Lab 4 Report

# Communication Between Threads

## 1.a One Producer One Consumer Thread

This code uses one channel to pass data from a producer to a consumer. The code is in Appendix A. The producer and consumer work well together as can be seen from the output of the code.

**Results of Code:**

Results:

Embedded Message Passing Example Program

creating channel

could not create channel

user has been called

user connecting to channel.

user:go ahead, enter a string

friend has been called

friend connecting to channel.

friend: I am waiting to receive message from user.

11

user:waiting to send: = 11 =.

friend: received = 11 = from user.

user: received a reply from friend: = Got it! =

friend: I am waiting to receive message from user.

user:go ahead, enter a string

## 1.b Two Producers One Consumer Thread

There are two producers and only one consumer on the channel. The code can be seen in Appendix B. The consumer is able to receive the data which is from one of the producers although it does not receive from the other producer.

**Results of Code:**

Two Producers and one consumer

Embedded Message Passing Example Program

creating channel

could not create channel

user has been called

user connecting to channel.

user:go ahead, enter a string

friend has been called

Producer has been called

friend connecting to channel.

Producer connecting to channel

friend: I am waiting to receive message from user.

user wasnts to send DT;°friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

mcmcmcproducer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°user:waiting to send: = mcmcmc =.

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

mcmcm

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = mcmcmc = from user.

user: received a reply from friend: = Got it! =

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

user:go ahead, enter a string

producer: reply from friend Got it!user wasnts to send DT;°user:waiting to send: = mcmcm =.

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = mcmcm = from user.

user: received a reply from friend: = Got it! =

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

user:go ahead, enter a string

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

cmmcmcmdscproducer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°user:waiting to send: = cmmcmcmdsc =.

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

cmmproducer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = cmmcmcmdsc = from user.

user: received a reply from friend: = Got it! =

cmc

mfriend: I am waiting to receive message from user.

friend: received = DT;° = from user.

user:go ahead, enter a string

producer: reply from friend Got it!user wasnts to send DT;°user:waiting to send: = cmmcmc =.

cmcmc

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

producer: reply from friend Got it!user wasnts to send DT;°friend: I am waiting to receive message from user.

friend: received = cmmcmc = from user.

user: received a reply from friend: = Got it! =

friend: I am waiting to receive message from user.

friend: received = DT;° = from user.

user:go ahead, enter a string

producer: reply from friend Got it!user wasnts to send DT;°user:waiting to send: = mcmcmc =.

## 1.c One Producer and Several Consumers

Two consumers were coded with only one producer, the code can be found in Appendix C.

Embedded Message Passing Example Program

creating channel

could not create channel

user has been called

user connecting to channel.

user:go ahead, enter a string

friend has been called

consumer has been called

friend connecting to channel.

consumer connecting to channel.

friend: I am waiting to receive message from user.

consumer: I am waiting to receive message from user.

a

user:waiting to send: = a =.

consumer: received = a = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

b

user:waiting to send: = b =.

consumer: received = b = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

c

user:waiting to send: = c =.

consumer: received = c = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

d

user:waiting to send: = d =.

consumer: received = d = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

e

user:waiting to send: = e =.

consumer: received = e = from user.

user: received a reply from friend: = Got it! =

fconsumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = f =.

consumer: received = f = from user.

user: received a reply from friend: = Got it! =

g

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = g =.

consumer: received = g = from user.

user: received a reply from friend: = Got it! =

h

iconsumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = h =.

consumer: received = h = from user.

user: received a reply from friend: = Got it! =

j

kconsumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = i =.

consumer: received = i = from user.

user: received a reply from friend: = Got it! =

lconsumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = j =.

consumer: received = j = from user.

user: received a reply from friend: = Got it! =

m

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = k =.

consumer: received = k = from user.

user: received a reply from friend: = Got it! =

n

opconsumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = l =.

consumer: received = l = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = m =.

consumer: received = m = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = n =.

consumer: received = n = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

user:waiting to send: = op =.

consumer: received = op = from user.

user: received a reply from friend: = Got it! =

consumer: I am waiting to receive message from user.

user:go ahead, enter a string

# Communication Between Processes

## 2.a Experimenting with the forkerc program

## 2.b One Producer and One Consumer between Processes

Embedded Message Passing Example Program

creating channel

could not create channel

PID=49173, PPID=28694 I am the main program.

I am done waiting for PID[-1] with status 0

Fork now..

PID=53271, PPID=49173 I am the child.

friend has been called

Exec now..

friend connecting to channel.

user has been called

user connecting to channel.

friend: I am waiting to receive message from user.

user:go ahead, enter a string

122

user:waiting to send: = 122 =.

friend: received = 122 = from user.

user: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

5555

user:waiting to send: = 5555 =.

friend: received = 5555 = from user.

user: received a reply from friend: = Got it! =

484PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

8648

user:waiting to send: = 4848648 =.

friend: received = 4848648 = from user.

user: received a reply from friend: = Got it! =

7PID=49173, PPID=28694 I am the parent.

friend: I am waiting to receive message from user.

user:go ahead, enter a string

878

user:waiting to send: = 7878 =.

friend: received = 7878 = from user.

user: received a reply from friend: = Got it! =

8PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

48648

user:waiting to send: = 848648 =.

friend: received = 848648 = from user.

user: received a reply from friend: = Got it! =

48PID=49173, PPID=28694 I am the parent.

friend: I am waiting to receive message from user.

user:go ahead, enter a string

48

user:waiting to send: = 4848 =.

friend: received = 4848 = from user.

user: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

## 2.c Multiple Producer One Consumers

Embedded Message Passing Example Program

creating channel

could not create channel

PID=49173, PPID=28694 I am the main program.

I am done waiting for PID[-1] with status 1

Fork now..

PID=53271, PPID=49173 I am the child.

Exec now..

user has been called

user connecting to channel.

user:go ahead, enter a string

PID=49173, PPID=28694 I am the parent.

Exec now..

PID=49173, PPID=28694 I am the second child.

prod has been called

friend has been called

prod connecting to channel.

friend connecting to channel.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

2222PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

2

user:waiting to send: = 22222 =.

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = 22222 = from user.

user: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

222PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

22

user:waiting to send: = 22222 =.

2PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = 22222 = from user.

user: received a reply from friend: = Got it! =

2222

PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

friend: received = hello = from user.

user:waiting to send: = 22222 =.

prod: received a reply from friend: = Got it! =

222

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = 22222 = from user.

user: received a reply from friend: = Got it! =

222

22PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

friend: received = hello = from user.

user:waiting to send: = 222 =.

prod: received a reply from friend: = Got it! =

2

222PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = 222 = from user.

user: received a reply from friend: = Got it! =

22

2PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

user:waiting to send: = 222 =.

friend: received = hello = from user.

prod: received a reply from friend: = Got it! =

22222

PID=49173, PPID=28694 I am the parent.

prod:go ahead, enter a string

friend: I am waiting to receive message from user.

prod:waiting to send: = hello =.

friend: received = 222 = from user.

user: received a reply from friend: = Got it! =

PID=49173, PPID=28694 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

friend: received = hello = from user.

user:waiting to send: = 222 =.

prod: received a reply from friend: = Got it! =

## 2.d One Producer Multiple Consumer

Embedded Message Passing Example Program

creating channel

could not create channel

PID=77846, PPID=28695 I am the main program.

I am done waiting for PID[-1] with status 1

Fork now..

PID=81944, PPID=77846 I am the child.

Exec now..

user has been called

user connecting to channel.

user:go ahead, enter a string

PID=77846, PPID=28695 I am the parent.

Exec now..

PID=77846, PPID=28695 I am the second child.

friend has been called

friend has been called

friend connecting to channel.

friend connecting to channel.

friend: I am waiting to receive message from user.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

1111PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

user:waiting to send: = 1111 =.

friend: received = 1111 = from user.

user: received a reply from friend: = Got it! =

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

user:go ahead, enter a string

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

11

user:waiting to send: = 11 =.

friend: received = 11 = from user.

user: received a reply from friend: = Got it! =

1PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

1

1PID=77846, PPID=28695 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

user:waiting to send: = 11 =.

friend: received = 11 = from user.

user: received a reply from friend: = Got it! =

11PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

11PID=77846, PPID=28695 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

user:waiting to send: = 111 =.

friend: received = 111 = from user.

user: received a reply from friend: = Got it! =

1PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

11

PID=77846, PPID=28695 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

user:waiting to send: = 11111 =.

friend: received = 11111 = from user.

user: received a reply from friend: = Got it! =

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

user:go ahead, enter a string

friend: I am waiting to receive message from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

PID=77846, PPID=28695 I am the parent.

friend: I am waiting to receive message from user.

friend: received = = from user.

# Appendix A: messtestc.c Code

|  |
| --- |
| /\* |
|  | scriptname: mseetestc.c |
|  | author: Richard Tervo |
|  | purpose: to be a part of lab |
|  | description: todoloo |
|  |  |
|  | \*/ |
|  |  |
|  |  |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | #include <pthread.h> |
|  | #include <sys/neutrino.h> |
|  | #define MSGSIZE 80 |
|  | int chid; |
|  | int rcvid; |
|  | unsigned flags; |
|  | char smsg[MSGSIZE]; |
|  | char rmsg[MSGSIZE]; |
|  | char rdr[MSGSIZE]; |
|  | char rdrreply[MSGSIZE]; |
|  |  |
|  |  |
|  | int main(void) |
|  | { |
|  | puts("Embedded Message Passing |
|  | Example Program"); |
|  | puts("creating channel"); |
|  | flags = NULL; |
|  | chid = ChannelCreate(flags); |
|  | if (chid == ‑1){ |
|  | printf("could not create channel \n"); |
|  | return(‑1); |
|  | } |
|  | pthread\_create(NULL, NULL, &user, |
|  | NULL); |
|  | pthread\_create(NULL, NULL, &friend, |
|  | NULL); |
|  | sleep(120); |
|  | return(0); |
|  | } |
|  |  |
|  |  |
|  | void \* user() |
|  | { |
|  | int ucoid; |
|  | puts("user has been called"); |
|  | puts("user connecting to channel. \n"); |
|  | ucoid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | while(1) { |
|  | puts("user:go ahead, enter a string"); |
|  | gets(smsg); |
|  | printf("user:waiting to send: = %s =.\n",smsg); |
|  | MsgSend(ucoid, &smsg, sizeof(smsg), &rmsg, sizeof(rmsg)); |
|  | printf("user: received a reply from friend: = %s =\n", rmsg); |
|  | sleep(1); |
|  | } |
|  | } |
|  |  |
|  | void \* friend() |
|  | { |
|  | int fcoid; |
|  | puts("friend has been called"); |
|  | puts("friend connecting to channel. \n"); |
|  | fcoid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | strcpy(rdrreply, "Got it!"); |
|  | while(1) { |
|  | printf("friend: I am waiting to receive message from user.\n"); |
|  | rcvid = MsgReceive(chid, &rdr, sizeof(rdr), NULL); |
|  | printf("friend: received = %s = from user.\n", rdr); |
|  | MsgReply(rcvid, NULL, &rdrreply, sizeof(rdrreply)); |
|  | sleep(1); |
|  | } |
|  | } |

# Appendix B: two\_prod\_one\_cons.c Code

|  |
| --- |
| /\* |
|  | scriptname: two\_prod\_one\_cons.c |
|  | author: Richard Tervo |
|  | purpose: to be a part of lab |
|  | description: todoloo |
|  |  |
|  | \*/ |
|  |  |
|  |  |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | #include <pthread.h> |
|  | #include <sys/neutrino.h> |
|  | #define MSGSIZE 80 |
|  | int chid; |
|  | int rcvid; |
|  | unsigned flags; |
|  | char send\_message[MSGSIZE]; //send message |
|  | char receive\_message[MSGSIZE]; //receive message |
|  | char reader[MSGSIZE]; // |
|  | char readerreply[MSGSIZE]; |
|  |  |
|  | // Thread Definitions |
|  | void \* user() |
|  | { |
|  | // Sends the data from command line to the friend. |
|  | int ucoid; |
|  | puts("user has been called"); |
|  | puts("user connecting to channel. \n"); |
|  | ucoid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | while(1) { |
|  | puts("user:go ahead, enter a string"); |
|  | gets(send\_message); |
|  | printf("user:waiting to send: = %s =.\n",send\_message); |
|  | MsgSend(ucoid, &send\_message, sizeof(send\_message), &receive\_message, sizeof(receive\_message)); |
|  | printf("user: received a reply from friend: = %s =\n", receive\_message); |
|  | sleep(1); |
|  | } |
|  | } |
|  |  |
|  | void \* friend() |
|  | { |
|  | int fcoid; |
|  | puts("friend has been called"); |
|  | puts("friend connecting to channel. \n"); |
|  | fcoid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | strcpy(readerreply, "Got it!"); |
|  | while(1) { |
|  | printf("friend: I am waiting to receive message from user.\n"); |
|  | rcvid = MsgReceive(chid, &reader, sizeof(reader), NULL); |
|  | printf("friend: received = %s = from user.\n", reader); |
|  | MsgReply(rcvid, NULL, &readerreply, sizeof(readerreply)); |
|  | sleep(1); |
|  | } |
|  | } |
|  |  |
|  | void \* add\_producer() |
|  | { |
|  | int prod\_coid; |
|  | char temp[MSGSIZE]; |
|  | temp[0] = 'D';//(char[MSGSIZE]) |
|  | puts("Producer has been called"); |
|  | puts("Producer connecting to channel"); |
|  | prod\_coid=ConnectAttach(0,0,chid,0,NULL); |
|  | while(1){ |
|  | printf("user wasnts to send %s",temp); |
|  | MsgSend(prod\_coid, &temp, sizeof(temp), &receive\_message, sizeof(receive\_message)); |
|  | printf("producer: reply from friend %s", receive\_message); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | // MAIN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | int main(void) |
|  | { |
|  | puts("Embedded Message Passing Example Program"); |
|  | puts("creating channel"); |
|  | flags = NULL; |
|  | chid = ChannelCreate(flags); |
|  | if (chid==1){ |
|  | printf("could not create channel \n"); |
|  | //exit(‑1); |
|  | } |
|  | pthread\_create(NULL, NULL, &user,NULL); |
|  | pthread\_create(NULL, NULL, &friend,NULL); |
|  | pthread\_create(NULL,NULL, &add\_producer,NULL); |
|  | sleep(120); |
|  | return(0); |
|  | } |

# Appendix C: one\_prod\_two\_cons.c

|  |
| --- |
| /\* |
|  | scriptname: one\_prod\_two\_cons.c |
|  | author: Richard Tervo |
|  | purpose: to be a part of lab |
|  | description: todoloo |
|  |  |
|  | \*/ |
|  |  |
|  |  |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | #include <pthread.h> |
|  | #include <sys/neutrino.h> |
|  | #define MSGSIZE 80 |
|  | int chid; |
|  | int rcvid; |
|  | unsigned flags; |
|  | char send\_message[MSGSIZE]; //send message |
|  | char receive\_message[MSGSIZE]; //receive message |
|  | char reader[MSGSIZE]; // |
|  | char readerreply[MSGSIZE]; |
|  |  |
|  | // Thread Definitions |
|  | void \* user() |
|  | { |
|  | // Sends the data from command line to the friend. |
|  | int ucoid; |
|  | puts("user has been called"); |
|  | puts("user connecting to channel. \n"); |
|  | ucoid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | while(1) { |
|  | puts("user:go ahead, enter a string"); |
|  | gets(send\_message); |
|  | printf("user:waiting to send: = %s =.\n",send\_message); |
|  | MsgSend(ucoid, &send\_message, sizeof(send\_message), &receive\_message, sizeof(receive\_message)); |
|  | printf("user: received a reply from friend: = %s =\n", receive\_message); |
|  | sleep(1); |
|  | } |
|  | } |
|  |  |
|  | void \* friend() |
|  | { |
|  | int fcoid; |
|  | puts("friend has been called"); |
|  | puts("friend connecting to channel. \n"); |
|  | fcoid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | strcpy(readerreply, "Got it!"); |
|  | while(1) { |
|  | printf("friend: I am waiting to receive message from user.\n"); |
|  | rcvid = MsgReceive(chid, &reader, sizeof(reader), NULL); |
|  | printf("friend: received = %s = from user.\n", reader); |
|  | MsgReply(rcvid, NULL, &readerreply, sizeof(readerreply)); |
|  | sleep(1); |
|  | } |
|  | } |
|  |  |
|  | void \* add\_cons() |
|  | { |
|  | int cons\_coid; |
|  | puts("consumer has been called"); |
|  | puts("consumer connecting to channel. \n"); |
|  | cons\_coid = ConnectAttach(0, 0, chid, 0, NULL); |
|  | strcpy(readerreply, "Got it!"); |
|  | while(1) { |
|  | printf("consumer: I am waiting to receive message from user.\n"); |
|  | rcvid = MsgReceive(chid, &reader, sizeof(reader), NULL); |
|  | printf("consumer: received = %s = from user.\n", reader); |
|  | MsgReply(rcvid, NULL, &readerreply, sizeof(readerreply)); |
|  | sleep(1); |
|  | } |
|  | } |
|  |  |
|  | /\* |
|  | void \* add\_producer() |
|  | { |
|  | int prod\_coid; |
|  | char temp[MSGSIZE]; |
|  | temp[0] = 'D';//(char[MSGSIZE]) |
|  | puts("Producer has been called"); |
|  | puts("Producer connecting to channel"); |
|  | prod\_coid=ConnectAttach(0,0,chid,0,NULL); |
|  | while(1){ |
|  | printf("user wasnts to send %s",temp); |
|  | MsgSend(prod\_coid, &temp, sizeof(temp), &receive\_message, sizeof(receive\_message)); |
|  | printf("producer: reply from friend %s", receive\_message); |
|  | } |
|  | } |
|  | \*/ |
|  |  |
|  | // MAIN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | int main(void) |
|  | { |
|  | puts("Embedded Message Passing Example Program"); |
|  | puts("creating channel"); |
|  | flags = NULL; |
|  | chid = ChannelCreate(flags); |
|  | if (chid==1){ |
|  | printf("could not create channel \n"); |
|  | //exit(‑1); |
|  | } |
|  | pthread\_create(NULL, NULL, &user,NULL); |
|  | pthread\_create(NULL, NULL, &friend,NULL); |
|  | pthread\_create(NULL,NULL, &add\_cons,NULL); |
|  | sleep(120); |
|  | return(0); |
|  | } |

# Appendix D: forker.c Code

|  |
| --- |
| /\*\*\* |
|  | forker.c R.Tervo Jan 2013 |
|  | ------------------------------------- |
|  | Exploring fork() and exec() |
|  | This program monitors PID and PPID in |
|  | while spawning a second program. |
|  | ------------------------------------- |
|  | \*\*\*/ |
|  |  |
|  | #include <unistd.h> |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | #include <pthread.h> |
|  | #include <sys/neutrino.h> |
|  | #define MSGSIZE 80 |
|  | int chid; |
|  | int rcvid; |
|  | unsigned flags; |
|  | char send\_message[MSGSIZE]; //send message |
|  | char receive\_message[MSGSIZE]; //receive message |
|  | char reader[MSGSIZE]; // |
|  | char readerreply[MSGSIZE]; |
|  |  |
|  | pid\_t pidA, ppidA; // define PID storage |
|  | pid\_t pidB, ppidB; |
|  | pid\_t pidC, pid; |
|  |  |
|  |  |
|  | main( ) |
|  | { |
|  |  |
|  |  |
|  | int status; // status of wait() |
|  |  |
|  | puts("Embedded Message Passing Example Program"); |
|  | puts("creating channel"); |
|  | flags = NULL; |
|  | chid = ChannelCreate(flags); |
|  | if (chid==1){ |
|  | printf("could not create channel \n"); |
|  | //exit(‑1); |
|  | } |
|  |  |
|  | pidA = getpid(); // establish main program PID |
|  | ppidA = getppid(); |
|  | printf("PID=%i, PPID=%i I am the main program.\n", pidA, ppidA); |
|  |  |
|  | pid = wait(&status); // wait for child process to end (none) |
|  | printf("I am done waiting for PID[%i] with status %i\n",pid, status); |
|  |  |
|  | printf("Fork now..\n"); |
|  | fflush(stdout); // complete any pending prints |
|  |  |
|  | pidC = fork(); // returns child PID (or 0 if child) |
|  |  |
|  | pidB = getpid(); |
|  | ppidB = getppid(); |
|  |  |
|  | if(pidA != pidB) |
|  | { |
|  | printf("PID=%i, PPID=%i I am the child. \n", pidB, ppidB); |
|  |  |
|  | printf("Exec now..\n"); |
|  | fflush(stdout); // complete any pending prints |
|  | int ucoid; |
|  | puts("user has been called"); |
|  | puts("user connecting to channel. \n"); |
|  | ucoid = ConnectAttach(0, pidA, chid, 0, NULL); |
|  | while(1) { |
|  | puts("user:go ahead, enter a string"); |
|  | gets(send\_message); |
|  | printf("user:waiting to send: = %s =.\n",send\_message); |
|  | MsgSend(ucoid, &send\_message, sizeof(send\_message), &receive\_message, sizeof(receive\_message)); |
|  | printf("user: received a reply from friend: = %s =\n", receive\_message); |
|  | sleep(1); |
|  | } |
|  | //execl("./hello","hello",'\0'); // replace this process with another |
|  |  |
|  | printf(" ...we never print this line...\n"); |
|  | } else |
|  | { |
|  | int fcoid; |
|  | puts("friend has been called"); |
|  | puts("friend connecting to channel. \n"); |
|  | fcoid = ConnectAttach(0, pidB, chid, 0, NULL); |
|  | strcpy(readerreply, "Got it!"); |
|  | while(1) { |
|  | printf("friend: I am waiting to receive message from user.\n"); |
|  | rcvid = MsgReceive(chid, &reader, sizeof(reader), NULL); |
|  | printf("friend: received = %s = from user.\n", reader); |
|  | MsgReply(rcvid, NULL, &readerreply, sizeof(readerreply)); |
|  | sleep(1); |
|  | printf("PID=%i, PPID=%i I am the parent. \n", pidB, ppidB); |
|  | } |
|  |  |
|  | pid = wait(&status); // wait for child process to end (none) |
|  | printf("I am done waiting for PID[%i] with status %i\n",pid, status); |
|  | } |
|  | return(0); |
|  | } // end main |

# Appendix E: Multi\_Process\_Prod.c

|  |
| --- |
| /\*\*\* |
|  | forker.c R.Tervo Jan 2013 |
|  | ------------------------------------- |
|  | Exploring fork() and exec() |
|  | This program monitors PID and PPID in |
|  | while spawning a second program. |
|  | ------------------------------------- |
|  | \*\*\*/ |
|  |  |
|  | #include <unistd.h> |
|  | #include <stdio.h> |
|  | #include <stdlib.h> |
|  | #include <string.h> |
|  | #include <pthread.h> |
|  | #include <sys/neutrino.h> |
|  | #define MSGSIZE 80 |
|  | int chid; |
|  | int rcvid; |
|  | unsigned flags; |
|  | char send\_message[MSGSIZE]; //send message |
|  | char receive\_message[MSGSIZE]; //receive message |
|  | char reader[MSGSIZE]; // |
|  | char readerreply[MSGSIZE]; |
|  |  |
|  |  |
|  |  |
|  |  |
|  | main( ) |
|  | { |
|  | pid\_t pidA, ppidA; // define PID storage |
|  | pid\_t pidB, ppidB; |
|  | pid\_t pidC, pid; |
|  | pid\_t pidD, pidE; |
|  | pid\_t ppidE; |
|  |  |
|  | int status; // status of wait() |
|  |  |
|  | puts("Embedded Message Passing Example Program"); |
|  | puts("creating channel"); |
|  | flags = NULL; |
|  | chid = ChannelCreate(flags); |
|  | if (chid==1){ |
|  | printf("could not create channel \n"); |
|  | //exit(‑1); |
|  | } |
|  |  |
|  | pidA = getpid(); // establish main program PID |
|  | ppidA = getppid(); |
|  | printf("PID=%i, PPID=%i I am the main program.\n", pidA, ppidA); |
|  |  |
|  | pid = wait(&status); // wait for child process to end (none) |
|  | printf("I am done waiting for PID[%i] with status %i\n",pid, status); |
|  |  |
|  | printf("Fork now..\n"); |
|  | fflush(stdout); // complete any pending prints |
|  |  |
|  | pidC = fork(); // returns child PID (or 0 if child) |
|  |  |
|  | pidB = getpid(); |
|  | ppidB = getppid(); |
|  |  |
|  |  |
|  |  |
|  | if(pidA != pidB) |
|  | { |
|  | printf("PID=%i, PPID=%i I am the child. \n", pidB, ppidB); |
|  |  |
|  | printf("Exec now..\n"); |
|  | fflush(stdout); // complete any pending prints |
|  | int ucoid; |
|  | puts("user has been called"); |
|  | puts("user connecting to channel. \n"); |
|  | ucoid = ConnectAttach(0, pidA, chid, 0, NULL); |
|  | while(1) { |
|  | puts("user:go ahead, enter a string"); |
|  | gets(send\_message); |
|  | printf("user:waiting to send: = %s =.\n",send\_message); |
|  | MsgSend(ucoid, &send\_message, sizeof(send\_message), &receive\_message, sizeof(receive\_message)); |
|  | printf("user: received a reply from friend: = %s =\n", receive\_message); |
|  | sleep(1); |
|  | } |
|  | } |
|  | else |
|  | { |
|  | pidD= fork(); |
|  | pidE=getpid(); |
|  | ppidE=getppid(); |
|  | if(pidA != pidE){ |
|  | printf("PID=%i, PPID=%i I am the parent. \n", pidA, ppidA); |
|  |  |
|  | printf("Exec now..\n"); |
|  | fflush(stdout); // complete any pending prints |
|  | int prodcoid; |
|  | puts("prod has been called"); |
|  | puts("prod connecting to channel. \n"); |
|  | prodcoid = ConnectAttach(0, pidA, chid, 0, NULL); |
|  | while(1) { |
|  | puts("prod:go ahead, enter a string"); |
|  | //gets(send\_message); |
|  | strcpy(send\_message,"hello"); |
|  | printf("prod:waiting to send: = %s =.\n",send\_message); |
|  | MsgSend(prodcoid, &send\_message, sizeof(send\_message), &receive\_message, sizeof(receive\_message)); |
|  | printf("prod: received a reply from friend: = %s =\n", receive\_message); |
|  | sleep(1); |
|  | } |
|  | } |
|  | else{ |
|  | //-- |
|  | printf("PID=%i, PPID=%i I am the second child. \n", pidE, ppidE); |
|  | int fcoid; |
|  | puts("friend has been called"); |
|  | puts("friend connecting to channel. \n"); |
|  | fcoid = ConnectAttach(0, pidA, chid, 0, NULL); |
|  | strcpy(readerreply, "Got it!"); |
|  | while(1) { |
|  | printf("friend: I am waiting to receive message from user.\n"); |
|  | rcvid = MsgReceive(chid, &reader, sizeof(reader), NULL); |
|  | printf("friend: received = %s = from user.\n", reader); |
|  | MsgReply(rcvid, NULL, &readerreply, sizeof(readerreply)); |
|  | sleep(1); |
|  | printf("PID=%i, PPID=%i I am the parent. \n", pidB, ppidB); |
|  | } |
|  | pid = wait(&status); // wait for child process to end (none) |
|  | printf("I am done waiting for PID[%i] with status %i\n",pid, status); |
|  | //-- |
|  |  |
|  | //execl("./hello","hello",'\0'); // replace this process with another |
|  |  |
|  | printf(" ...we never print this line2...\n"); |
|  | } |
|  |  |
|  | } |
|  | return(0); |
|  | } // end main |

# Appendix F: Multi\_Process\_Cons.c

/\*\*\*

forker.c R.Tervo Jan 2013

-------------------------------------

Exploring fork() and exec()

This program monitors PID and PPID in

while spawning a second program.

-------------------------------------

\*\*\*/

**#include** <unistd.h>

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <string.h>

**#include** <pthread.h>

**#include** <sys/neutrino.h>

**#define** MSGSIZE 80

**int** chid;

**int** rcvid;

**unsigned** flags;

**char** send\_message[MSGSIZE]; //send message

**char** receive\_message[MSGSIZE]; //receive message

**char** reader[MSGSIZE]; //

**char** readerreply[MSGSIZE];

**main**( )

{

pid\_t pidA, ppidA; // define PID storage

pid\_t pidB, ppidB;

pid\_t pidC, pid;

pid\_t pidD, pidE;

pid\_t ppidE;

**int** status; // status of wait()

**puts**("Embedded Message Passing Example Program");

**puts**("creating channel");

flags = NULL;

chid = **ChannelCreate**(flags);

**if** (chid==1){

**printf**("could not create channel \n");

//exit(\_1);

}

pidA = **getpid**(); // establish main program PID

ppidA = **getppid**();

**printf**("PID=%i, PPID=%i I am the main program.\n", pidA, ppidA);

pid = **wait**(&status); // wait for child process to end (none)

**printf**("I am done waiting for PID[%i] with status %i\n",pid, status);

**printf**("Fork now..\n");

**fflush**(stdout); // complete any pending prints

pidC = **fork**(); // returns child PID (or 0 if child)

pidB = **getpid**();

ppidB = **getppid**();

**if**(pidA != pidB)

{

**printf**("PID=%i, PPID=%i I am the child. \n", pidB, ppidB);

**printf**("Exec now..\n");

**fflush**(stdout); // complete any pending prints

**int** ucoid;

**puts**("user has been called");

**puts**("user connecting to channel. \n");

ucoid = **ConnectAttach**(0, pidA, chid, 0, NULL);

**while**(1) {

**puts**("user:go ahead, enter a string");

**gets**(send\_message);

**printf**("user:waiting to send: = %s =.\n",send\_message);

**MsgSend**(ucoid, &send\_message, **sizeof**(send\_message), &receive\_message, **sizeof**(receive\_message));

**printf**("user: received a reply from friend: = %s =\n", receive\_message);

**sleep**(1);

}

}

**else**

{

pidD= **fork**();

pidE=**getpid**();

ppidE=**getppid**();

**if**(pidA != pidE){

**printf**("PID=%i, PPID=%i I am the parent. \n", pidA, ppidA);

**printf**("Exec now..\n");

**fflush**(stdout); // complete any pending prints

**int** conscoid;

**puts**("friend has been called");

**puts**("friend connecting to channel. \n");

conscoid = **ConnectAttach**(0, pidA, chid, 0, NULL);

**strcpy**(readerreply, "Got it!");

**while**(1) {

**printf**("friend: I am waiting to receive message from user.\n");

rcvid = **MsgReceive**(chid, &reader, **sizeof**(reader), NULL);

**printf**("friend: received = %s = from user.\n", reader);

**MsgReply**(rcvid, NULL, &readerreply, **sizeof**(readerreply));

**sleep**(1);

**printf**("PID=%i, PPID=%i I am the parent. \n", pidB, ppidB);

}

}

**else**{

//--

**printf**("PID=%i, PPID=%i I am the second child. \n", pidE, ppidE);

**int** fcoid;

**puts**("friend has been called");

**puts**("friend connecting to channel. \n");

fcoid = **ConnectAttach**(0, pidA, chid, 0, NULL);

**strcpy**(readerreply, "Got it!");

**while**(1) {

**printf**("friend: I am waiting to receive message from user.\n");

rcvid = **MsgReceive**(chid, &reader, **sizeof**(reader), NULL);

**printf**("friend: received = %s = from user.\n", reader);

**MsgReply**(rcvid, NULL, &readerreply, **sizeof**(readerreply));

**sleep**(1);

**printf**("PID=%i, PPID=%i I am the parent. \n", pidB, ppidB);

}

pid = **wait**(&status); // wait for child process to end (none)

**printf**("I am done waiting for PID[%i] with status %i\n",pid, status);

//--

//execl("./hello","hello",'\0'); // replace this process with another

**printf**(" ...we never print this line2...\n");

}

}

**return**(0);

} // end main