Logan Woolery && Jack Potter
CprE 308: Section L
Project 02
4/8/2019

IPCS

Unnamed Pipe

1pt) Output of pipe_test.c

My child asked "Are you my mummy?"
And then returned 42

1pt) What do you notice about the timing of the printing?

It waits for a second or two, then prints both lines at once

2pt) What happens when more than one process tries to write to a pipe at the same time? Be specific: using the number of bytes that each might be trying to write and how that effects what happens.

The pipe will accept data upto the pipe capacity. If the data from the second process does not exceed the capacity, it will write. If the data does exceed the capacity, write will block until the pipe has been read from.

1pt) How does the output of pipe_test.c change if you move the sleep statement from the child process before the fgets of the parent?

No apparent change

1pt) What is the maximum size of a pipe in linux since kernel 2.6.11?

65536 bytes.

Named Pipe (FIFO)

1pt) What happens when you run the echo command?

The echo command sends the string “hello fifo” into the FIFO, which causes the cat command in the second terminal to output “hello fifo” before exiting successfully. Before running echo, the cat command was hanging and not doing anything.

1pt) What happens when you run the echo first and then the cat?

The echo command hangs until cat runs. Running cat causes the string “hello fifo” to be output and the command exits successfully. The echo command also exits successfully as soon as cat is run.

2pt) Look at the man page fifo(7). Where is the data that is sent through the FIFO stored?

It is kept internally within the kernel, never being written to the filesystem. It exists only in volatile memory.

2pt) Write a short program that uses named FIFO (mkfifo(3)) to print any line entered into the program on one terminal out on the other terminal.
Codes here

Socket

2pt) What are the six types of sockets?
SOCK_STREAM
SOCL_DGRAM
SOCK_SEQPACKET
SOCK_RAW
SOCK_RDM
SOCK_PACKET
1pt) What are the two domains that can be used for local communications?
AF_UNIX
AF_LOCAL

Message Queues

1pt) What is the output of mq_test1?
Received message "I am Clara"

1pt) What is the output of mq_test2?
Received message "I am the Doctor"
Received message "I am the Master"
2pt) Change mq_test2.c to send a second message which reads "I am X" where "X" is your favorite companion. Change mq_test1.c to wait for and print this second message before exiting.
woolery@co2048-14:~/308/cpre_308_proj2/Project2/ipc-types$ ./mq_test1
Received message "I am Clara"
Received message "I am Pond"

Shared Memory Space

1pt) What is the output if you run both at the same time calling shm_test1 first?
woolery@co2048-14:~/308/cpre_308_proj2/Project2/ipc-types$
./shm_test1.o && ./shm_test2.o
a_string = ""
an_array[] = {0, 1, 4, 9, 16}
a_ptr = 140721150294384 = "I am a string allocated on main's stack!"

1pt) What is the output if you run both at the same time calling shm_test2 first?

```
woolery@co2048-14:~/308/cpre_308_proj2/Project2/ipc-types$
./shm_test2.o && ./shm_test1.o
a_string = "I am a buffer in the shared memory area"
an_array[] = {42, 1, 4, 9, 16}
Segmentation fault
```

1pt) What if you run each by themselves?

```
woolery@co2048-14:~/308/cpre_308_proj2/Project2/ipc-types$
./shm_test1.o
a_string = "I am a buffer in the shared memory area"
an_array[] = {0, 1, 4, 9, 16}
a_ptr = 140721211721936 = "I am a string allocated on main's stack!"
woolery@co2048-14:~/308/cpre_308_proj2/Project2/ipc-types$
./shm_test2.o
a_string = "I am a buffer in the shared memory area"
an_array[] = {42, 1, 4, 9, 16}
Segmentation fault
woolery@co2048-14:~/308/cpre_308_proj2/Project2/ipc-types$
```

2pt) Why is shm_test2 causing a segfault? How could this be fixed?

Not properly dereferencing the pointer

1pt) What happens if the two applications both try to read and set a variable at the same time?

Undefined behaviour

Fix: `printf("a_ptr = %lu = \"%s\"\n", &(shared_mem->a_ptr), shared_mem->a_ptr);`

1pt) How can a shared memory space be deleted from the system?

shm_unlink

2pt) Change the code to share some useful piece of information?

Code on the following page

SHM_TEST1.c

```
int main(int argc, char** argv)
{
	int fd;
	fd = shm_open("/308LabIPC", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR |
S_IRGRP); // open the shared memory area creating it if it doesn't exist
	if(!fd)
	{
		perror("shm_open\n");
		return -1;
	}
	if(ftruncate(fd, sizeof(struct SHM_SHARED_MEMORY)))
	{
		perror("ftruncate\n");
		return -1;
	}
	struct SHM_SHARED_MEMORY* shared_mem;
	shared_mem = mmap(NULL, sizeof(struct SHM_SHARED_MEMORY), PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);
	if(!shared_mem)
	{
		perror("mmap\n");
		return -1;
	}
	if(close(fd))
	{
		perror("close\n");
		return -1;
	}
	int i;
	for(i = 0; i < 5; i++)
	{
		shared_mem->an_array[i] = i*i;
	}
	char my_string[] = "I am a string allocated on main's stack!";
	shared_mem->a_ptr = my_string;
	sleep(5);
	printf("a_string = \"%s\"\n", shared_mem->a_string);
	printf("an_array[] = {%d, %d, %d, %d, %d}\n", shared_mem->an_array[0],
shared_mem->an_array[1], shared_mem->an_array[2], shared_mem->an_array[3],
shared_mem->an_array[4]);
	if(shared_mem->a_ptr > 0)
	{
		printf("a_ptr = %lu = \"%s\"\n", shared_mem->a_ptr,
shared_mem->a_ptr);
	}
	else
	{
		printf("a_ptr = NULL\n");
	}
}
```

shm_test2

```
int main(int argc, char** argv)
{
	int fd;
	fd = shm_open("/308LabIPC", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR |
S_IRGRP); // open the shared memory area creating it if it doesn't exist
	if(!fd)
	{
		perror("shm_open\n");
		return -1;
	}
	if(ftruncate(fd, sizeof(struct SHM_SHARED_MEMORY)))
	{
		perror("ftruncate\n");
		return -1;
	}
	struct SHM_SHARED_MEMORY* shared_mem;
	shared_mem = mmap(NULL, sizeof(struct SHM_SHARED_MEMORY), PROT_READ |
PROT_WRITE, MAP_SHARED, fd, 0);
	if(!shared_mem)
	{
		perror("mmap\n");
		return -1;
	}
	if(close(fd))
	{
		perror("close\n");
		return -1;
	}
	strcpy(shared_mem->a_string, "I am a buffer in the shared memory area");
	shared_mem->an_array[0] = 42;
	sleep(5);
	printf("a_string = \"%s\"\n", shared_mem->a_string);
	printf("an_array[] = {%d, %d, %d, %d, %d}\n", shared_mem->an_array[0],
shared_mem->an_array[1], shared_mem->an_array[2], shared_mem->an_array[3],
shared_mem->an_array[4]);
	fflush(stdout);
	if(shared_mem->a_ptr > 0)
	{
		printf("a_ptr = %lu = \"%s\"\n", shared_mem->a_ptr,
shared_mem->a_ptr);
	}
	else
	{
		printf("a_ptr = NULL\n");
	}
}
```

Unnamed Semaphores

2pt) What is the function call that would be needed to create an unnamed semaphore in a shared memory space called shared_mem->my_sem and assign it an intial value of 5?

```
sem_init(&shared_mem->my_sem, 1, 5);
```

Named Semaphores

1pt) How long do semaphores last in the kernel?

Until destroyed

1pt) What causes them to be destroyed?

sem_ublink() or a reboot of the system

2pt) What is the basic process for creating and using named semaphores? (List the functions that would need to be called, and their order).

```
sem_t *mysem;
unsigned int myvalue = 1;
ysem = sem_open("mysem", O_CREAT | O_EXCL, 0644, myvalue);
sem_wait(mysem);
sem_unlink("mysem");
```

Signals

1pt) What happens when you try to use CTRL+C to break out of the infinite loop?

The program outputs “Ah Ah Ah, you didn't say the magic word” and continues the loop.

1pt) What is the signal number that CTRL+C sends?

2 - SIGINT

1pt) When a process forks, does the child still use the same signal handler?

Yes, signals sent to the process group after a fork are delivered to both the parent and the child.

1pt) How about during a exec call?

No, as exec calls don’t return on success, meaning that when they overwrite the stack with their own data they don’t restore the state of the program. Nothing is preserved after an exec call, including the signal handlers.

5pt) Write two programs. One which will send a signal of number 42 to the other process. The other program should catch that signal and print out the message “I got the signal!”

The code can be found on the next page. After compiling both programs, the receiving program is executed. The first line of output

will be the process id of the receiving program. Run the sending program with one argument, the pid of the receiving program. It will send the signal number 42, which will be handled by the receiving program.

receive.c:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>

void sighandler(int);

int main () {
   signal(42, sighandler);
   printf("%d\n", getpid());
   while(1) {
      printf("Going to sleep for a second...\n");
      sleep(1);
   }
   return(0);
}

void sighandler(int signum) {
      if(signum == 42){
      printf("I got the signal!\n");
      exit(1);
      }
}
```

send.c

```
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>

int main(int argc, char * argv[]){
      int i = atoi(argv[1]);
      kill(i, 42);
}
```

Part 4+

Dynamically / Statically Linked Libraries

1pt) First output of lib_test:

Answer here

1pt) Second output of lib_test after exporting the library:

Answer here

Project 2

If you worked with someone else - who was it?

Logan Woolery and Jack Potter

5pt) Summary

Answer here

If you did extra credit - tell us what the functionality and how to use it here:

Answer here

How to run Project 2

Terminal 1: ./src/printer-server/printer

```
rm /drivers/*
make clean
make
./virt-printer -n printer0
./virt-printer -n printer3
```

Terminal 2: ./src/printer-server

```
make clean
make
./main -d
```

Terminal 3: ./src/libprintserver

```
make clean
make
```

Terminal 4: ./src/cli-printer

```
make clean
make
export LD_LIBRARY_PATH="../libprintserver/"
LIST FUNCTUION:
     ./cli-printer -l file_name
PRINT FUNCTION
     ./cli-printer -d driver -s description -o output_name file_name
```

Notes:

Any other details needed to run the program