[40 out of 40]

OS Ch3 Spring 2013

Name: **Ryan Morris**

[4] 1. On a **quad-core** 2.5 GHz cpu, with a time slice of 10 milliseconds, and assuming one instruction per clock cycle, about how many instructions will be executed in a time slice?

**4 \* [(10\*10^-3) \* (2.5\*10^9)] = *1.0 x 10^8 instructions***

*4 cores isn't useful for this calculation – there are just 4 schedulers (more or less) for each core.*

*Real Anwser 1.0 x 10^8 /4 = 2.5 x 10^7*

[4] 2. Why might a process use less than an entire time slice? **A process can only execute instructions when it has something to do, however, many process will waste time waiting for thinks like File Reads/Writes, input from the user and resource locks.**

[4] 3. Even though fork-exec (unix) is conceptually more complex than CreateProcess (win32), what advantages does it have? **Unix fork() uses copy-on write so duplicating a process occurs faster and requires fewer resources to do so, forked process can then be replaced with any other desired binary with exec().**

[4] 4. What is the difference between synchronous and asynchronous communication? **Synchronous transfers are blocking and the process sending information simply waits for the information to be sent before moving on to its next task. Asynchronous communication is non blocking and while often faster then synchronous communication requires logic to handle collisions and determine who gets to send and receive when.**

*Synchronous communication has the sender block for a confirmation from the receiver before continuing. Asynchronous communication does not requires this. As such it can be faster, but typically is more complex.*

[4] 5. Why should there be an upper limit to a packet size? **If there was no upper limit to packet size it would become possible and arbitrary to send a packet so large that it disrupts all other network communication or even takes down the network hardware. Flow control would also become impossible.**

[4] 6. How many processes are created by the following code:

int main()

{

if (fork() == 0) {

if (fork() != 0) {

fork();

}

}

printf(“process %d exiting\n”,getpid());

return 0;

}

**Three new processes are created four are running including the parent.**

[4] 7. If child processes must exit before parent process, the active processes must form a tree. Are there any advantages to this? **Yes the process tree allows for resource management and process control. Terminating a parent process will also terminate any of its children which is useful for stopping crashed programs /zombie processes.**

*The main advantage is runaway processes always live in the process tree below the parent. So if some application has an error that creates many processes, you can always decide who they belong to because of the tree structure. At any given time, you can know who spawned which processes.*

[4] 8. Describe a few mechanisms, giving pros and cons, for interprocess communication. **Direct memory access allows for the fastest interprocess communication but is dangerous as another process is writing arbitrary data to the running process this can cause memory corruption, execution of malicious code, hangs or crashes in one or both processes. Message passing is a slower but more robust mechanism for interprocess communication that involves passing messages of information between two or more running processes. Another advantage of message passing is that it can be done over networks not just on the local machine.**

[4] 9. When sharing information between different systems, what are some issues that must be addressed? **Endianess of the byte encoding, size of data-types, and processor architecture can all cause problems sharing data between different systems.**

[4] 10. Cloning an entire process seems expensive for a fork(), but it typically is not. Why not?

**Fork uses copy on write and and passes pointers to the memory of the parent process unless part of the child process has actually changed in which case it only writes what has changed to memory.**