In Chapter 5, we showed how the CPU can be shared by a set of processes. As a result of CPU scheduling, we can improve both the utilization of the CPU and the speed of the computer’s response to its users. To realize this increase in performance, however, we must keep many processes in memory—that is, we must share memory.

In this chapter, we discuss various ways to manage memory. The memory-management algorithms vary from a primitive bare-machine approach to a strategy that uses paging. Each approach has its own advantages and disadvantages. Selection of a memory-management method for a specific system depends on many factors, especially on the hardware design of the system. As we shall see, most algorithms require hardware support, leading many systems to have closely integrated hardware and operating-system memory management.

Bibliographical Notes

Dynamic storage allocation was discussed by [Knuth (1973)] (Section 2.5), who found through simulation that first fit is generally superior to best fit. [Knuth (1973)] also discussed the 50-percent rule.

The concept of paging can be credited to the designers of the Atlas system, which has been described by [Kilburn et al. (1961)] and by [Howarth et al. (1961)]. The concept of segmentation was first discussed by [Dennis (1965)]. Paged segmentation was first supported in the GE 645, on which MULTICS was originally implemented ([Organick (1972)] and [Daley and Dennis (1967)]).

Inverted page tables are discussed in an article about the IBM RT storage manager by [Chang and Mergen (1988)].

[Hennessy and Patterson (2012)] explains the hardware aspects of TLBs, caches, and MMUs. [Talluri et al. (1995)] discusses page tables for 64-bit address spaces. [Jacob and Mudge (2001)] describes techniques for managing the TLB. [Fang et al. (2001)] evaluates support for large pages.


Bibliography


