Operating Systems Lab

CPE 435-01

Lab 05: Messaging in Linux

By: David Thornton

Lab Date: 8 February 2021

Lab Due: 17 February 2021

Demonstration Due: 17 February 2021

Introduction

The purpose of this lab is to give students an introduction to message queues in Linux.

Theory

Topic 1: msgget()

- “The msgget() system call returns the System V message queue identifier associated with the value of the key argument. It may be used either to obtain the identifier of a previously created message queue (when msgflg is zero and key does not have the value IPC_PRIVATE), or to create a new set.” - Source

Topic 2: msgsnd()

- “The msgsnd() and msgrcv() system calls are used, respectively, to send messages to, and receive messages from, a message queue. The calling process must have write permission on the message queue in order to send a message, and read permission to receive a message.” - Source

Topic 3: msgrcv()

- “The msgsnd() and msgrcv() system calls are used, respectively, to send messages to, and receive messages from, a message queue. The calling process must have write permission on the message queue in order to send a message, and read permission to receive a message.” - Source

Topic 4: msgctl()

- “msgctl() performs the control operation specified by cmd on the System V message queue with identifier msqid.” - Source. In this lab, msgctl is used to delete the message queue.

Lab Assignment

You will write two applications (named Process A and Process B). Process A should create a message queue and write a message to the queue (read from user through stdin). Process Should wait for data to be received from Process B. Process B should send some message toProcess A using the message queue. The message sent by Process B should be user typed string through stdin. Process A should respond to Process B with another message sent through the message queue. This is also the message typed by the user in Process A's stdin. Keep the process of sending messages back and forth until one types "Exit". The received messages should be displayed to the terminal by both process A and B.Please make sure to delete the message queue after you are done.

Observations

All code performs as expected.

Figure 1: Normal execution of processA.cpp

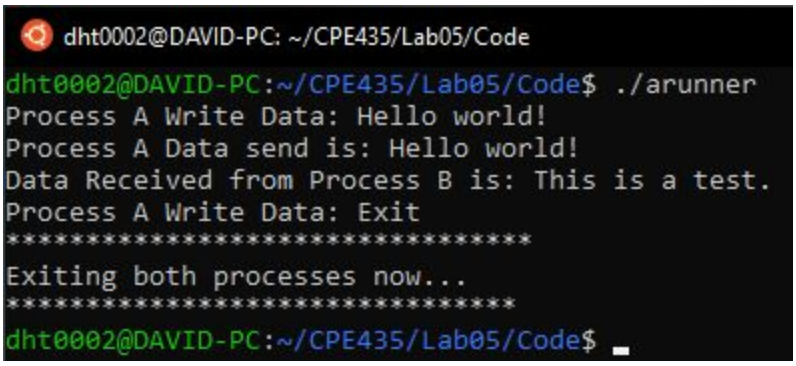
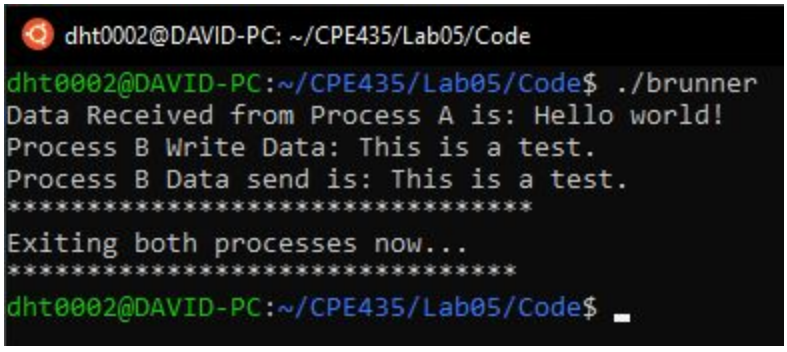


Figure 2: Normal execution of processB.cpp



Lab Questions

1. How do you make a process wait to receive a message and not return immediately?
    i. In the struct, declare a variable long mesg_type that can be set to handle message typing.

2. Message Queue vs Shared Memory (discuss use and differences).
    i.

3. Research use of function ftok(), what is its use?
    i. “The ftok() function uses the identity of the file named by the given pathname (which must refer to an existing, accessible file) and the least significant 8 bits of proj_id (which must be nonzero) to generate a key_t type System V IPC key, suitable for use with msgget(2), semget(2), or shmget(2). The resulting value is the same for all pathnames that name the same file, when the same value of proj_id is used. The value returned should be different when the (simultaneously existing) files or the project IDs differ.” - Source

4. What does IPC_NOWAIT do?
    i. This flag removes the check from the msgsnd()/msgrcv() command.
    ii. “If **IPC_NOWAIT** is passed as a flag, and no messages are available, the call returns ENOMSG to the calling process. Otherwise, the calling process blocks until a message arrives in the queue that satisfies the msgrcv() parameters. If the queue is deleted while a client is waiting on a message, EIDRM is returned. EINTR is returned if a signal is caught while the process is in the middle of blocking, and waiting for a message to arrive.” - Source

Conclusion

This lab was successful in introducing me to the concepts involved with process communication.
Demo link

Appendix

Appendix 1: processA.c

```
// ****************************************
// Program Title: Lab 05
```

```c
// Project File: processA.c
// Name: David Thornton
// Course Section: CPE-435, SP 2021
// Due Date: 02/17/2021
// ***************************************
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdlib.h>
#include "header.h"

void exitProcess(int msgid);

int main()
{
	struct text_message message;
	int msgid = msgget(key, 0666 | IPC_CREAT);
	while(1) // Infinite loop
	{
		printf("Process A Write Data: ");
		fgets(message.mesg_text, MAX, stdin);
		message.mesg_type = 1;
		msgsnd(msgid, &message, sizeof(message), 0);
		if(strcmp(message.mesg_text, "Exit\n") == 0)
		{
			exitProcess(msgid);
		}
		printf("Process A Data send is: %s", message.mesg_text);

		msgrcv(msgid, &message, sizeof(message), 2, 0);
		if(strcmp(message.mesg_text, "Exit\n") == 0)
		{
			exitProcess(msgid);
		}
		printf("Data Received from Process B is: %s",
message.mesg_text);
	}
	msgctl(msgid, IPC_RMID, 0);
	return 0;
}
```

```
void exitProcess(int msgid)
{
	printf("********************************");
	printf("\nExiting both processes now...\n");
	printf("********************************\n");
	msgctl(msgid, IPC_RMID, 0);
	exit(0);
}
```

Appendix 2: processB.c

```
// ****************************************
// Program Title: Lab 05
// Project File: processB.c
// Name: David Thornton
// Course Section: CPE-435, SP 2021
// Due Date: 02/17/2021
// ****************************************
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdlib.h>
#include "header.h"

void exitProcess(int msgid);

int main()
{
	struct text_message message;
	int msgid = msgget(key, 0666 | IPC_CREAT);
	while(1) // Infinite loop
	{
		msgrcv(msgid, &message, sizeof(message), 1, 0);
		if(strcmp(message.mesg_text, "Exit\n") == 0)
		{
			exitProcess(msgid);
		}
		printf("Data Received from Process A is: %s",
```

```
message.mesg_text);

            printf("Process B Write Data: ");
            fgets(message.mesg_text, MAX, stdin);
            message.mesg_type = 2;
            msgsnd(msgid, &message, sizeof(message), 0);
            if(strcmp(message.mesg_text, "Exit\n") == 0)
            {
                  exitProcess(msgid);
            }
            printf("Process B Data send is: %s", message.mesg_text);
      }
      msgctl(msgid, IPC_RMID, 0);
      return 0;
}

void exitProcess(int msgid)
{
      printf("*********************************");
      printf("\nExiting both processes now...\n");
      printf("*********************************\n");
      msgctl(msgid, IPC_RMID, 0);
      exit(0);
}
```

Appendix 3: header.h

```
// ****************************************
// Program Title: Lab 05
// Project File: header.h
// Name: David Thornton
// Course Section: CPE-435, SP 2021
// Due Date: 02/17/2021
// ****************************************
#define key ((key_t)(1234))
#define MAX 100

struct text_message {
      long mesg_type;
      char mesg_text[MAX];
};
```