Chap. 1 (Operating Systems Concepts, 10th Ed.)
- Operating system vs. user application (Fig. 1.1)
- Computer system architecture (Fig. 1.2)
- Bus, interrupt, interrupt request line (Fig. 1.4)
- RAM, hard-disk drive (HDD), nonvolatile (flash) memory (Fig. 1.6)
- Single-processor system vs. multi-processor system
- Physical vs. virtual memory
- Multiprogramming and multitasking (Fig. 1.12)
- User vs. kernel mode, privileged instructions (Fig. 1.13)

Chap. 2 (same; except Sects. 2.8.2, 2.9)
- OS functions (program execution, resource management, I/O)
- System calls and types thereof (Figs. 2.5 and 2.8)
- Linkers and loaders
- Application programming interface (API) vs. application binary interface (ABI)
- ELF executable format
- Separation of policy from mechanism
- Monolithic OSes (Figs. 2.12 and 2.13)
- Layered OSes (Fig. 2.14)

Chap. 3 (same)
- Process definition
- Text, data, heap, stack parts of processes (Fig. 3.1)
- Process states (Fig. 3.2)

Chap. 4 (same; except Sect. 4.5)
- Thread definition, contrast with process
- Thread-private info vs. info shared among threads
- Motivation for, benefits of multitasking
- Concurrency vs. parallelism
- User-level vs. kernel-level threads
- Relationships between user-, kernel-level threads (Figs. 4.7–4.9)
- POSIX threads (Pthreads) library calls and semantics
- Issues:
  - fork(), exec() system calls
  - Signal handing
  - Thread cancellation
- Scheduler activations

Chap. 5 (same; except Sects. 5.7.2, 5.8)
- CPU, I/O burst cycles
- CPU scheduler
- Preemptive vs. non-preemptive scheduling
- Scheduling criteria for batch, interactive systems:
  - CPU utilization
  - Throughput
  - Turnaround time
  - Wait time
  - Response time
- Single-CPU scheduling algorithms:
  - FCFS
  - SJF (non-preemptive, preemptive)
  - Priority scheduling (non-preemptive, preemptive)
  - Round-robin
  - Multi-level queues
- Multiple-CPU scheduling:
  - Load balancing
  - Multicore CPUs
  - Gang scheduling
- Case study: Linux scheduling (high-level)

Chap. 6 (same; except Sect. 6.9)
- Race conditions
- Definition of critical-section problem
- Requirements for solution to critical-section problem:
  - Mutual exclusion
  - Progress
  - Bounded waiting
- Preemptive vs. non-preemptive kernels

Page 1 of 2
Chap. 6 (Operating Systems Concepts, 10TH ED., cont’d; except Sect. 6.9)
- Non-solutions:
  o Disabling interrupts
  o Lock variables
  o Strict alternation
- Busy waiting approaches:
  o Peterson’s solution
  o Test_set_lock() (TSL), compare-and-swap() (CAS) hardware instructions
  o Sleep and wakeup
- Semaphores and mutex locks
  o Semantics
  o Implementations
- Monitors
- Barriers
- Liveness failures:
  o Deadlocks
  o Priority inversion

Chap. 7 (same; except Sects. 7.4–7.5)
- Bounded-buffer problem
- Readers-writers problem
- Dining-philosophers problem
- Solutions using C (Pthreads)

Chap. 8 (same)
- Resource classifications:
  - Serially reusable vs consumable
  - Preemptible vs. non-preemptible
  - Shared among processes vs. dedicated to one process exclusively
- Deadlock definition
- Deadlocked process vs. infinitely postponed
- Deadlock vs. livelock
- Necessary conditions for deadlock:
  o Mutual exclusion
  o Hold and wait
  o No preemption
  o Circular wait
- Resource-allocation graphs
  o Cases: one resource vs. multiple resources per type
  o Implication of cycles in graphs (both cases)
- Non-solution for deadlock: ostrich algorithm
- Deadlock prevention:
  o Havender’s algorithms (break necessary conditions)
  o Two-phase locking
- Deadlock avoidance:
  o Safe vs. unsafe states
  o Resource-allocation graph (one resource per type)
  o Banker’s algorithm (multiple resources per type)
- Deadlock detection
  o Resource-allocation and wait-for graphs
  o Variant of banker’s algorithm
- Deadlock recovery
  o Process and thread termination
  o Resource preemption and implication of starvation
  o Checkpoints for processes