Sean Hinchee  
CprE 308: Section G  
Project 02  
3/28/18

**IPCS**

**Unnamed Pipe**

**Output of pipe\_test.c**

% ./a.out

My child asked "Are you my mummy?"

And then returned 42

**What do you notice about the timing of the printing?**

Timing had a slight delay, then, both print statements display.

**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.**

POSIX.1 dictates that writes of less than PIPE\_BUF bytes must be atomic. The output data is thusly written to the pipe as a continuous sequence. Writes greater in size than PIPE\_BUF may not be atomic. In the case of nonatomic writes, the kernel may interleave the data written between processes. POSIX.1 dictates that PIPE\_BUF is at least 512 bytes (on linux this is 4096 bytes). Blocking can be disabled or enabled if desired.

**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?**

The output with the sleep statement moved from the child to before the fgets of the parent:

% ./a.out

My child asked "Are you my mummy?"

And then returned 42

The original (unmodified) output:

% ./a.out

My child asked "Are you my mummy?"

And then returned 42

Thusly, there is no change between the outputs (tested on linux-7 server).

**What is the maximum size of a pipe in linux since kernel 2.6.11?**

The size is located in /proc/sys/fs/pipe-max-size and is 1048576 (16 pages).

**Named Pipe (FIFO)**

**What happens when you run the echo command?**

Text is written into the fifo and the write is blocked until something opens the fifo for reading.

**What happens when you run the echo first and then the cat?**

The cat reads and outputs the text being blocked on the echo write, releasing the echo writer.

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

Data sent through the FIFO is stored in the kernel. Said data does not get written to the filesystem.

**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**

**What are the six types of sockets?**

STREAM, DGRAM, SEQPACKET, RAW, RDM, PACKET

**What are the two domains that can be used for local communications?**

AF\_UNIX, AF\_LOCAL

**Message Queues**

**What is the output of mq\_test1?**

% ./mq\_test1

Received message "I am Clara"

**What is the output of mq\_test2?**

% ./mq\_test2

Received message "I am the Doctor"

Received message "I am the Master"

**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.**

% ./mq\_test1

Received message "I am Clara"

Received message "I am Rose"

% ./mq\_test2

Received message "I am the Doctor"

Received message "I am the Master"

**Shared Memory Space**

**What is the output if you run both at the same time calling shm\_test1 first?**

Answer here

**What is the output if you run both at the same time calling shm\_test2 first?**

Answer here

**What if you run each by themselves?**

Answer here

**Why is shm\_test2 causing a segfault? How could this be fixed?**

Answer here

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

Answer here

**How can a shared memory space be deleted from the system?**

Answer here

**Change the code to share some useful piece of information?**

Code here

**Unnamed Semaphores**

**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**

**How long do semaphores last in the kernel?**

POSIX semaphores are kernel persistent, meaning that even if no process has the semaphore open, the value is held.

**What causes them to be destroyed?**

sem\_unlink() is the function by which to fully remove a semaphore (once the reference count is 0) from the kernel. Alternatively, the system could reboot.

**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 \*sem;

unsigned int myvalue = 1;

sem = sem\_open("mysem", O\_CREAT | O\_EXCL, 0644, myvalue);

sem\_wait(sem);

sem\_post(sem);

sem\_unlink("mysem");

**Signals**

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

% ./a.out

.....^?

Ah Ah Ah, you didn't say the magic word

**What is the signal number that CTRL+C sends?**

SIGINT.

**When a process forks, does the child still use the same signal handler?**

If fork() is called without a further exec (or similar), then then the child will use the same signal handler as the parent.

**How about during a exec call?**

If exec is called after fork, the child will use a different signal handler than the original parent.

**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!”**

Code here

**Dynamically / Statically Linked Libraries**

**First output of lib\_test:**

% ./lib\_test

./lib\_test: error while loading shared libraries: libhello.so: cannot open shared object file: No such file or directory

**Second output of lib\_test after exporting the library:**

sh-4.2$ export =`pwd`

sh-4.2$ $LD\_LIBRARY\_PATH

sh: /home/seh/cpre308/proj2/Project2-1/library: Is a directory

sh-4.2$ ./lib\_test

Hello

World

World

World

i=42

**Project 2**

**If you worked with someone else – who was it?**

N/A.

**Summary**

The implementation was designed around a socket interface. The library has the address (127.0.0.1) and the port (13337) hardcoded (as does print-server). Each communication is processed into a generalized two step process: a command message (one of MKJOB or GETDRIVERS) followed by a data message. Data messages are formatted as ~-separated tuples (as a result, ~ is a restricted character in these transmissions).

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

cli-printer supports all proposed options.

**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