TidBIT

Understanding how an operating system or platform handles multiple tasks at once, the difference between a process and thread, and the difference between main memory and virtual memory are concepts that you need to be familiar with. It is not necessary to memorize this information. Rather, you should grasp these concepts to the extent that you can explain them to others in nontechnical terminology.

People who create operating systems, cloud platforms, and language compilers are intimately familiar with all the low-level details supporting these concepts. For us, as computer science students early in our journey, it is necessary to know *of* them and the implications they have on application design.

Review these resources in preparation for next week's concepts, which include a deeper dive into the characteristics, advantages, and weaknesses of various operating platforms from a software developer's perspective.

Required Resources

Reading: *Operating System Concepts*, Chapter 3 (https://go.oreilly.com/SNHU/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_c03_r1.html)
Review the following sections to learn about the basic characteristics of a process, how processes are scheduled, and strategies for communication in client-server systems:

- Process Concept

- Process Scheduling
- Communication in Client-Server Systems

When you start a program you wrote or an app like Microsoft Word, the process is the runtime space that the operating platform sets aside to execute the program within. To the running application, the process provides the illusion that the entire computer's resources (cpu, memory, storage, input/output) are available. The operating system manages many processes at once, with each being isolated and independent from the others. Once a process starts running, it is allowed to execute instructions for a short period of time before it is paused and another process is given a chance to execute instructions. Processes provide isolation so that one application cannot interfere with another application, and the operating system is constantly switching between processes. In relative terms, a process can be expensive to start up and manage because each process essentially sees the entire operating platform as being isolated to itself.

Reading: *Operating System Concepts*, Chapter 4 (https://go.oreilly.com/SNHU/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_c04_r1.html)
Review the following sections to learn about the different types of multithreading models for establishing a relationship between user threads and kernel threads and important issues to consider with multithreaded programs:

- Multithreading Models
- Threading Issues

Threads are how your programs are able to multitask by doing several things at once. Threads are written and managed within application code and require more advanced programming effort than a simple set of linear instructions such as a main() method. An application that is updating a status bar or showing a progress bar while also allowing the user to type input and use a mouse to select graphical icons is a familiar example of multi-threaded programming.

Reading: *Operating System Concepts*, Chapter 8 (https://go.oreilly.com/SNHU/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_c08_r1.html)
Review the following sections to learn about the characteristics of memory, the function of swapping processes, contiguous memory allocation (a common method for allocating memory in an efficient manner), and how paging supports memory management.

- Chapter 8 Background
- Swapping
- Contiguous Memory Allocation
- Paging

Reading: *Operating System Concepts*, Chapter 9 (https://go.oreilly.com/SNHU/library/view/operating-system-concepts/9780470128725/silb_9780470128725_oeb_c09_r1.html)
Review the following sections to learn about the benefits of and common techniques used in virtual memory systems.

- Chapter 9 Background
- Demand Paging
- Copy-on-Write
- Page Replacement