History of Operating Systems

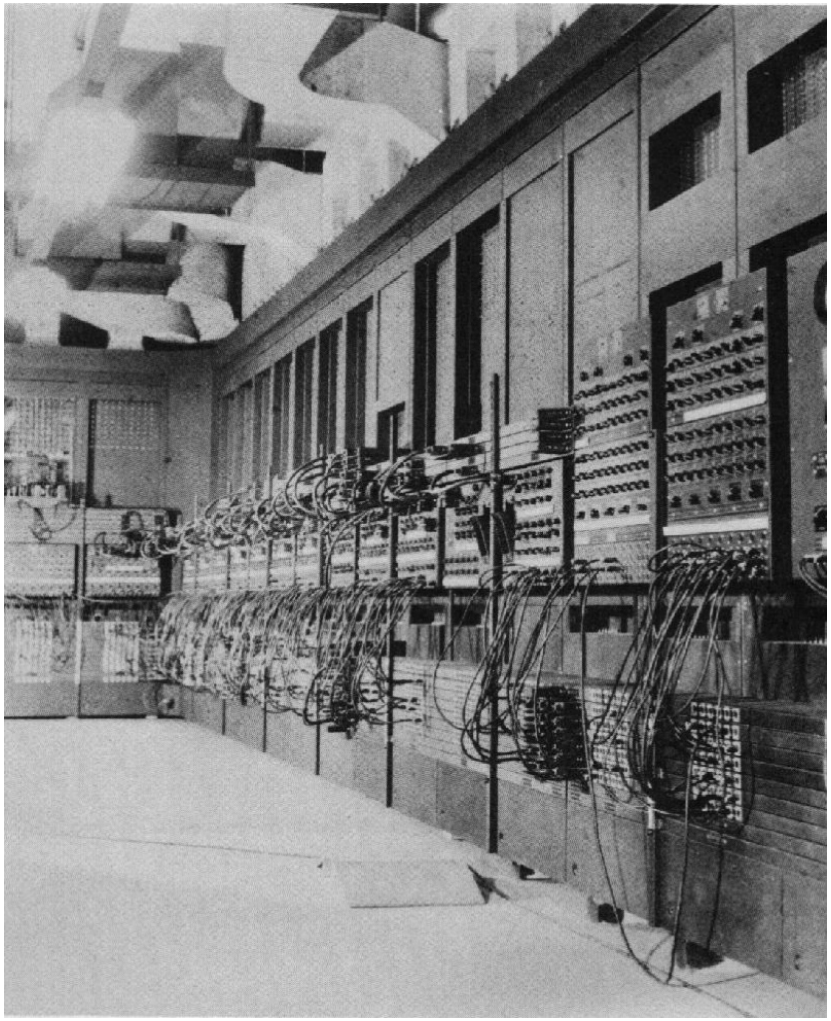
Overview

- Phase 1: Hardware expensive, humans cheap
- Phase 2: Hardware cheap, humans expensive
- Phase 3: Hardware very cheap, humans very expensive

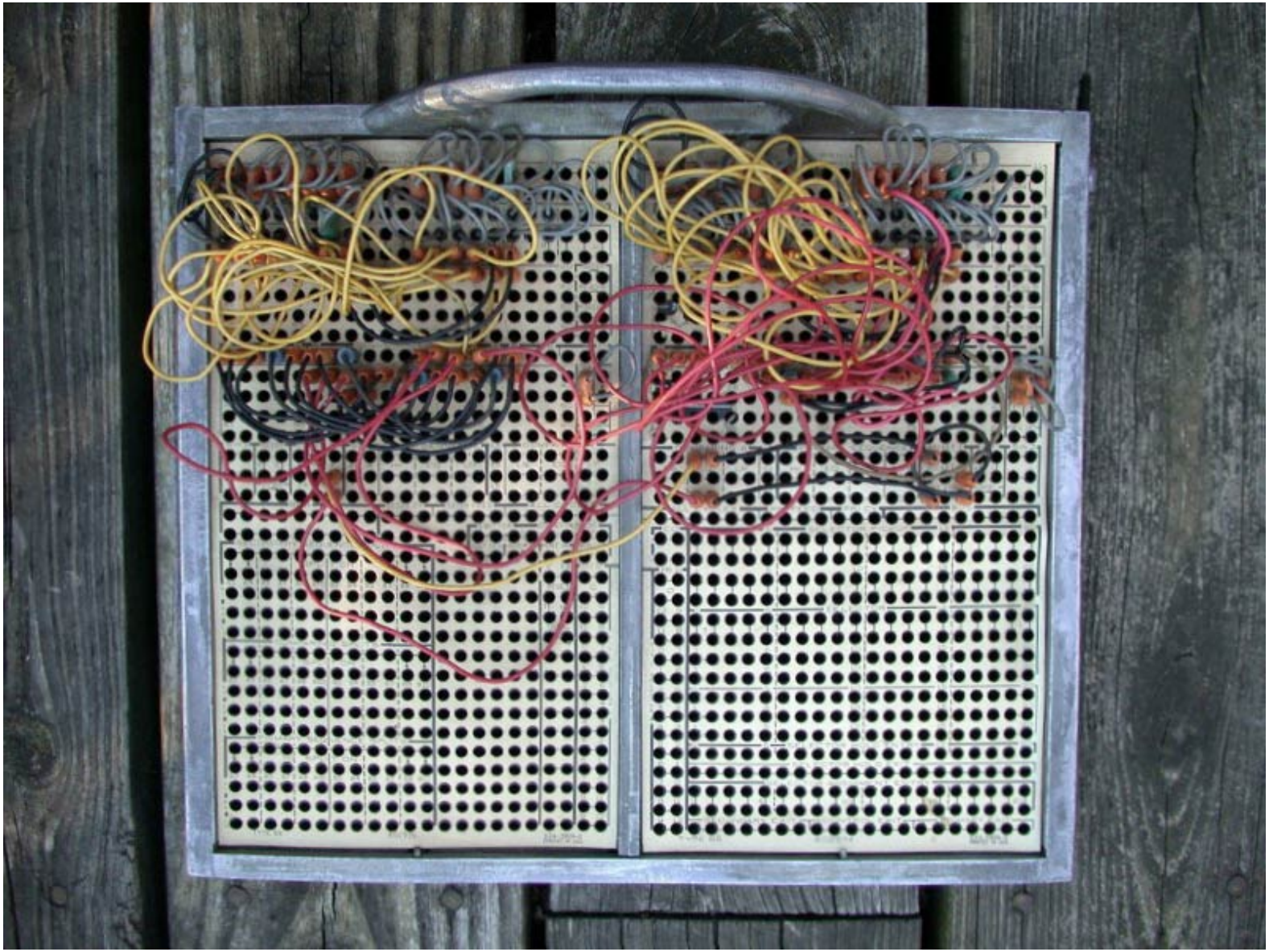
The Beginnings

Phase 1: Expensive Hardware, Cheap Humans

1. One user on the console, one process at a time (1945-1955)
   - Single user system
   - Beginning: programming directly with wires
   - Later: program is a stack of cards
   - OS is a subroutine library (and a loader)
     - A stack of cards you pull off the shelf to do, for example, a matrix multiply
   - Problem: Low utilization of expensive components



A Sample Plugboard

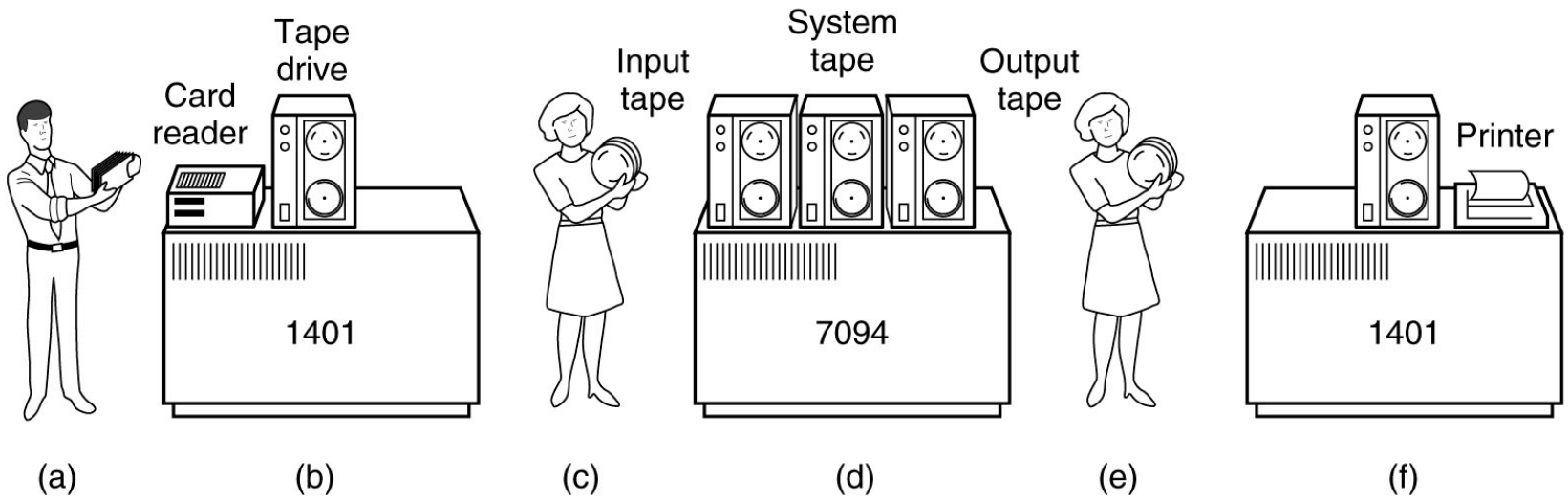


From IBM's "History of the 1401"

Phase 1:
Expensive Hardware, Cheap Humans

2. Batch processing: load program, run, output to tape, print results, repeat (1955-1965)
   - Users give their program (on punch cards) to a human who schedules the jobs in batches
   - Each batch is input into a card reader and output on tape
   - Tape is carried to CPU machine
   - OS on CPU machine loads each job in the bath from tape, runs, and writes any output to another tape
   - Advantage: next job can be loaded immediately as previous one finishes
     - Better use of hardware but debugging is much more difficult
   - Disadvantages:
     - No protection---a buggy program can crash the batch monitor
     - Computer is idle during I/O

Batch Processing

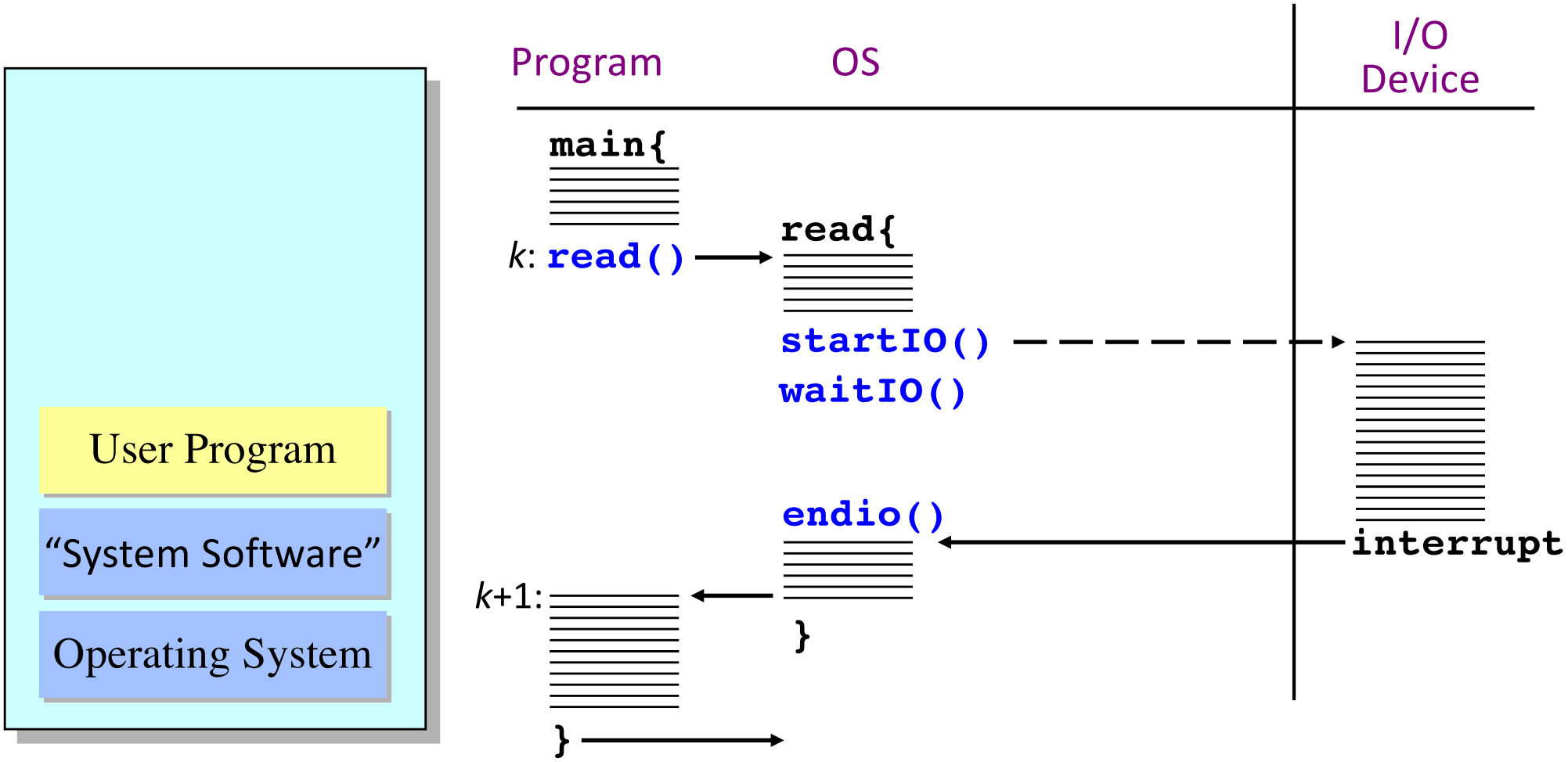


An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

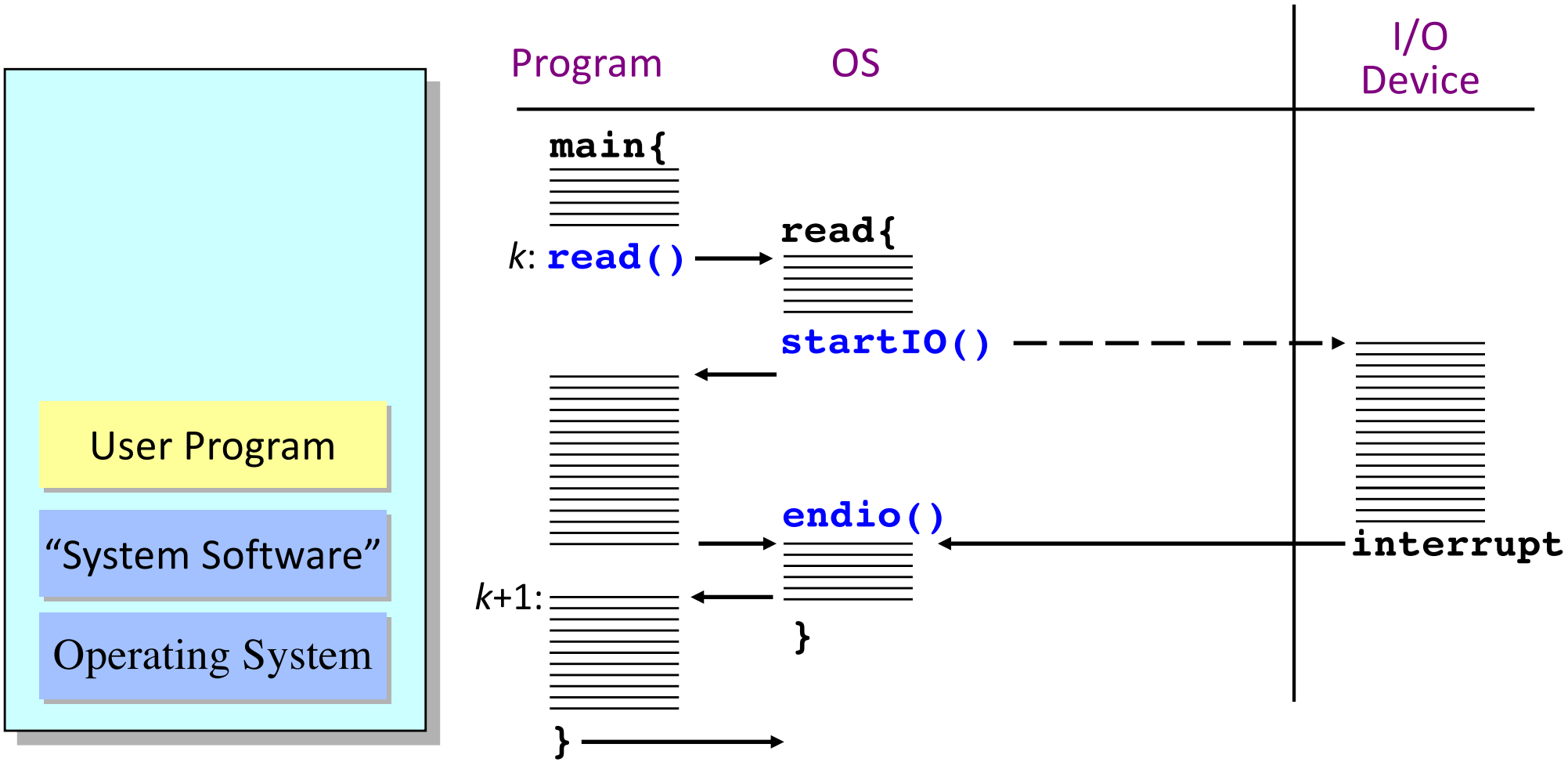
Phase 1:
Expensive Hardware, Cheap Humans

3. Overlap of I/O and computation, interrupts
   - OS requests I/O, goes back to computation, gets interrupt when I/O device finishes
   - No sharing, only protect OS from applications
   - Add concurrency *within same process*
   - Buffering and interrupt handling in OS
   - Spool jobs on the drum
   - Performance improves because I/O and processing happen concurrently

Overlapping I/O and Computation



Overlapping I/O and Computation



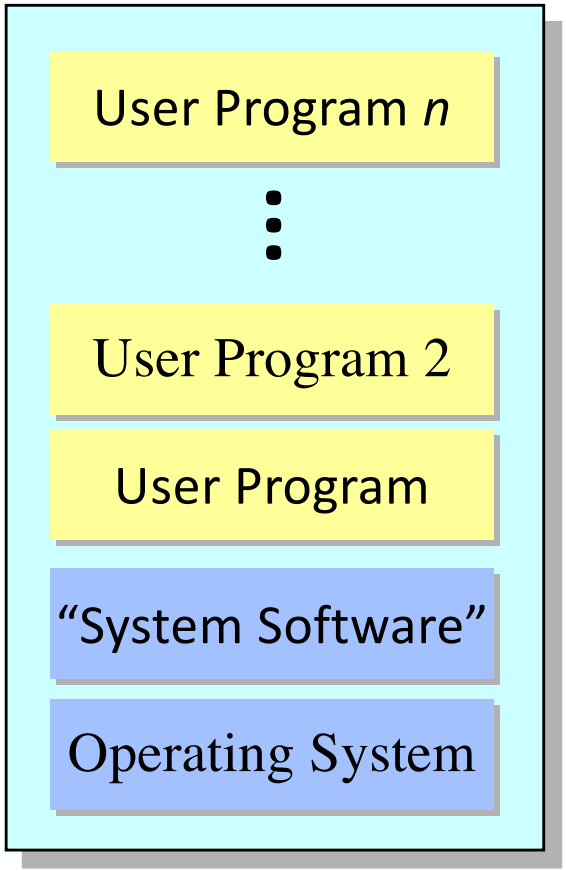
Multiprogramming

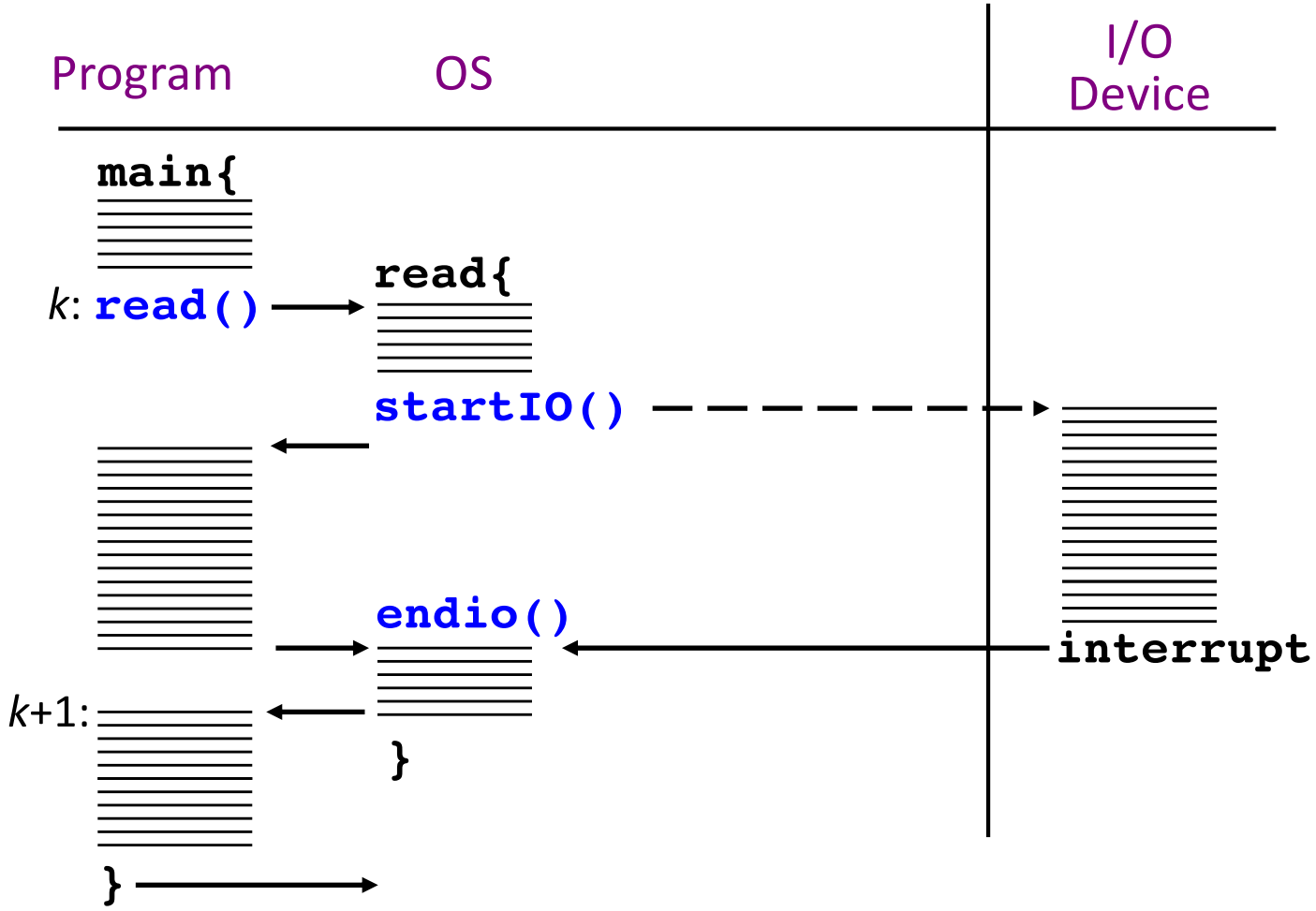
Phase 1:
Expensive Hardware, Cheap Humans

4. Multiprogramming: several programs run at the same time sharing the machine
   - One job runs until it performs I/O, then another job gets the CPU
   - OS manages interactions between concurrent programs (which ones start and execute, provides protection)
   - Requires: Memory protection and relocation

Multiprogramming (1965-1980)

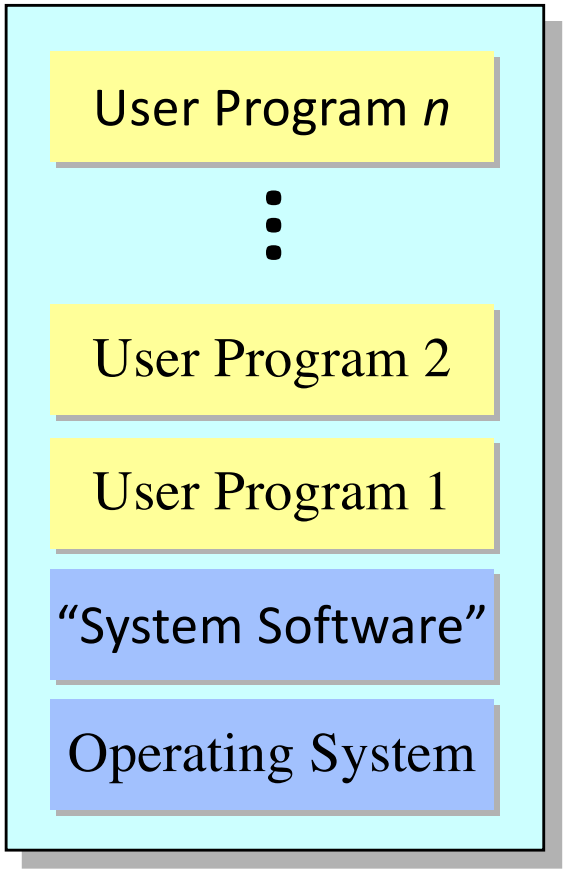
Keep several jobs in memory and multiplex CPU between jobs

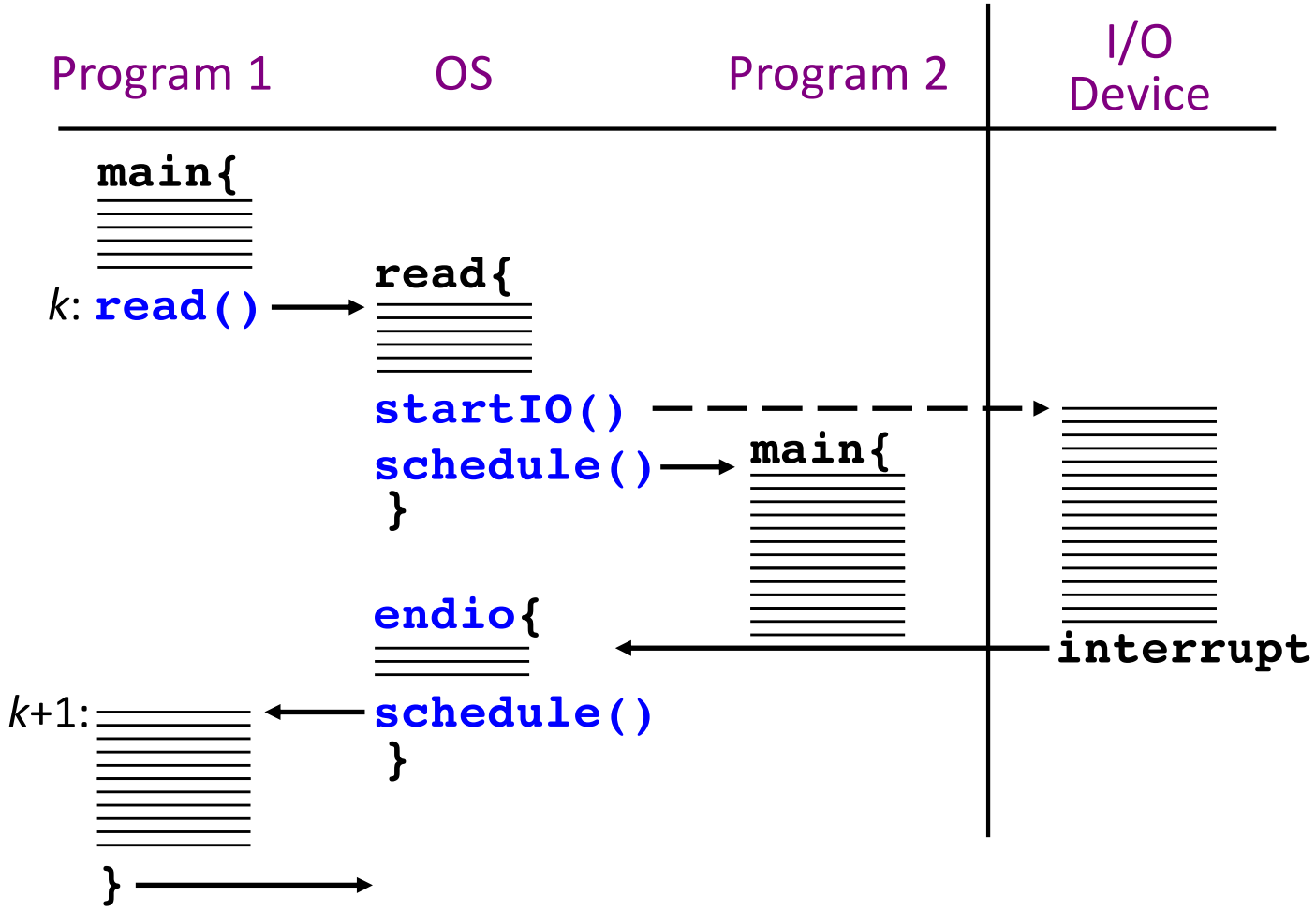




Multiprogramming (1965-1980)

Keep several jobs in memory and multiplex CPU between jobs





iClicker Question

In batch systems, each job must completely finish before the next job may begin.

A. True

B. False

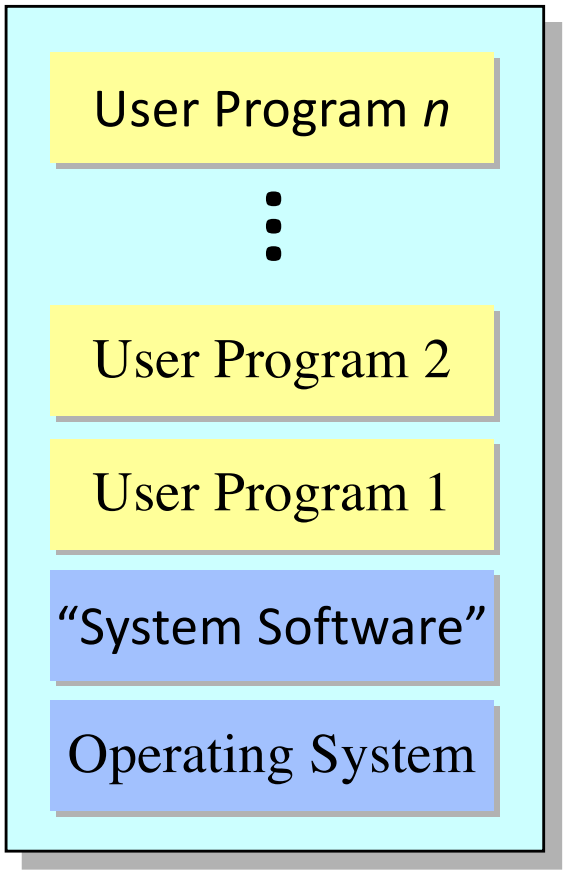
Interactive Timesharing

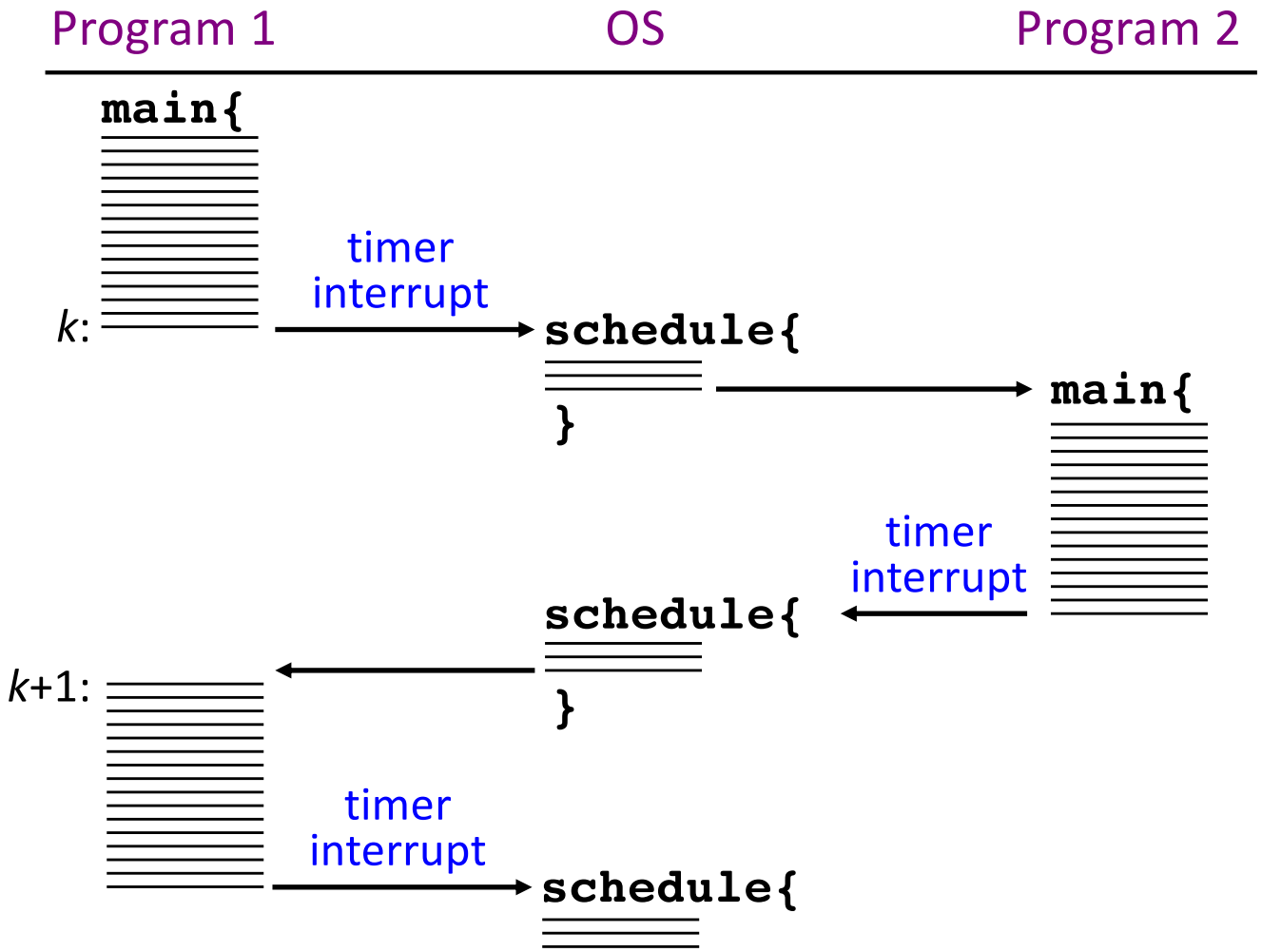
Phase 2:
Cheap Hardware, Expensive Humans

5. Interactive timesharing (1970-)
   - Use cheap terminals to let multiple users interact with the system at the same time
     - Debugging is a lot easier
     - Process switching occurs much more frequently
   - Requires: more sharing, more protection, more concurrency
   - New OS services: shell, file system, rapid process switching (users can interact!), virtual memory (processes running simultaneously!)
   - New problems: response time, thrashing

Timesharing (1970-)

A timer interrupt is used to multiplex CPU among jobs





Today

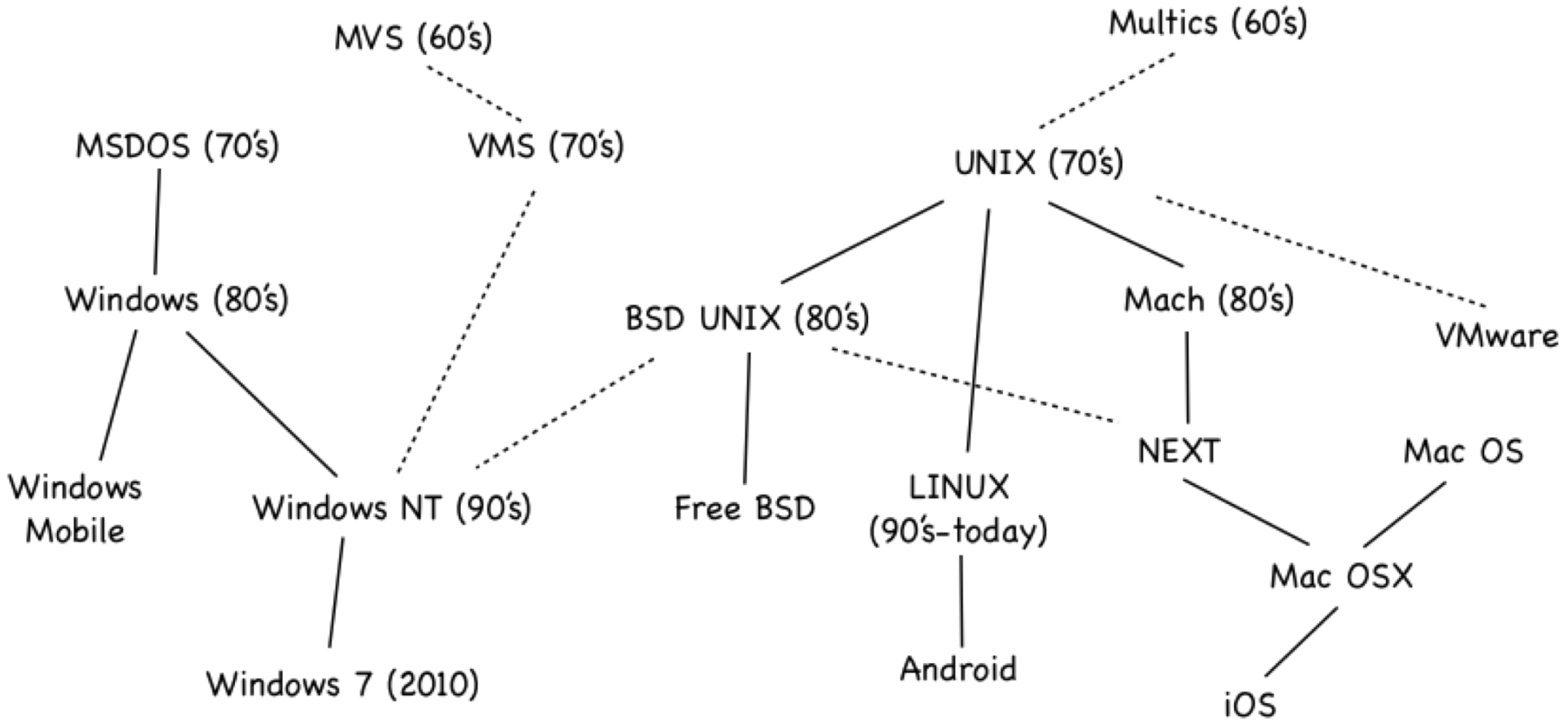
Phase 3: Very Cheap Hardware, Very Expensive Humans

6. Personal computing
   - Computers are cheap, so give everyone a computer
   - Simplify OS by eliminating multiprogramming, concurrency, and protection
     - A subroutine library again! (MSDos, MacOS)
     - Failed: humans are expensive, so don't waste their time letting programs crash each other!

Phase 3: Very Cheap Hardware, Very Expensive Humans

7. Parallel and Distributed Computing
   - Computers are SO cheap, give people a bunch of them!
   - In parallel systems, multiple processors are in the same machine, sharing memory, I/O devices, …
   - In distributed systems, multiple processors communicate via a network
   - Advantages: increased performance, increased reliability, sharing of specialized resources

Genealogy of Modern Operating Systems



Operating System Complexity