George Suarez

CSE 460

Lab 5 - Study of Interprocess Communication (IPC) and XV6

1. Message Queues

   msgctl() – Performs the control operation specified by a command on the message queue with an identifier *msgid.*

   msgget() – Returns the message queue identifier associated with the value *key* argument.

   msgrcv() – Receives messages from the message queue

   msgsnd() – Sends messages to the message queue

Output of *msg1.cpp & msg2.cpp*

```
georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 5 on master*
$ ./msg1
You wrote: Hello
You wrote: From The Other Side
You wrote: end

georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 5 on master*
$ ./msg2
Enter some text: Hello
Enter some text: From The Other Side
Enter some text: end
```

msg1.cpp (modified)

```
//msg1.cpp
/* Here's the receiver program. */
```

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

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MAX_TEXT 512

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

int main()
{
    int running = 1;
    struct my_msg_st some_data, send_data;
    long int msg_to_receive = 1;
    char buffer[BUFSIZ];



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

    int msgid1 = msgget((key_t)1234, 0666 | IPC_CREAT);
    int msgid2 = msgget((key_t)2345, 0666 | IPC_CREAT);

    if (msgid1 == -1 || msgid2 == -1)
    {
        fprintf(stderr, "msgget failed with error: %d\n", errno);
        exit(EXIT_FAILURE);
    }
    /* Then the messages are retrieved from the queue, until an end message is encountered.
```

Lastly, the message queue is deleted. */

```
  while (running)
  {
     printf("\nWaiting...\n");
     if (msgrcv(msgid1, (void *)&some_data, BUFSIZ, msg_to_receive, 0) == -1)
     {
        fprintf(stderr, "msgrcv failed with error: %d\n", errno);
        exit(EXIT_FAILURE);
     }

     printf("You wrote: %s", some_data.some_text);

     if (strncmp(some_data.some_text, "end", 3) == 0)
     {
        running = 0;
     }
     else
     {
              printf("Enter some text: ");
                   fgets(buffer, BUFSIZ, stdin);
                   send_data.my_msg_type = 1;
                   strcpy(send_data.some_text, buffer);

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

             if (msgctl(msgid1, IPC_RMID, 0) == -1)
```

```
        {
            fprintf(stderr, "msgctl(IPC_RMID) failed\n");
            exit(EXIT_FAILURE);
        }

        exit(EXIT_SUCCESS);
    }
```

msg2.cpp (modified)

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

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MAX_TEXT 512

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

int main()
{
    int running = 1;
    struct my_msg_st some_data, send_data;
    long int msg_to_recieve = 1;
    char buffer[BUFSIZ];

    send_data.my_msg_type = 1;
```

```
int msgid1 = msgget((key_t)2345, 0666 | IPC_CREAT); // Recieve
int msgid2 = msgget((key_t)1234, 0666 | IPC_CREAT);   // Send

if (msgid1 == -1 || msgid2 == -1)
{
   fprintf(stderr, "msgget failed with error: %d\n", errno);
   exit(EXIT_FAILURE);
}

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

   if (msgsnd(msgid2, (void *)&send_data, MAX_TEXT, 0) == -1)
   {
      fprintf(stderr, "msgsnd failed\n");
      exit(EXIT_FAILURE);
   }
   if (strncmp(buffer, "end", 3) == 0)
   {
      running = 0;
   }
   else
          {
                printf("\nWaiting...\n");
                      if (msgrcv(msgid1, (void *)&some_data, BUFSIZ, msg_to_recieve, 0) == -1)
                      {
                         fprintf(stderr, "msgrcv failed with error: %d\n", errno);
                         exit(EXIT_FAILURE);
                      }
                      printf("\nYou wrote: %s", some_data.some_text);
                      if (strncmp(some_data.some_text, "end", 3) == 0)
                      {
```

```
            running = 0;
        }
    }
}

if (msgctl(msgid1, IPC_RMID, 0) == -1)
{
    fprintf(stderr, "msgctl(IPC_RMID) failed\n");
    exit(EXIT_FAILURE);
}

exit(EXIT_SUCCESS);
}
```

Output of the modified *msg1.cpp & msg2.cpp*

```
georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 5 on master*
$ ./msg2
Enter some text: Hello. This is from msg2

Waiting...

You wrote: Hello from msg1
Enter some text: end
msgsnd failed


georgesuarez at MacBook-Pro in ~/University/CSE-460/Labs/Lab 5 on master*
$ ./msg1

Waiting...
You wrote: Hello
Enter some text: Hello from msg1

Waiting...
You wrote: Hello. This is from msg2
```

```
Enter some text: end
```

2. IPC Status Commands

   ipcs – It is a utility that provides information on System V interprocess communication (IPC) facilities.

   ipcrm – Removes the specified message queues, semaphore sets, and shared memory segments.

   Outputs of the **ipcs** commands

```
[006098556@csusb.edu@csevnc ~]$ ipcs -s

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

- Displays all the semaphore arrays of the running system interprocess.

```
[006098556@csusb.edu@csevnc ~]$ ipcs -m

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 360448     005512737@ 600        16777216   2          dest
0x00000000 458753     005512737@ 600        4194304    2          dest
0x00000000 393218     005512737@ 600        524288     2          dest
0x00000000 425987     005512737@ 600        524288     2          dest
0x00000000 13139974   005512737@ 600        524288     2          dest
0x00000000 21757959   005512737@ 600        524288     2          dest
0x00000000 23789577   005512737@ 600        524288     2          dest
```

- Displays all the shared memory segments of the running system interprocess.

```
[006098556@csusb.edu@csevnc ~]$ ipcs -q

------ Message Queues --------
        key      msqid     owner     perms     used-bytes   messages
```

- Displays all the message queues in the running system interprocess.

3. Study of XV6
    - Here shows loading the kernel into the debugger and putting a break point at function called *swtch* to examine how the context switching is done in **xv6.**

```
(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) ste
Ambiguous command "ste": step, stepi, stepping.
(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) step
swtch () at swtch.S:25
25   popl %edi
(gdb) step
swtch () at swtch.S:26
26   popl %esi
(gdb) step
swtch () at swtch.S:27
27   popl %ebx
(gdb) step
swtch () at swtch.S:28
28   popl %ebp
(gdb) step
swtch () at swtch.S:29
29           ret
```

- Here is showing putting a breakpoint at the *exec* function to show what is being executed when entering in a command in **xv6** which in this case it is *ls -l*.

```
(gdb) continue
Continuing.

Thread 2 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.

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.
[Switching to Thread 1]

Thread 1 hit Breakpoint 2, exec (path=0x1880 "ls", argv=0x8dfbeed0) at exec.c:12
12 {
(gdb) print argv[0]
$1 = 0x1880 "ls"
(gdb) print argv[1]
$2 = 0x1883 "-l"
```

- Here is showing the backtrace of the *exec* function which shows the calls that the function makes to the system.

```
(gdb) backtrace
#0  exec (path=0x1880 "ls", argv=0x8dfbeed0) at exec.c:12
#1  0x801053a0 in sys_exec () at sysfile.c:420
#2  0x80104879 in syscall () at syscall.c:139
#3  0x80105835 in trap (tf=0x8dfbefb4) at trap.c:43
#4  0x8010564f in alltraps () at trapasm.S:20
#5  0x8dfbefb4 in ?? ()
```

- This shows the code where the *exec* function is being called from in the **xv6**.

```
Thread 1 hit Breakpoint 2, exec (path=0x1880 "ls", argv=0x8df23ed0) at exec.c:12
12 {
(gdb) up
#1  0x801053a0 in sys_exec () at sysfile.c:420
420         return exec(path, argv);
(gdb) list
415           break;
416         }
417         if(fetchstr(uarg, &argv[i]) < 0)
418           return -1;
419       }
420       return exec(path, argv);
421     }
422
423     int
424     sys_pipe(void)
(gdb)
```

Examining *proc.c*

- Here is showing the function *scheduler* in the file *proc.c* which is a function that called in *main.c* file which starts scheduling the processes that are running in **xv6**.

```
(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) up
#1  0x80102e8f in mpmain () at main.c:57
57   scheduler();     // start running processes
(gdb) list
52 mpmain(void)
53 {
54   cprintf("cpu%d: starting %d\n", cpuid(), cpuid());
55   idtinit();       // load idt register
56   xchg(&(mycpu()->started), 1); // tell startothers() we're up
57   scheduler();     // start running processes
58 }
59
60 pde_t entrypgdir[];  // For entry.S
61
```

Discussion: I have successfully done each part in this lab. I should get **20 points**.