Cole Vikupitz

CIS 415

20 April 2017

Assignment 1

Textbook Questions

1. OSC 2.18: *What are the two models of inter-process communication? What are the strengths and weaknesses of the two approaches?*

The two models are **message-passing** and **shared memory**.

For message-passing, the communicating processes exchange messages through a common mailbox. Therefore the strength of this model is that the messages can be exchanged directly or indirectly between processes which means no conflicts need to be avoided. It is also easier to implement. The weakness of this model is its cost in speed and convenience of communication.

For shared memory, processes use system calls to create and gain access to memory regions used or owned by other processes. The advantage of this model is that a mailbox is not needed for communication, therefore maximizes the performance of communication. The weakness of this model is that direct access of memory used by other processes can cause problems with security and synchronization, ignoring the producer-consumer problem.

2. OSC 2.19: *Why is the separation of mechanism and policy desirable?*

This separation is desirable because it's important for maintaining a flexible operating system. Policies determine what will be done, and mechanisms determine how they will be done. Any desirable changes we wish to make on our operating system requires changes in policies. By keeping this separation, changes we make to our policies shouldn't result in massive amounts of changes made to our mechanisms. Since policies are likely to change over time, mechanisms that are insensitive to these changes are desirable.

3. OSC 3.9: *Describe the actions taken by a kernel to context-switch between processes?*

To context switch between processes, the kernel must perform a state save of the current running process, then performs a state restore on another process to resume operations. When saving the current context of the currently running process, the kernel is saving the process control block of the process which includes its state, counter, CPU register usage, scheduling information, memory-management information, etc. After saving this, the kernel then loads the save state of the next process scheduled to run, which is also saved in the form of a process control block.

4. OSC 3.18: *What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.*

a. *Synchronous and asynchronous communication*

Synchronous and asynchronous communication are also referred to as blocking and non-blocking, respectively. For blocking, we use blocking send and receive system calls, which gives us a rendezvous between the sender and receiver. The advantage of synchronous communication is that it solves the producer-consumer problem (synchronization between

processes). The disadvantage of this is that blocking may not always be necessary. The opposite is true for asynchronous communication: its advantage is not blocking messages, but its disadvantage is that it ignores the producer-consumer problem.

b. *Automatic and explicit buffering*

The advantage of using automatic buffering is that we have a choice between a bounded or unbounded queue, which both have their own advantages and disadvantages. In both cases, the advantage of automatic buffering is not always needing to block messages. The disadvantage is that there are synchronization issues with non-blocking messages. The advantage of using explicit buffering is that we use a queue with a capacity of zero, meaning that we only receive one message at a time, minimizing memory and synchronization issues. The disadvantage of this is that the sender always blocks the message.

c. *Send by copy and send by reference*

The advantage of send by copy is that the user or system can modify the local copy of the data as desired without modifying the original data. The disadvantage of this is that to keep a local copy means that it requires more memory from the system. Send by reference has the advantage of not requiring more memory for copies, but has the disadvantage of the programmer or system modifying, or potentially destroying, the original data.

d. *Fixed-sizes and variable-sized messages*

Messages sent between processes may either be fixed or variable in size. The advantage of keeping fixed-sized messages is that it makes the system-level implementation simpler. The disadvantage is that it makes the task of programming more difficult, and memory is likely to be wasted if not all of it is used. For variable-sized messages, the opposite is true. This type of trade-off is common and reviewed by operating system designers.

Process Analysis

We want to find out as much as possible about a running process on our machine with no access to its source code. Specifically, we want to find information on the command line that created the process, the running executable image, aspects of the status, and how much IO was performed. We can start this by running the following commands:

```
$ sleep 600&

[1] 406

$ cd /proc/406/

$ ls
```

attr	cpuset	maps	oom_adj	smaps
autogroup	cwd	mem	oom_score	stack
auxv	environ	mountinfo	oom_score_adj	stat
cgroup	exe	mounts	pagemap	statm
clear_refs	fd	mountstats	personality	status
cmdline	fdinfo	net	root	syscall
comm	io	ns	sched	task
coredump_filter	limits	numa_maps	schedstat	wchan

If we want to find information on the command line that was used to create the process, the file 'cmdline' contains useful information. We can view the contents by running the command:

```
$ cat cmdline

sleep600
```

This prints out the command invoked on the command line along with the arguments passed. The command and each following argument is separated by the null byte '\0', rather than a space as you may see when viewing the file.

The running executable image is the file 'exe' (highlighted in light blue). This is essentially a link to the same executable image you would find inside the directory /usr/bin/ (in this case, this executable image is at /usr/bin/sleep). This is proven by invoking the command:

```
$ whereis sleep

/usr/bin/sleep
```

To view aspects of the status, we can see the status of the process in the file 'status'. To view the contents, we run the command:

```
$ cat status

Name:  sleep

State: S (sleeping)

Tgid:  406
```

```
Ngid:  0
Pid:   406
PPid:  404
TracerPid:   0
Uid:   1000   1000   1000   1000
Gid:   1000   1000   1000   1000
FDSize:      256
Groups:      10 1000
VmPeak:          5768 kB
VmSize:          5768 kB
VmLck:        0 kB
VmPin:        0 kB
VmHWM:      616 kB
VmRSS:      616 kB
VmData:           172 kB
VmStk:      136 kB
VmExe:       28 kB
VmLib:     1772 kB
VmPTE:       32 kB
VmSwap:             0 kB
Threads:     1
SigQ:  0/5924
SigPnd:      0000000000000000
ShdPnd:      0000000000000000
SigBlk:      0000000000000000
SigIgn:      0000000000000000
SigCgt:      0000000000000000
CapInh:      0000000000000000
CapPrm:      0000000000000000
CapEff:      0000000000000000
CapBnd:      0000001fffffffff
Seccomp:     0
Cpus_allowed:1
```

```
Cpus_allowed_list:  0

Mems_allowed:00000000,00000001

Mems_allowed_list:  0

voluntary_ctxt_switches:  3

nonvoluntary_ctxt_switches:      0
```

This file is useful in that it provides all the information we would find in the files ‘stat’ and ‘statm’, but in a more readable format. The file contains information on the process’s status, its ID, its parent’s ID, thread ID, session ID, etc. Other useful information on the process you would want to know such as use of threads and memory are also listed in this file.

Finally, to see the IO performance of the thread, we can view the contents of the io file by running the command:

```
$ cat io

rchar: 2012

wchar: 0

syscr: 7

syscw: 0

read_bytes: 0

write_bytes: 0

cancelled_write_bytes: 0
```

Here, the file reports the number of characters read and written, the number of system calls read and written, and the number of bytes read and written by the process.

More information on processes, their attributes, and the /proc/PID/ directory can be found on the Linux manual page for PROC(5).