char message[50];

};

buf msg;

int rand\_num;

int event = 100;

char msg\_rOne[50], msg\_rTwo[50];

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

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

do{

//generates random number divisible by 997

do{

rand\_num = rand();

event = rand\_num;

}while(rand\_num % ref\_num != 0);

//check for loop exit condition

if(event < 100) {

cout << "Event smaller than 100" << endl;

break;

}

//creates and sends message to queue for 251/997 receiver

cout << getpid() << ": sends event for 251/997 receiver" << endl;

msg.event\_num = event;

msg.mtype = ref\_num\_a;

msg.source = ref\_num;

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

//creates and sends message to queue for 997/257 receiver

cout << getpid() << ": sends event for 997/257 receiver" << endl;

msg.event\_num = event;

msg.mtype = ref\_num\_b;

msg.source = ref\_num;

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

//receives acknowledgement from 251 & 997 reciever

msgrcv(qid, (struct msgbuf \*)&msg, size, 320, 0);

cout << getpid() << ": gets ack from 251/997 receiver" << endl;

cout << "reply: " << msg.message << endl;

strcpy(msg\_rOne, msg.message);

//receives acknowledgement from 997 and 257 reciever

msgrcv(qid, (struct msgbuf \*)&msg, size, 300, 0);

cout << getpid() << ": gets ack from 997/257 receiver" << endl;

cout << "reply: " << msg.message << endl;

strcpy(msg\_rTwo, msg.message);

loop++;

cout << endl;

/\*sends message back to receivers once it gets

ack messages from 251 & 997 receiver and 997 & 257 receiver\*/

if(msg\_rOne != NULL && msg\_rTwo != NULL)

continue;

}while(event != 0);

//terminates when event number is less than 100

strcpy(msg.message, "terminate");

msg.mtype = ref\_num\_a;

msg.source = ref\_num;

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

cout << getpid() << ": now exits" << endl;

exit(0);

}

**257 Sender**

// Mark Tan

// 257sender.cpp

// This sender generates a random number divisible by 257 and sends it to a receiver.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <cstring>

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include <cstdlib>

using namespace std;

int ref\_num = 257; //defines the message source

int ref\_num\_b = 750; //defines the message destination

int main() {

struct buf {

long mtype;

int source;

int event\_num;

char message[50];

};

buf msg;

int event, rand\_num;

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

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

do{

//generates random number divisible by 257

do{

rand\_num = rand();

event = rand\_num;

}while(rand\_num % ref\_num != 0);

//check if 997 & 257 receiver has terminated

string msg\_check = string(msg.message), term\_msg = string("terminate");

if(msg\_check == term\_msg)

break;

//creates and sends message to queue

msg.event\_num = event;

cout << getpid() << ": sends event" << endl;

msg.mtype = ref\_num\_b;

msg.source = ref\_num;

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

//receive message from 997 & 257 receiver to terminate and exit program

msgrcv (qid, (struct msgbuf \*)&msg, size, 300, 0);

cout << endl;

}while(msg.message != "terminate");

//delete message queue

cout << getpid() << ": now exits" << endl;

msgctl (qid, IPC\_RMID, NULL);

exit(0);

}

**251 & 997 Receiver**

// Mark Tan

// 251\_997receiver.cpp

// This receiver gets a message from 251 and 997 senders and displays the message and sender.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <cstring>

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include <cstdlib>

using namespace std;

int main() {

struct buf {

long mtype;

int source;

int event\_num;

char message[50];

};

buf msg;

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

//creates message queue

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

do{

//read message

msgrcv(qid, (struct msgbuf \*)&msg, size, 500, 0);

cout << getpid() << ": gets event" << endl;

cout << "source: " << msg.source << endl;

cout << "event: " << msg.event\_num << endl;

//check if 251 & 997 senders have terminated

string msg\_check = string(msg.message), term\_msg = string("terminate");

if(msg\_check == term\_msg)

break;

//sends ack message to sender

strcpy(msg.message, "Got it!");

cout << getpid() << ": sends ack message to " << msg.source << endl;

msg.mtype = 320;

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

cout << endl;

}while(msg.message != "terminate");

//if messages say to terminate, then exit program

cout << getpid() << ": now exits" << endl;

exit(0);

}

**997 & 257 Receiver**

// Mark Tan

// 997\_257receiver.cpp

// This receiver gets a message from 997 and 257 senders and displays the message and sender.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <cstring>

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

#include <cstdlib>

using namespace std;

int main() {

struct buf {

long mtype;

int source;

int event\_num;

char message[50];

};

buf msg;

int size = sizeof(msg)-sizeof(long); int qid = msgget(ftok(".",'u'), 0);

int msg\_count;

do{

//reads message

msgrcv(qid, (struct msgbuf \*)&msg, size, 750, 0);

cout << getpid() << ": gets event" << endl;

cout << "source: " << msg.source << endl;

cout << "event: " << msg.event\_num << endl;

if(msg.event\_num != 0)

++msg\_count;

cout << "Messages Received: " << msg\_count << endl;

//sends ack message to sender

strcpy(msg.message, "Got it!");

cout << getpid() << ": sends ack message to " << msg.source << endl;

msg.mtype = 300;

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

cout << endl;

}while(msg\_count < 5000);

strcpy(msg.message, "terminate");

msg.mtype = 300;

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

cout << getpid() << ": now exits" << endl;

exit(0);

}