Department of Physics,
Computer Science & Engineering

CPSC 410 – Operating Systems I

Chapter 2: Operating System Overview

Keith Perkins

Original slides by Dr. Roberto A. Flores

Chapter 2 Topics

- OS functions
  - Objectives, OS as user/computer interface, OS as resource manager
- OS evolution
  - Serial, Batch, Multi-programming, Time sharing
- Achievements
  - Process, Memory management, Information security, Scheduling, System structure

OS Functions

- Functions
  - User/Computer interface
    - An interface between applications and hardware
  - Resource manager
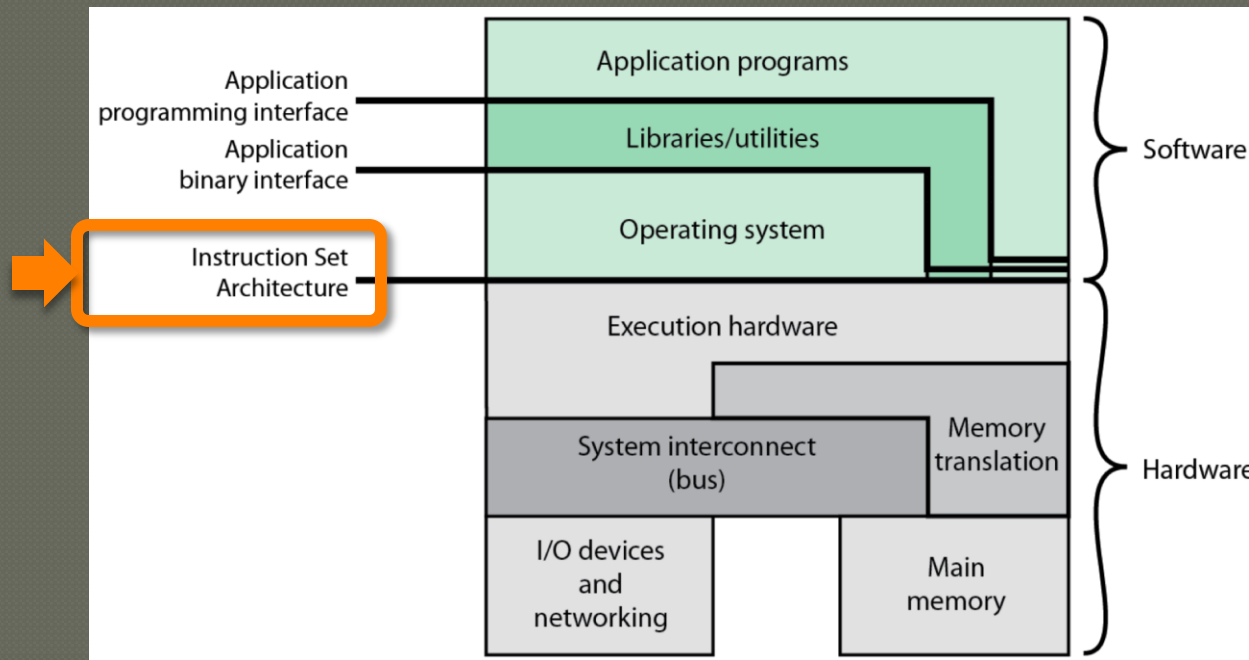   - A program controlling execution of application programs

- Objectives
  - Convenience
  - Efficiency
  - Ability to evolve
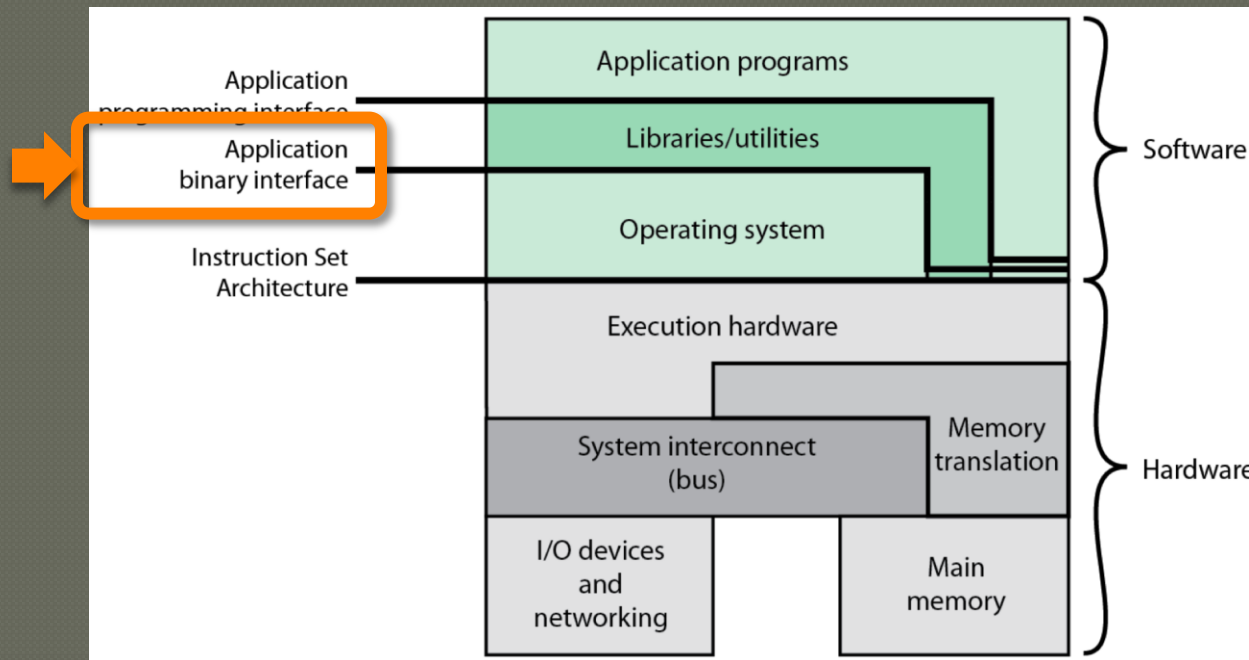
User/Computer Interface

- Key interfaces
  - ISA : Instruction set architecture
    - Machine language instructions hardware can execute
      - add registers, fetch memory, …

User/Computer Interface
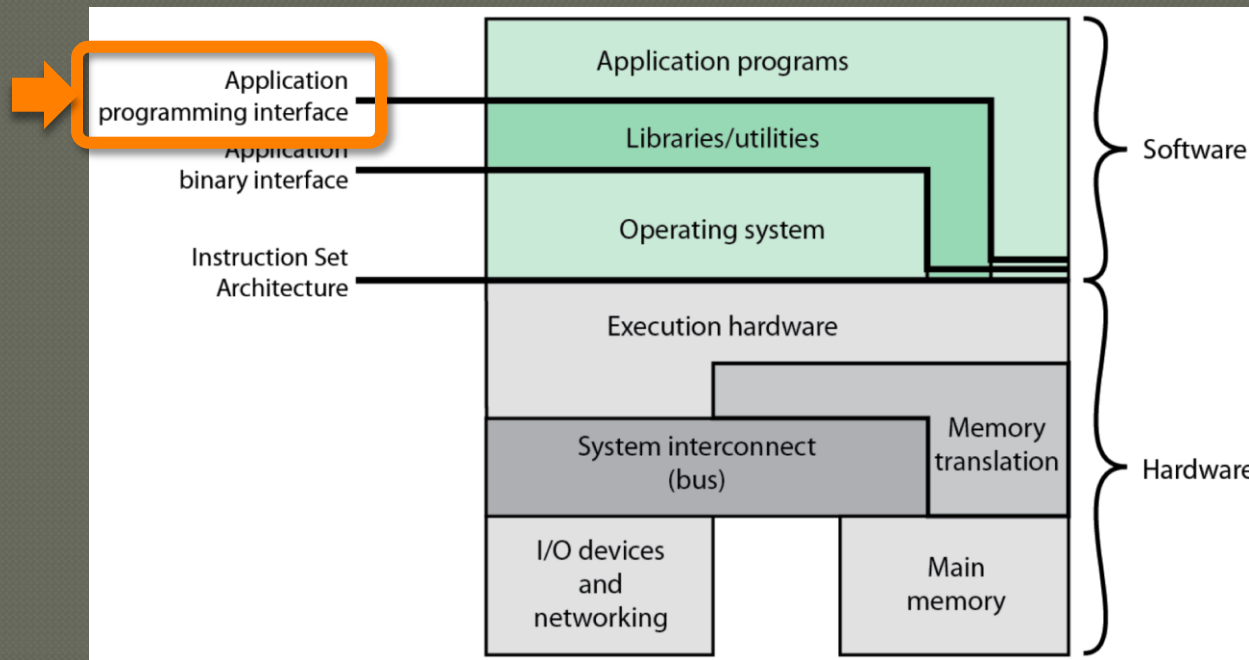
- Key interfaces
  - ABI : Application binary interface
    - Masks hardware details
    - Mediate between programs & computer resources/services
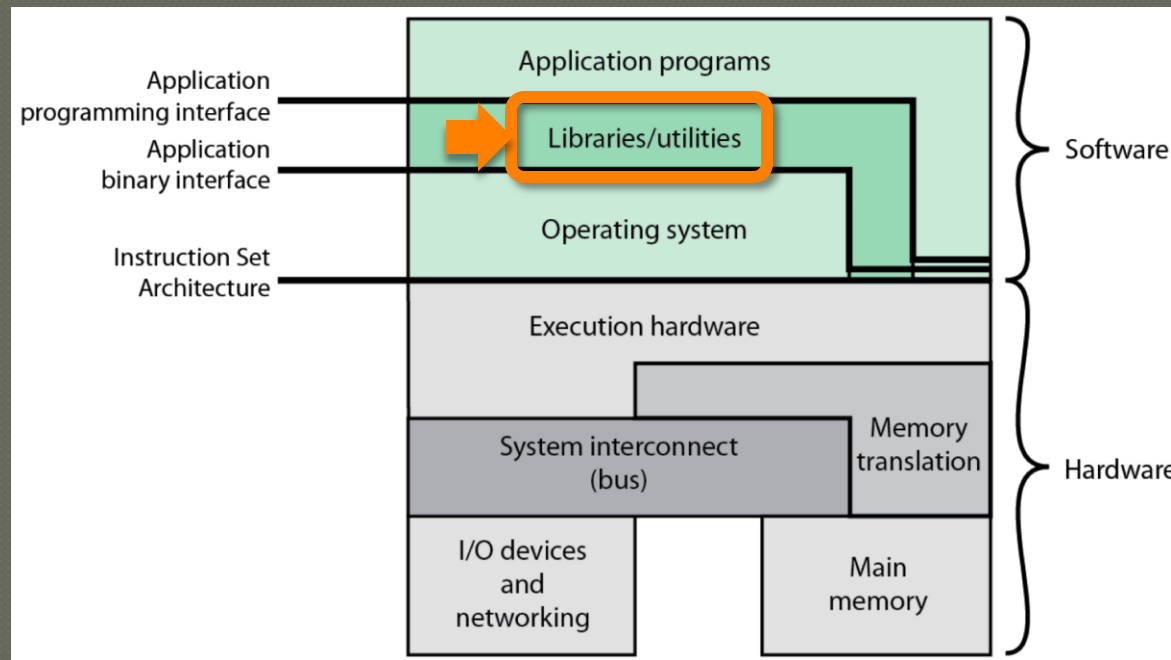
User/Computer Interface

- Key interfaces
  - API : Application programming interface
    - High-level language instructions
    - Facilitates source code portability (re-compilation)

Resource Manager
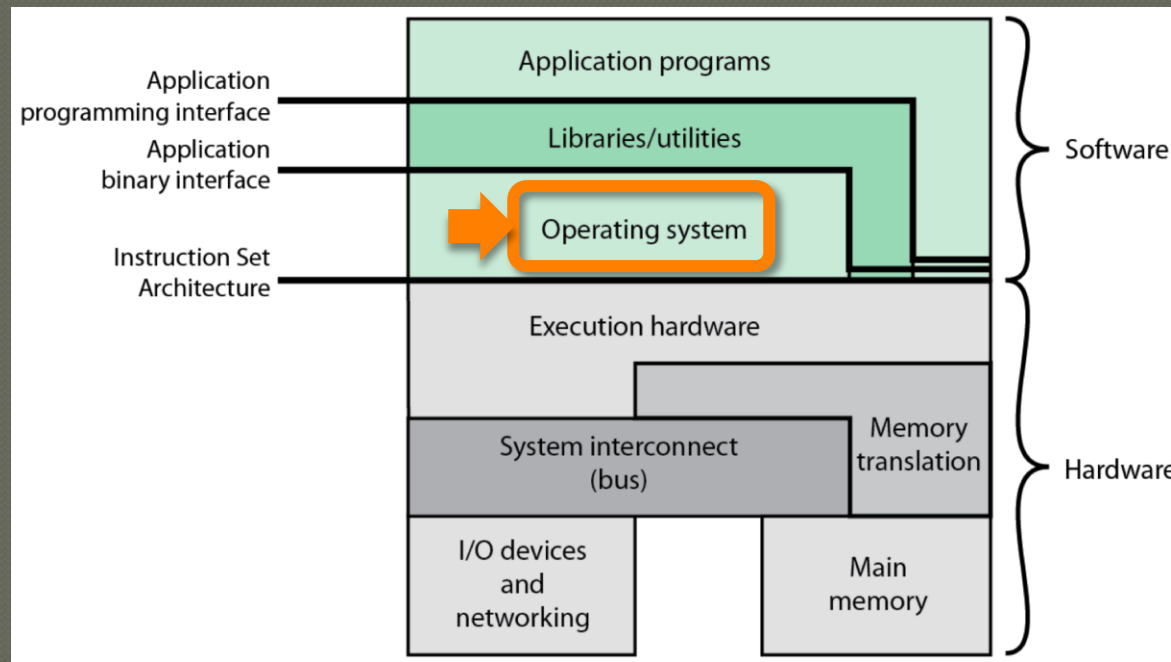
- ◉ OS comes with libraries
  - Implementing functions to support creating programs, managing files, and controlling I/O devices
    - editors, interpreters, I/O modules, …

Resource Manager

- OS
  - Controls hardware resources
  - Manages system services
    - printer spooling, audio, sockets, …

Resource Manager

- OS services
  - Program development
    - editors, compilers, debuggers (not OS)
  - Program execution
    - load data & instructions into memory
    - initialize I/O devices & files
  - Access to I/O devices
    - uniform interface to access I/O devices
      - Standard API (like read/write) to vendor supplied device drivers
  - Controlled access to files
    - control orderly access to files (data integrity).

Resource Manager

- OS services (II)
  - System access
    - control access to resources (<u>permissions</u>)
    - resolve conflicts for resource contention (<u>privileges</u>).
  - Error detection & response
    - minimize disrupting running programs (least disruptive)
      - reporting error, retrying operation, ending erring program
  - Accounting
    - Keeping logs & statistics

Chapter 2 Topics

- OS functions
  - Objectives, OS as user/computer interface, OS as resource manager
- OS evolution
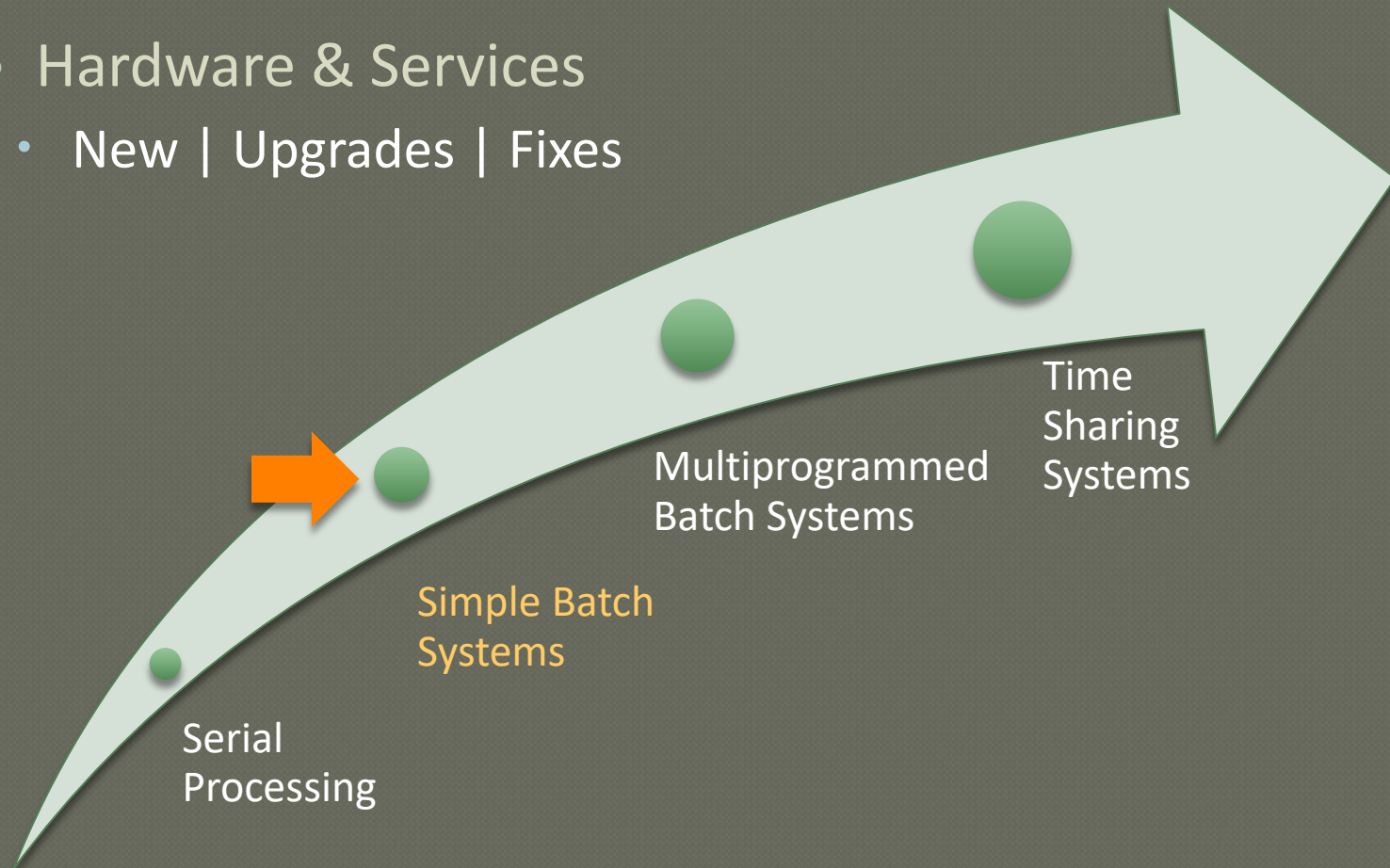 - Serial, Batch, Multi-programming, Time sharing
- Achievements
  - Process, Memory management, Information security, Scheduling, System structure
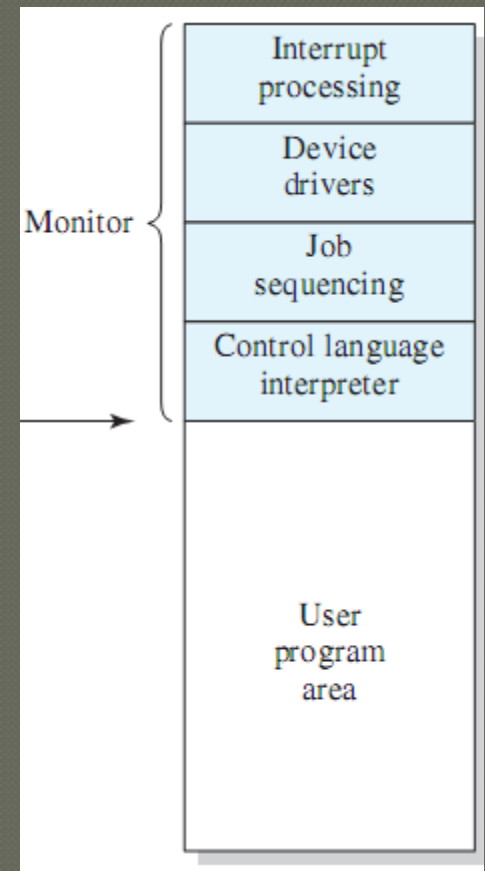
Evolution

- Reasons for OS to evolve
  - Hardware & Services
    - New | Upgrades | Fixes

Time Sharing Systems

Multiprogrammed Batch Systems

Simple Batch Systems

Serial Processing

Simple Batch Systems

- improving computer utilization
  - programmer has no direct access to computer
  - operator batches jobs, feeds them to an input device, then…
- Monitor (aka Batch OS)
  - program controlling the execution of jobs
  - 1. monitor reads next job & yields control of CPU to the job
    - "control is passed to a job" : CPU starts running user program
  - 2. user program ends & monitor continues running again
    - "control is returned to the monitor" : CPU runs monitor

- Simple Batch Systems (II)
  - Job Control Language (JCL)
    - Instructions meant for the monitor (like pre-processing)
      - $JOB $FTN <source code> $LOAD $RUN <data> $END

  - Hardware support of Monitor
    - Memory protection
      - Memory where monitor resides is out-of-bounds for jobs
    - Timer
      - Notifies when jobs run longer than anticipated
    - Privileged instructions
      - Instructions that only the monitor can execute (e.g., load job)
    - Interrupts
      - Signals giving CPU a degree of flexibility

OS Evolution

- **Simple Batch Systems (II)**
  - Job Control Language (JCL)
    - Instructions meant for the monitor (like pre-processing)
      - $JOB $FTN <source code> $LOAD $RUN <data> $END

  - Hardware support of Monitor
    - Memory protection
      - Memory where monit
    - Timer
      - Notifies when jobs ru
    - Privileged instructions
      - Instructions that only the monitor can execute (e.g., load job)
    - Interrupts
      - Signals giving CPU a degree of flexibility

	User Mode	**Kernel Mode**
Applies to…	User programs	Monitor
Memory access	Restricted	Unrestricted
Instructions	Limited	Unlimited

⦿ Reasons for OS to evolve
- Hardware & Services
  - New | Upgrades | Fixes

Time Sharing Systems

Multiprogrammed Batch Systems

Simple Batch Systems

Serial Processing

Multiprogramming

(c) Multiprogramming with three programs

- ⦿ Multiprogramming
  - • also known as multitasking
  - • memory is expanded to hold three, four, or more programs and switch among all of them

Multiprogramming Example

Table 2.1 Sample Program Execution Attributes

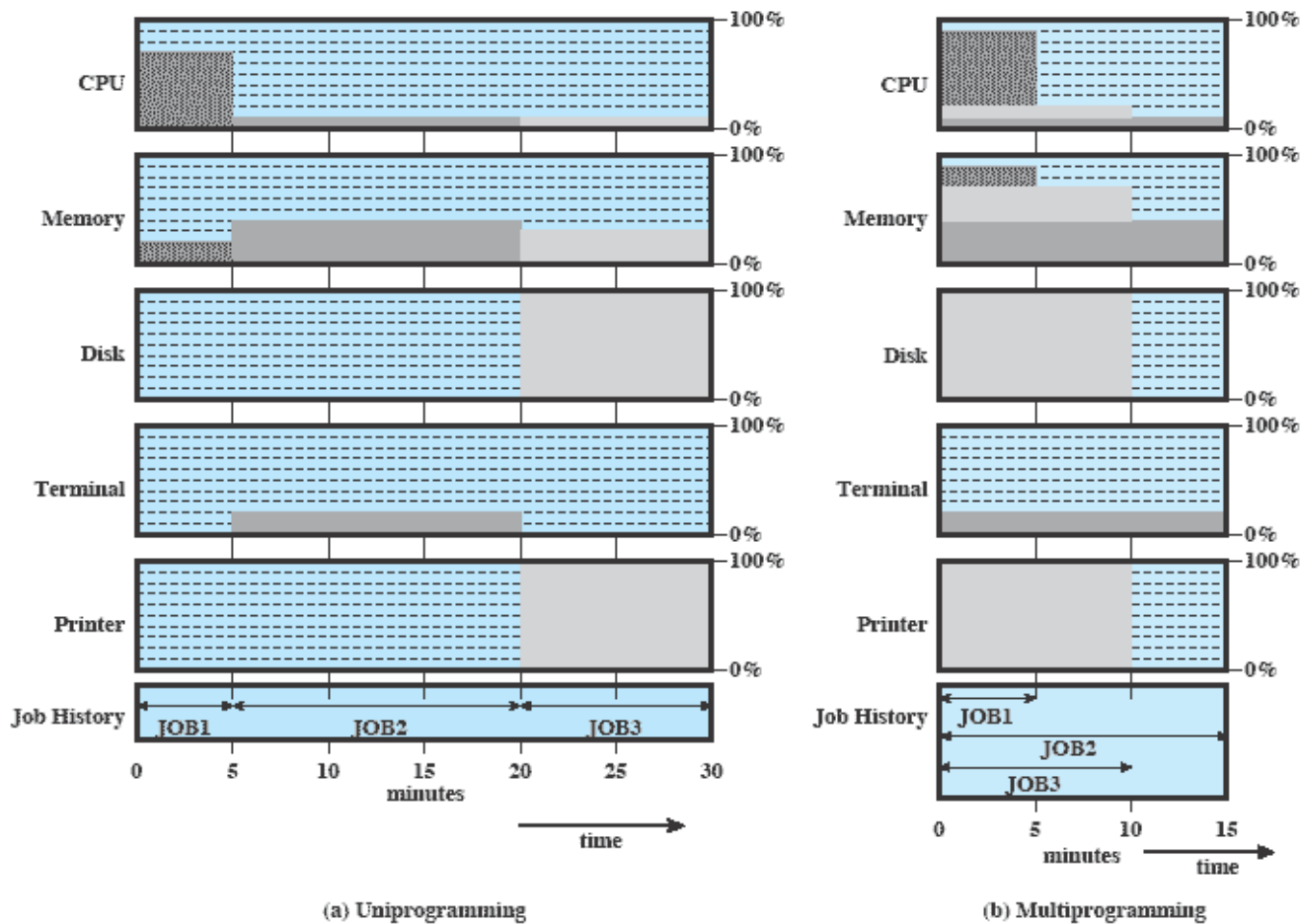|	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Utilization Histograms

Figure 2.6 Utilization Histograms

Effects on Resource Utilization

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Know how to calculate these numbers please

- Reasons for OS to evolve
  - Hardware & Services
    - New | Upgrades | Fixes

Time Sharing Systems

Multiprogrammed Batch Systems

Simple Batch Systems

Serial Processing
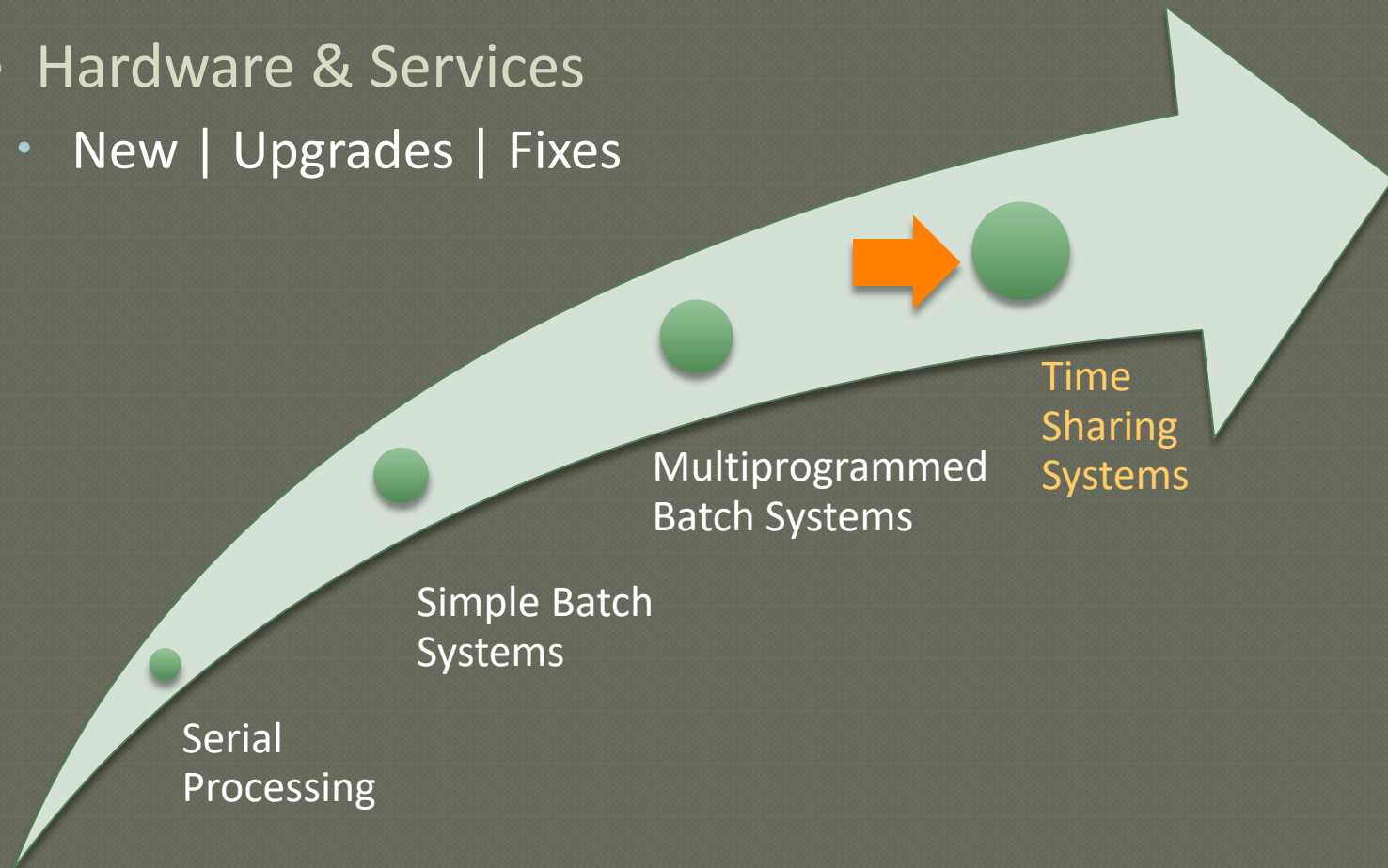
- Time Sharing Systems
  - Users access system simultaneously using terminals
  - Time Slicing
    - Timer generates interrupts every 0.x seconds (small number)
    - OS preempts current program and loads in another
    - Preempted program & data are stored to disk (in old days)
    - *(but keep an eye on swapping overhead!)*

  - Multi-Programming vs. Time sharing

	Batch Multi-programming	**Time sharing**
Objective	Maximize processor use	Minimize response time
Source of instructions	Job Control Language (JCL)	Commands entered in terminal

Chapter 2 Topics

- OS functions
  - Objectives, OS as user/computer interface, OS as resource manager
- OS evolution
  - Serial, Batch, Multi-programming, Time sharing
- Achievements
  - Process, Memory management, Information security, Scheduling, System structure

Achievements

- Major advances in OS development
  - Processes
    - Definition, Errors, Components
  - Memory management
    - OS responsibilities, Virtual memory
  - Information protection & security
  - Scheduling & resource management
  - System structure

Process

- Fundamental to the structure of operating systems

A *process* can be defined as:

a program in execution

an instance of a running program

the entity that can be assigned to, and executed on, a processor

a unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

Development of the Process

- Three major lines of computer system development created problems in timing and synchronization that contributed to the development:

multiprogramming batch operation
- processor is switched among the various programs residing in main memory with a goal of maximum efficiency

time sharing
- be responsive to the individual user but be able to support many users simultaneously

real-time transaction systems
- a number of users are entering queries or updates against a database

Causes of Errors

- **Improper synchronization**
  - a program must wait until the data are available in a buffer
  - improper design of the signaling mechanism can result in loss or duplication

- **Failed mutual exclusion**
  - more than one user or program attempts to make use of a shared resource at the same time

- **Nondeterminate program operation**
  - program execution is interleaved by the processor when memory is shared
 - the order in which programs are scheduled may affect their outcome

- **Deadlocks**
  - it is possible for two or more programs to be hung up waiting for each other
  - may depend on the chance timing of resource allocation and release

Process Management

- Processes (components)
  - Executable code
  - Data
    - e.g., variables, buffers, …
  - Execution context (aka "process state")
    - internal data used by the OS to control the process
      - e.g., registers, priority, whether it is waiting for an I/O event
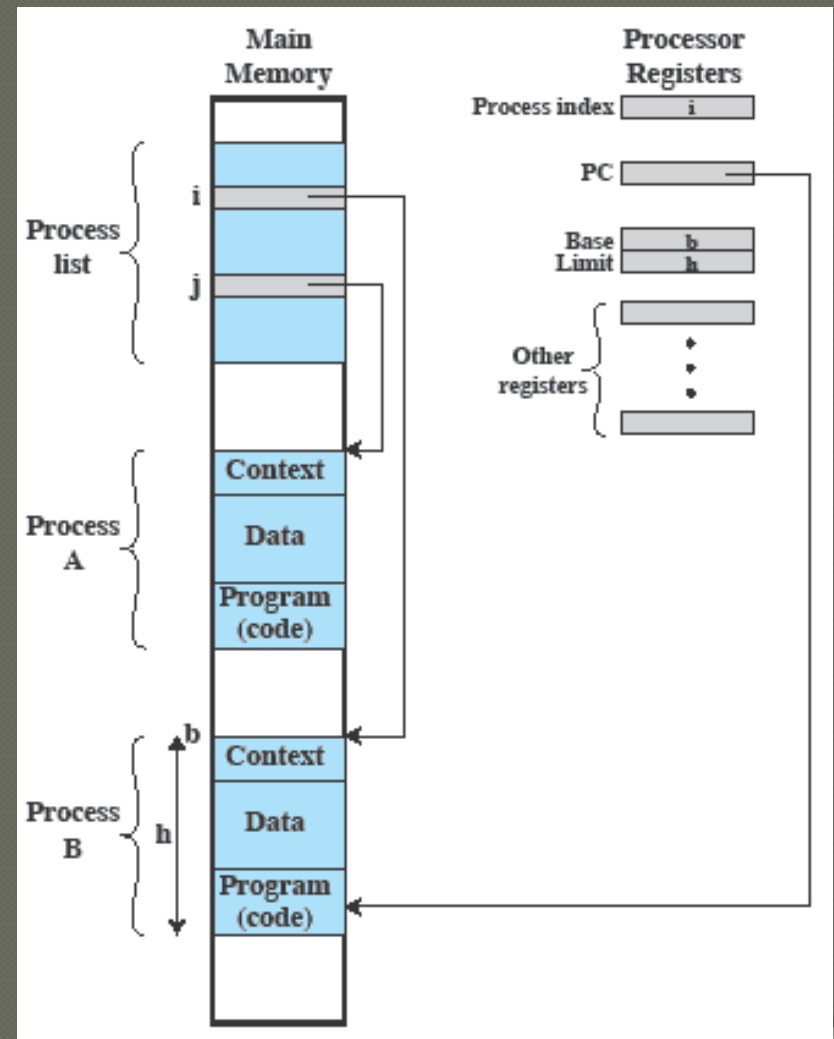
Achievements

- Memory management (OS responsibilities)
  - Process isolation

   …are prevented from interfering with each other
  - Automatic allocation & management
    - …are not concerned about their own allocation
  - Support of modular programming
    - …are able to add/remove modules
  - Protection & access control
    - …are assured the integrity of data in shared memory
  - Long-term storage
    - …are able to store data for later runs (including power down)

How to handle simultaneous processes if they do not fit all in main memory?

Achievements

- Memory management (Virtual Memory)
  - Handling many processes with limited memory
  - Paging
    - Processes are broken into blocks (aka pages)
      - Pages can be anywhere in main memory
    - CPU uses virtual addresses to find instructions/data
      - Addresses are page number + offset within page

Main Memory

Disk
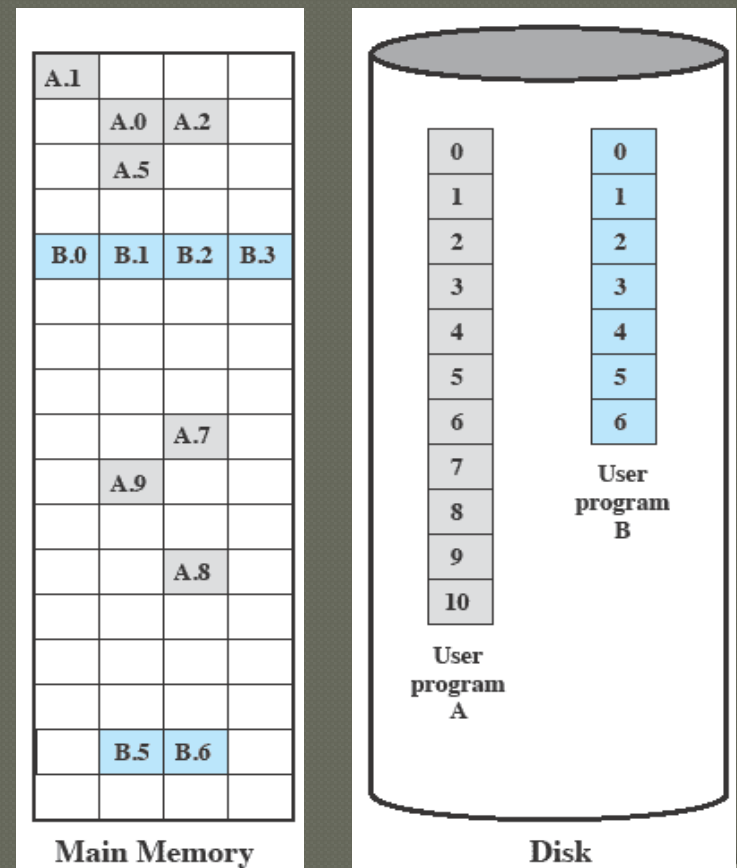
Achievements

- Information protection & security
  - controlling access to processes & data
  - Availability
    - Protection against interruption
  - Confidentiality
    - Protection against unauthorized access
  - Data integrity
    - Protection against unauthorized modification
  - Authenticity
    - Protection against misrepresentation & data validation

Achievements

- Scheduling & resource management
  - OS manages resources (main memory, I/O devices, processors) and schedules their use by processes
  - Fairness
    - Equal processes given equal and fair access to resources.
  - Differential responsiveness
    - Different processes treated differently according to their needs.
  - Efficiency
    - Overall performance is a goal
      - maximize throughput
      - minimize response time
      - accommodate as many users as possible

These criteria conflict (what's the right balance?)
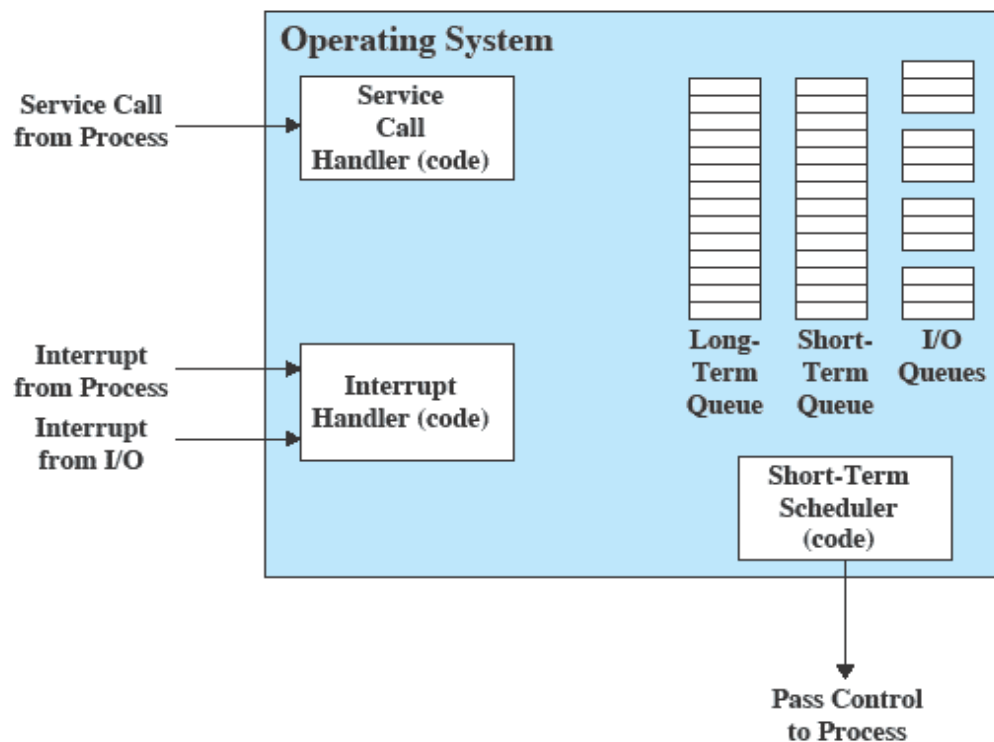
Key Elements of Schedule and Dispatch

Figure 2.11 Key Elements of an Operating System for Multiprogramming

Achievements

- System structure
  - Up to now
    - OS are monolithic programs
    - processes are linearly executed

    What to do about it?

  - Microkernel Architecture
    - Keep essential functions in kernel
      - memory addressing, inter-process communication (IPC), scheduling
    - Modularize the rest (towards object-oriented approach)
      - modules dynamically linked, easier to replace
  - Advantages
    - Flexibility: low coupling/high cohesion (props up distributed OS – illusion of unified memory & resources)

Achievements

- System structure
  - Up to now
    - OS are monolithic programs
    - processes are linearly executed   *What to do about it?*
  - Symmetric multiprocessing (add CPUs)
    - 2+ CPU run in parallel (hardware + OS exploiting it)
    - Processes scheduled to separate CPU (but share resources)
  - Multi-threading (divide processes)
    - Process broken into parts that run concurrently (own thread)
    - Process = ∑ (threads = concurrent unit of work)
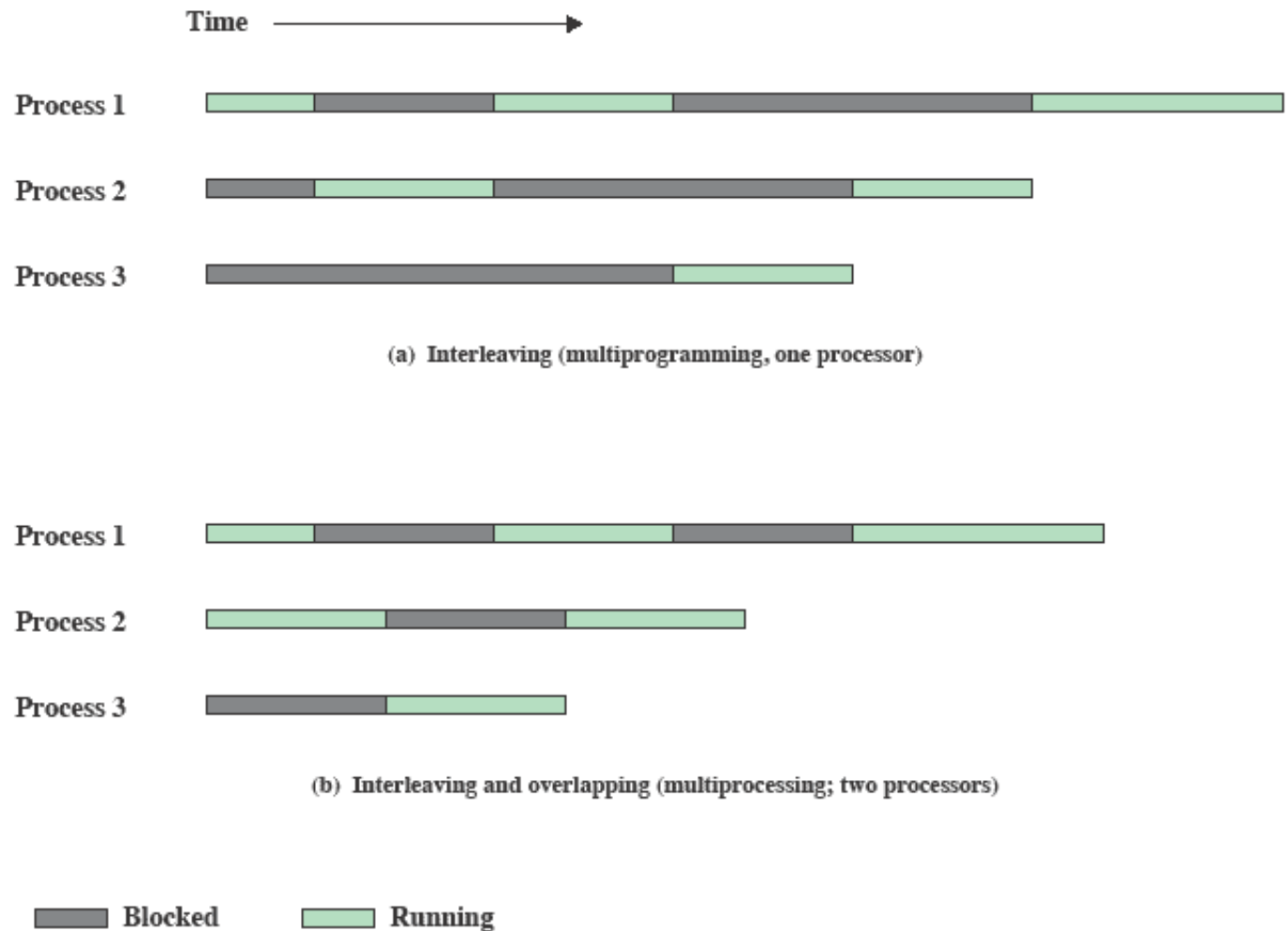   - Programmers control scope & timing of concurrency

Figure 2.12 Multiprogramming and Multiprocessing

Symmetric multiprocessing

Challenges
- Kernel concurrency: Kernel processes allow concurrent CPU access (state integrity)
- Scheduling: Scheduling across CPUs must be coordinated (avoid duplicated runs)
- Synchronization: Access to resources must be synchronized (use locks)
- Memory management: Page reuse (coordinating page replacements)
- Fault tolerance: Graceful degradation

Parallelism opportunities
- Multiprogramming & multi-threading in each processor
- A process could have its threads executed in different CPUs

Processes scheduled to separate CPU (but share resources)

- Multi-threading (divide processes)
  - Process broken into parts that run concurrently (own thread)
  - Process = ∑ (threads = concurrent unit of work)
  - Programmers control scope & timing of concurrency

- OS functions
  - Objectives, OS as user/computer interface, OS as resource manager
- OS evolution
  - Serial, Batch, Multi~~programming~~, Time sharing
- Achievements
  - Process, Memory management, Information security, Scheduling, System structure

Done!