Brian Duenas
CSE 460

Lab5 - Study of Interprocess Communication (IPC) and XV6

Part I - Message Queues

```
msgctl(int msqid, int cmd, struct msqid_ds *buf)
```
msgctl() performs the control operation specified by cmd on the System V message queue with identifier msqid.

```
msgget(key_t key, int msgflg)
```
msgget() returns value will be the message queue identifier

```
msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg)
```
msgrcv() receive messages from, a System V message queue.

```
msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg)
```
msgsnd() sends messages to a System V message queue.

Modified program so that each program can both receive and send messages alternatively:
What I did was combine code from each file and ran a receiver and sender simultaneously using a *fork*.
highlights = code added

```
//msg1.cpp
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>

#include <signal.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
using namespace std;
#define MAX_TEXT 512

struct my_msg_st {
	long int my_msg_type;
	char some_text[BUFSIZ];
};
struct my_msg {
	long int my_msg_type;
	char some_text[MAX_TEXT];
};

int main()
{
	int running = 1;
	int msgid;
	struct my_msg_st some_data;
	long int msg_to_receive = 0;

	/* First, we set up the message queue. */

	int sar = fork();  //fork for sending and recieving
	if (sar == 0) {
		msgid = msgget((key_t)1234, 0666 | IPC_CREAT);

		if (msgid == -1) {
			fprintf(stderr, "msgget failed with error: %d\n", errno);
			exit(EXIT_FAILURE);
		}
	}

	while (running && sar == 0) {
		if (msgrcv(msgid, (void *)&some_data, BUFSIZ,
			msg_to_receive, 0) == -1) {
			fprintf(stderr, "msgrcv failed with error: %d\n", errno);
			exit(EXIT_FAILURE);
		}
		printf("\nThem: %s", some_data.some_text);
		if (strncmp(some_data.some_text, "end", 3) == 0) {
			running = 0;
			kill(sar, SIGKILL);
		}
	}

	if (sar == 0) {
		if (msgctl(msgid, IPC_RMID, 0) == -1 && sar == 0) {
			fprintf(stderr, "msgctl(IPC_RMID) failed\n");
```

```
        exit(EXIT_FAILURE);
                }
        }

        if (sar != 0) {
                struct my_msg some_data2;
                int msgid2;
                char buffer[BUFSIZ];

                msgid2 = msgget((key_t)7777, 0777 | IPC_CREAT);

                if (msgid2 == -1) {
                        fprintf(stderr, "msgget failed with error: %d\n", errno);
                        exit(EXIT_FAILURE);
                }

                while (running) {
                        printf("\nEnter some text: ");
                        fgets(buffer, BUFSIZ, stdin);
                        some_data2.my_msg_type = 1;
                        strcpy(some_data2.some_text, buffer);

                        if (msgsnd(msgid2, (void *)&some_data2, MAX_TEXT, 0) == -1) {
                                fprintf(stderr, "msgsnd failed\n");
                                exit(EXIT_FAILURE);
                                kill(sar, SIGKILL);
                        }
                        if (strncmp(buffer, "end", 3) == 0) {
                                running = 0;
                                exit(EXIT_SUCCESS);
                        }
                }
        }

        exit(EXIT_SUCCESS);
}
```

```cpp
//msg2.cpp
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <iostream>

#include <signal.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
using namespace std;
#define MAX_TEXT 512

struct my_msg_st {
	long int my_msg_type;
	char some_text[MAX_TEXT];
};

struct my_msg_recieved {
	long int my_msg_type;
	char some_text[BUFSIZ];
};

int main()
{
	int running = 1;
	struct my_msg_st some_data;
	int msgid, msgid2;
	char buffer[BUFSIZ];

	int sar = fork();  //fork for sending and recieving
	if (sar != 0) {
		msgid = msgget((key_t)1234, 0666 | IPC_CREAT);

		if (msgid == -1) {
			fprintf(stderr, "msgget failed with error: %d\n", errno);
			exit(EXIT_FAILURE);
		}
	}

	while (running && sar != 0) {  //parent fork sends
		printf("\nEnter some text: ");
		fgets(buffer, BUFSIZ, stdin);
		some_data.my_msg_type = 1;
		strcpy(some_data.some_text, buffer);

		if (msgsnd(msgid, (void *)&some_data, MAX_TEXT, 0) == -1) {
			fprintf(stderr, "msgsnd failed\n");
			kill(sar, SIGKILL);
		}
		if (strncmp(buffer, "end", 3) == 0) {
			running = 0;
		}
	}

	struct my_msg_recieved some_data2;
```

```
long int msg_to_recieve = 0;

        if (sar == 0) {
                msgid2 = msgget((key_t)7777, 0777 | IPC_CREAT);

                if (msgid2 == -1) {
                        fprintf(stderr, "msgget failed with error: %d\n", errno);
                        exit(EXIT_FAILURE);
                }


                while (running) {
                        if (msgrcv(msgid2, (void *)&some_data2, BUFSIZ,
                                msg_to_recieve, 0) == -1) {
                                fprintf(stderr, "msgrcv failed with error: %d\n", errno);
                                exit(EXIT_FAILURE);
                        }
                        printf("\nThem: %s", some_data2.some_text);
                        if (strncmp(some_data2.some_text, "end", 3) == 0) {
                                running = 0;
                                kill(sar, SIGKILL);
                        }
                }

                if (msgctl(msgid2, IPC_RMID, 0) == -1) {
                        fprintf(stderr, "msgctl(IPC_RMID) failed\n");
                        exit(EXIT_FAILURE);
                }
        }

        exit(EXIT_SUCCESS);
}
```

Output for *msg1*:

```
[         }@csusb.edu@jb359-1 lab5]$ msg1
Enter some text: hey
Enter some text: whats up
Enter some text:
Them: not much

Them: gtg later!

Them: end
Killed
[_________}@csusb.edu@jb359-1 lab5]$
```

Output for *msg2*:

```
[         }@csusb.edu@jb359-1 lab5]$ msg2
Enter some text:
Them: hey
Them: whats up
not much
Enter some text: gtg later!
Enter some text: end
[_________}@csusb.edu@jb359-1 lab5]$
```

Part II - IPC Status Commands

```
[          @csusb.edu@jbh3-1 lab5]$ ipcs -s

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

[          @csusb.edu@jbh3-1 lab5]$ ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status

[          @csusb.edu@jbh3-1 lab5]$ ipcs -q

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages
0x00001e61 32768      005029683@ 777        0            0
0x00000929 65537      005029683@ 666        0            0

[          @csusb.edu@jbh3-1 lab5]$
```

The first two commands *ipcs –s* and *ipcs –m* don't show outputs. The command *ipcs –q* shows us that there is message queues present.

Part III - Study of XV6

GDB terminal:

```
(gdb) target remote :27050
Remote debugging using :27050
warning: Remote gdbserver does not support determining executable automatically.
RHEL <=6.8 and <=7.2 versions of gdbserver do not support such automatic executa
ble detection.
The following versions of gdbserver support it:
- Upstream version of gdbserver (unsupported) 7.10 or later
- Red Hat Developer Toolset (DTS) version of gdbserver from DTS 4.0 or later (on
ly on x86_64)
- RHEL-7.3 versions of gdbserver (on any architecture)
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x0000fff0 in ?? ()
(gdb) file kerenl
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
kerenl: No such file or directory.
(gdb) file kernel
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from kernel...done.
(gdb) break swtch
Breakpoint 1 at 0x8010469b: file swtch.S, line 11.
(gdb) continue
Continuing.

Thread 1 hit Breakpoint 1, swtch () at swtch.S:11
11        movl 4(%esp), %eax
(gdb) step
12        movl 8(%esp), %edx
(gdb) step
15        pushl %ebp
(gdb) step
swtch () at swtch.S:16
16        pushl %ebx
(gdb) step
swtch () at swtch.S:17
17        pushl %esi
(gdb) step
swtch () at swtch.S:18
18        pushl %edi
(gdb) step
swtch () at swtch.S:21
21        movl %esp, (%eax)
(gdb) step
22        movl %edx, %esp
(gdb) continue
Continuing.

Thread 1 hit Breakpoint 1, swtch () at swtch.S:11
11        movl 4(%esp), %eax
(gdb) clear
Deleted breakpoint 1
```

```
(gdb) break exec
Breakpoint 2 at 0x80100a10: file exec.c, line 12.
(gdb) continue
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 2, exec (path=0x1c "/init", argv=0x8dfffed0)
    at exec.c:12
12      {
(gdb) continue
Continuing.

Thread 2 hit Breakpoint 2, exec (path=0x816 "sh", argv=0x8dffeed0) at exec.c:12
12      {
(gdb) continue
Continuing.

Thread 2 hit Breakpoint 2, exec (path=0x1880 "ls", argv=0x8dfbeed0)
    at exec.c:12
12      {
(gdb) continue
Continuing.

Thread 2 hit Breakpoint 2, exec (path=0x1880 "ls", argv=0x8dee3ed0)
    at exec.c:12
12      {
(gdb)
```

qemu-nox terminal:

```
qemu-system-i386 -nographic -drive file=fs.img,index=1,media=disk,format=raw -drive file=
xv6.img,index=0,media=disk,format=raw -smp 2 -m 512  -S -gdb tcp::27050
xv6...
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
ls

$ ls -l
slls


asls
.              1 1 512
..             1 1 512
README         2 2 2290
cat            2 3 13680
echo           2 4 12688
forktest       2 5 8124
grep           2 6 15556
init           2 7 13276
kill           2 8 12740
ln             2 9 12644
ls             2 10 14828
mkdir          2 11 12820
rm             2 12 12804
sh             2 13 23288
stressfs       2 14 13468
usertests      2 15 56404
wc             2 16 14220
cp             2 17 13424
zombie         2 18 12468
console        3 19 0
▒L$▒▒▒▒q▒U▒▒W 2 23 2290
$ $
```

GDB terminal for scheduler in *proc.c:*

```
(gdb) break scheduler
Breakpoint 1 at 0x80103ab0: file proc.c, line 324.
(gdb) continue
Continuing.
[Switching to Thread 2]

Thread 2 hit Breakpoint 1, scheduler () at proc.c:324
324     {
(gdb) continue
Continuing.
[Switching to Thread 1]

Thread 1 hit Breakpoint 1, scheduler () at proc.c:324
324     {
(gdb) continue
Continuing.

Remote connection closed
(gdb)
The program is not being run.
(gdb)
```

qemu-nox terminal for scheduler in *proc.c*:

```
cpu1: starting 1
cpu0: starting 0
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
init: starting sh
$ ls -l
ls: cannot open -l
$ ls
.              1 1 512
..             1 1 512
README         2 2 2290
cat            2 3 13680
echo           2 4 12688
forktest       2 5 8124
grep           2 6 15556
init           2 7 13276
kill           2 8 12740
ln             2 9 12644
ls             2 10 14828
mkdir          2 11 12820
rm             2 12 12804
sh             2 13 23288
stressfs       2 14 13468
usertests      2 15 56404
wc             2 16 14220
cp             2 17 13424
zombie         2 18 12468
console        3 19 0
▒L$▒▒▒▒q▒U▒▒W 2 23 2290
$ QEMU: Terminated
```

Fully completed 20/20