Caleb Faruki
Dr. Werschulz
CISC3595 Operating Systems
October 7, 2013

Chapter 3 Homework

3.7

The kernel saves the context of the old process in its PCB (process control block). The PCB includes the values of the CPU registers being used, the state of the process, and memory-management information. Then, the kernel loads the saved context of the next process from the PCB that is scheduled to run.

3.9

Not including the parent process (i.e. the main one invoking the fork() functions), 3 processes have been created.

3.10

Using the program in Figure 3.23, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)
In my program, the output is as follows:

```
parent: pid = 21474
parent: pid1 = 21473
child: pid = 0
child: pid1 = 21474
```

The values of the pids from A to D are 0, 21474, 21474, and 21473. So, in the case, where the parent and child are 2600 and 2603:
A: 0, B: 2603: C: 2603, D: 2600.

3.13

a. Synchronous and asynchronous communication:

Synchronous communication has less overhead, since the need to establish a connection occurs once. These are also easier to implement. And they typically offer better error recovery and the advantage of responses in real-time. But the disadvantage is that the process for synchronous comms must be persistent, meaning that the resources for this process are tied up, even if messages aren't being sent.

Asynchronous communication doesn't require that sender

and receiver processes be simultaneously active. For that reason, these processes need not be persistent, so resources may be allocated when the sender wants to send or the receiver wants to receive. However, the disadvantage is that response times are unpredictable. And each message comes with overhead, meaning inefficiency.

b. Automatic and explicit buffering:

Automatic buffering (i.e. bounded/unbounded buffering) permits an indefinitely long queue for sending and receiving messages, meaning that a sender need not be blocked in order to assure a message is sent. But this means that memory might be wasted.

Explicit buffering means that wasting memory is less likely. However, processes might be blocked while waiting for available space in the queue or, in the case of no buffering, waiting for the process to be open.

c. Send by copy and send by reference:

Send-by-copy is good for tasks where the integrity of original data is important. This is a good practice for data protection. But it doesn't offer flexibility that passing-by-reference does.

Send-by-reference offers the flexibility of allowing changes to data that affect how subsequent processes will run. But it is a dangerous practice for programmers who are not thorough in controlling for errors. Modifying original data can cause bugs in other processes.

d. Fixed-sized and variable-sized messages:

For fixed-sized messages, system-level implementation. But the programming is more difficult.

Conversely, variable-sized messages are harder to implement. But programming it is not so hard.