Writing Topic 1: Processes

CS 444 Fall 2016

Kevin Stine

Processes

Most modern operating systems are built utilizing processes and threads. In order to maximize the efficiency of handling these processes and threads, operating systems rely on CPU scheduling to monitor their behavior. Three of the major modern operating systems: Windows, FreeBSD, and Linux all utilize processes, threads and scheduling. A process provides the resources which are needed to execute a program. This would include items such as the virtual address space, executable code, unique processes identifiers, environment variables, and at least one thread of execution. A thread is an entity within a process that can be scheduled for execution. These threads maintain exception handlers, scheduling priorities and a unique thread identifier. Multiple threads are able to exist within one process, each executing concurrently and sharing resources. These threads are managed independently by the scheduler of the operating system. CPU scheduling is utilized to keep all the computer resources busy or to allow multiple users to share system resources effectively. Each of the three respective operating systems implement some way of managing processes, threads and scheduling.

Microsoft Windows supports preemptive multitasking, a concept which creates the effect of simultaneous execution of multiple threads from multiple processes. While multiple threads from multiple processes are not actually capable of running simultaneously, the OS does a great job at hiding that fact. To the user, they see multiple windows open, multiple programs running and can assume that everything is just working simultaneously when in fact, the OS is working hard to provide each processes with a fair amount of time on the CPU. Within Windows, threads are scheduled to run based on their scheduling priority, which ranges from zero (lowest priority) to 31 (highest priority). Despite having a zero priority, only the zero-page thread is allowed to have a priority of zero. Windows also utilizes the round-robin method for process scheduling. The round-robin scheduler utilizes time-sharing, giving each job a time slice which is the amount of time it is allowed on the CPU. If the job is not completed when the time slice is up, the job is interrupted and resumed once another time slice is assigned to that process. Windows treats all threads with the same priority as equal, and begins by assigning time slices in a round-robin fashion to all threads with the highest priority. If those processes are not ready to run, the system moves to the next highest priority processes and assigns them a time slice. Windows creates a base priority, which is a combination of the process priority class and the thread priority level. The process priority classes can be idle, below normal, normal, above normal, high priority, and real time. The thread priorities can be idle, lowest, below normal, normal, above normal, highest and time critical.

References

[1] Marshall Kirk McCusick and George V. Neville-Neil, Design and Implementation of the FreeBSD Operating System 2/e, Addison-Wesley, 2015, ISBN: 978-0-32196897-5

[2] Robert Love, Linux Kernel Development 3/e, Addison-Wesley, 2010, ISBN: 978-1-672-32946-3

[3] Windows Dev Center, About Processes and Threads, Microsoft Corporation, https://msdn.microsoft.com/en-us/library/windows/desktop/ms681917(v=vs.85).aspx