Christian Shinkle

ComS 352 HW2

3.09 The kernel saves the context of the current process, which includes registers, I/O handles, and main memory points. Then, the kernel loads the instructions for the new process into main memory and begins executing the new process.

3.12 16 processes are created. The parent process will create 4 children. Then, the first child will create 3 children, the second child will create 2, the third child will create 1, and the last child will create 0. However, this process will continue for each sub-child and so on. In total, the sum comes out to 16.

3.15 A situation where a ordinary pipe is suitable is a pair of processes where one process has to send messages and the other has to read them, so a unilateral relationship and a parent-child relationship.

A situation where a named pipe is suitable is a set of processes that need to communicate both ways, so a bilateral relationship, such as processes over a network. Named pipes are also useful for when multiple processes have to write to the same pipe.

3.17 For line X, it prints off “CHILD:” followed by the negative square of each index in nums. For line Y, after line X finished, it print off “CHILD:“ followed by each number it nums. This occurs because the contents of nums is copied from the parent to the child process. So, the data in nums is modified in the child’s context, but not in the parent’s.

3.18

A. Synchronous communication blocks the consumer and producer from receiving messages. This allows for a rendezvous which solves the producer-consumer problem of when and when not to allow sending and receiving of messages. The downside for synchronous is a performance loss since blocked producer and consumer will need to wait for their respective pair to become synchronized. Asynchronous, on the other hand,

doesn’t need to worry whether a message has been received or not so no blocking is required. Many types of messages don’t need to wait for the consumer to receive them, so asynchronous may allow for better performance. The downside to asynchronous is it is more difficult to program because programmers need to build a buffer to account for the unblocked message to be received.

B. Automatic buffering allows for as many messaged to be stored as necessary. This is a benefit if the programmer doesn’t know ahead of time how many messaged need to be stored. The downside is lots of memory can be wasted on storing messages that could be used elsewhere. Explicit buffering allows for a strict limit of messaged to be stored before the producers are blocked. This allows for more efficient memory usage. The downside is producers will be stalled while waiting for explicit buffers to clear.

C. Send by copy works better if the programmer can guarantee the message won’t change or be changed by the consumer. It is much easy for the programmer to create this system. Send by reference allows for the message to be changed, but is more difficult to create.

D. Fixed-sized messages are easier for the kernel developer to create, but will cause difficult for the programmer because they will have to find ways around messagest that are larger than the fixed size. The opposite is true for variable-sized messages. They are more difficult for kernel developers to allow, but programmers find variable message to be easier to work with.