Jeremy Holloway

CPSC-3220-001

5/21/2019

Pre-class Assignment #5

1. Define the following terms:

   concurrency: Multiple activities that can happen at the same time.

   Thread: A single execution sequence that represents a separately scheduled task.

   thread scheduler: Software that maps threads to processors by switching between running threads and threads that are ready but not running.

2. Give four reasons to use threads within a single process.

   1. Program Structure: expressing logically concurrent tasks
   2. Responsiveness: shifting work into the background
   3. Performance: exploiting multiple processors
   4. Performance: managing I/O devices

3. What thread scheduling pattern should you base your program design upon? Explain your answer.

   Programs shouldn’t be designed upon and scheduling pattern because at any moment the processor could be taken from the program.

4. Is a kernel interrupt handler a thread?

   No, an interrupt scheduler is not independently schedulable.

5. Briefly explain what each of these system calls do.

   Create(): Create a new thread, executes the function with an argument arg.

Yield(): The calling thread voluntarily gives up the processor to let some other thread(s) run.

Join(): Wait for the thread to finish if it has not already done so, then return the value passed to thread_exit by that thread.

Exit(): Finish the current thread, then store the value returned in the current threads data structure.

6. Name at least three things that are unique to each thread.

1. Stack information
2. Saved registers
3. Thread metadata
4. Thread control block

7. Name at least three things threads share within a single process.

1. Code
2. Global variables
3. Heap
4. Memory map
5. Page table
6. Any open files

8. Define the purpose of each of the following thread scheduling data structures:

ready list: The set of threads that are ready to be run but which are not currently running.

running list: The set of threads that are currently running.

waiting list: The set of threads that are waiting for a synchronization event or timer expiration to occur before becoming eligible to be run.

finished list: The set of threads that are complete but not yet deallocated, because a join may read the return value from the thread control block.

```
9. Can two threads be ready to execute at the same time?
    Yes


10. Can two threads be executing at the same time?
    Yes, if there are multiple processors
```