# Chapter 3
# Process Description and Control

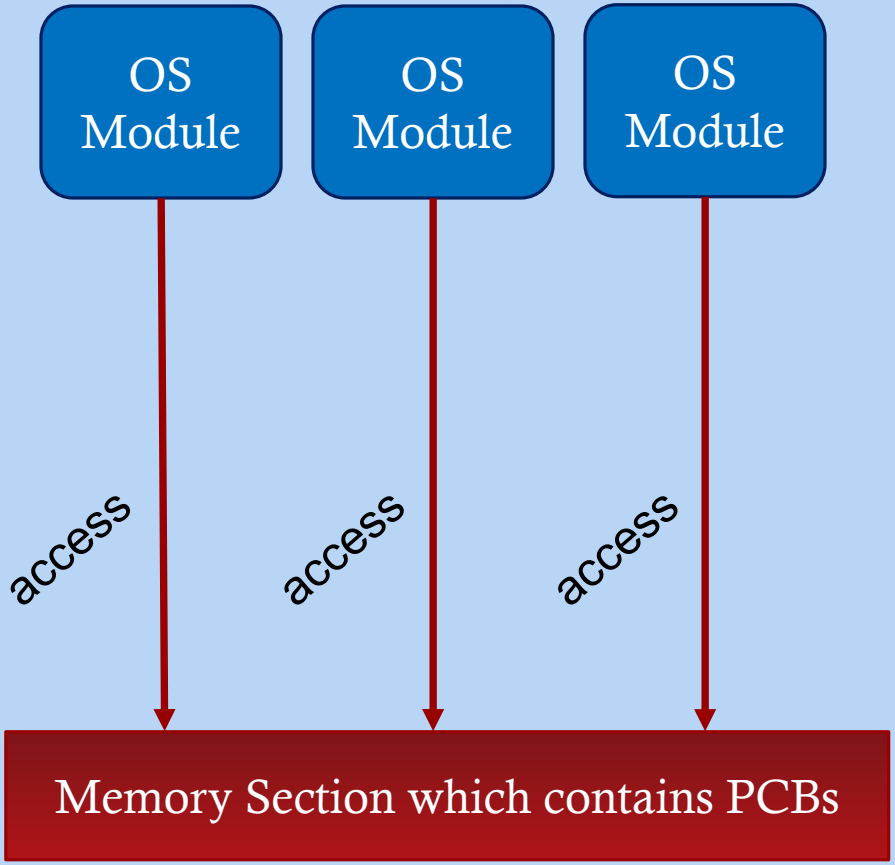
**Part B**: Section 3.3

# Role of the Process Control Block

- The most important **data structure** in an OS
  - Contains all of the information about a process that is needed by the OS
  - PCBs are read and/or modified by almost all modules of the OS
  - The **set of all PCBs** (for the user-processes as well as the PCBs for the Operating System's own processes) **captures the entire status of the OS** ("*system snapshot*") **at any given point in time.**
- Difficulty is not access, but **protection**
  - **Correctness**: A bug in a single routine could damage process control blocks, which could destroy the system's ability to manage the affected processes
  - **Software Engineering**: A design change in the data structure of the PCB (new version of OS) could affect many modules in the OS's software architecture.

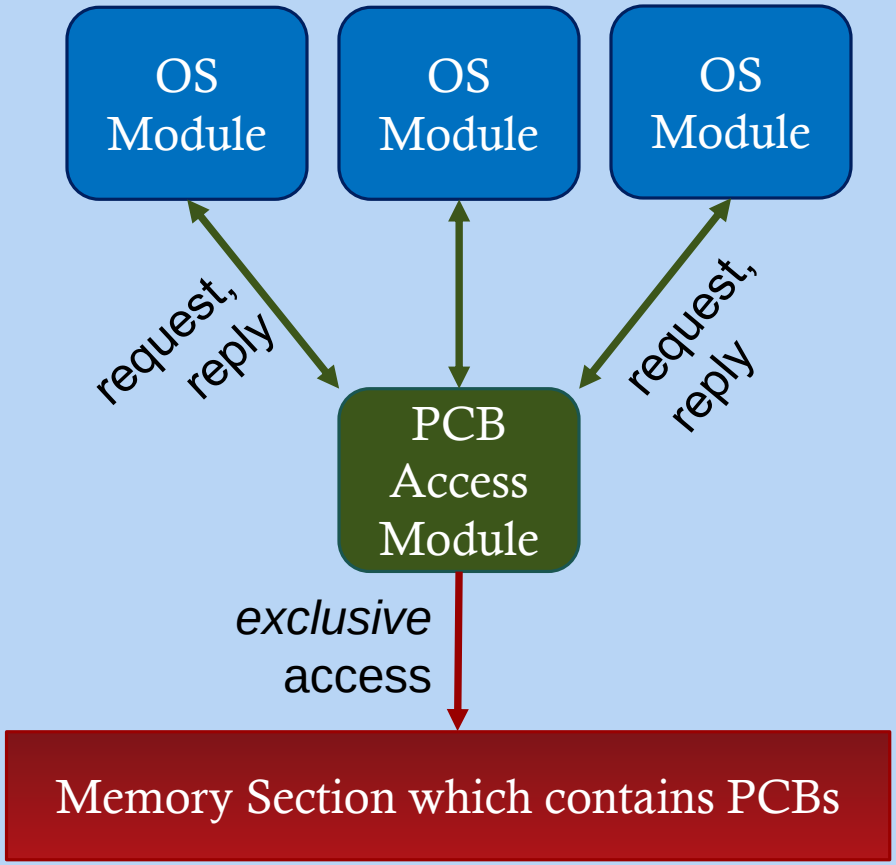
**Unsafe OS Design:**

* Many OS Modules can possibly damage the PCBs directly.
* Modification of PCBs affects many modules.




**Safe OS Design:**

* PCB Access Module acts as a mini-**"shield" inside** the **OS**'s architecture.
* However, *slower access-time to the PCBs is the price to be paid* for this improved safety.




These kinds of Design Considerations are studied in COS**301**(*Software Engineering*)

# Resource Allocation Diagram

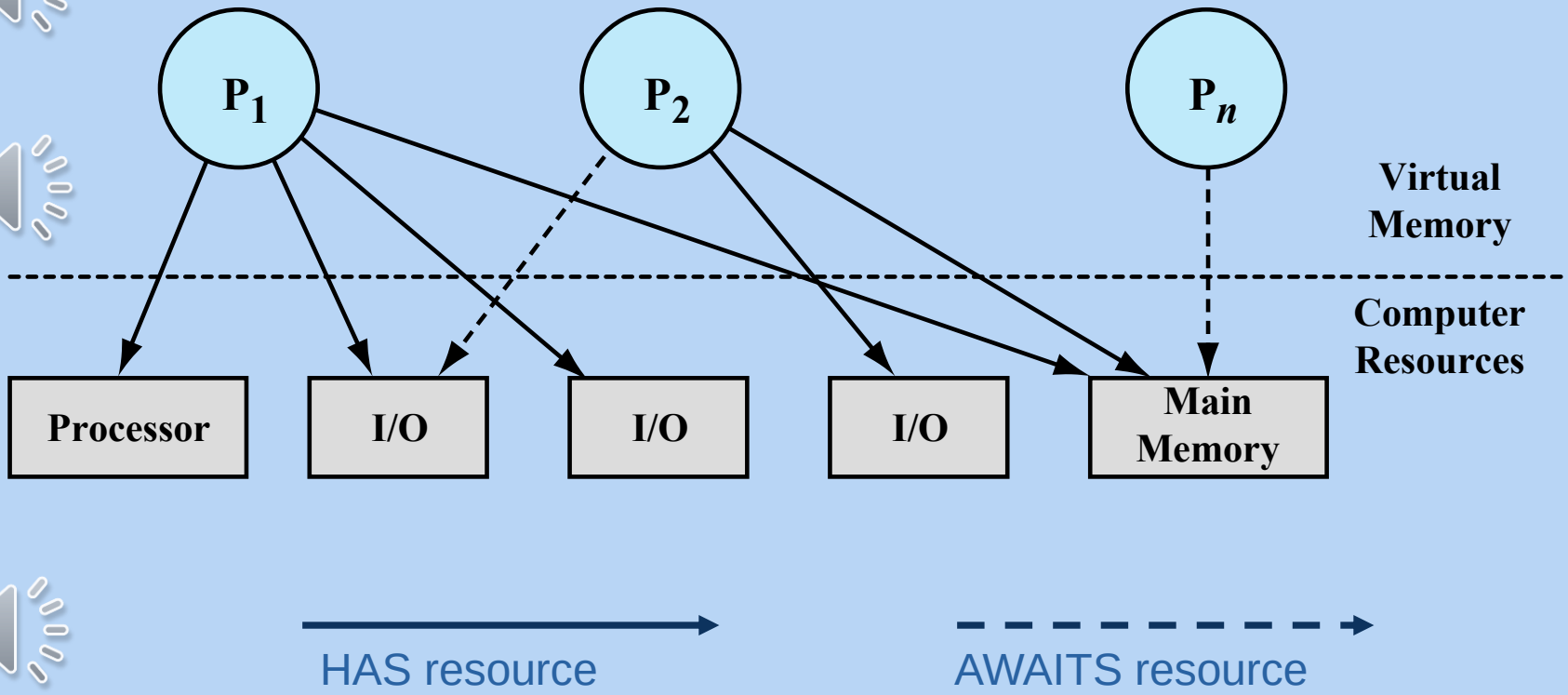



**Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)**

# Internal **Implementation** of the *Resource Allocation Diagram* by **Data Structures**

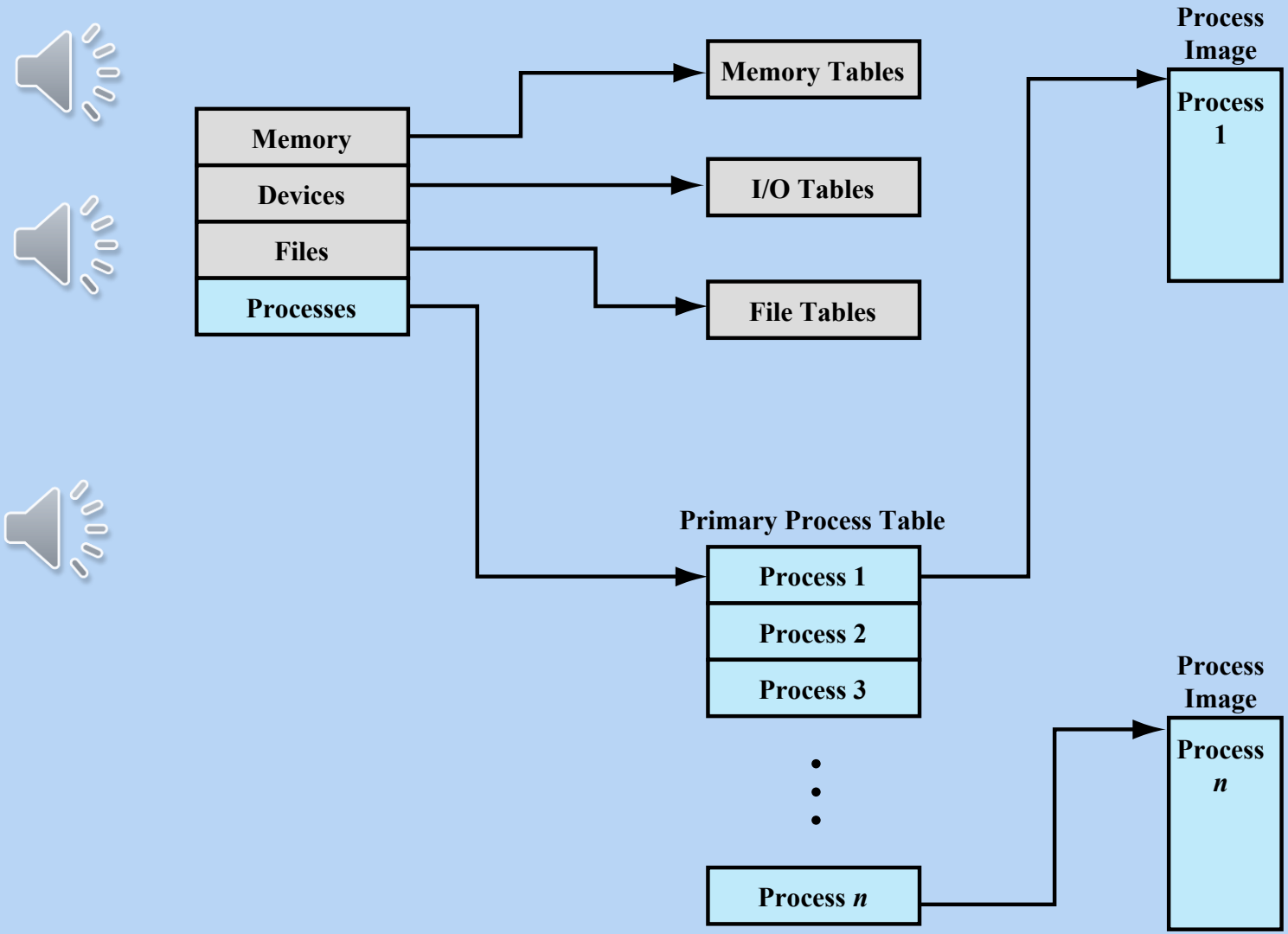
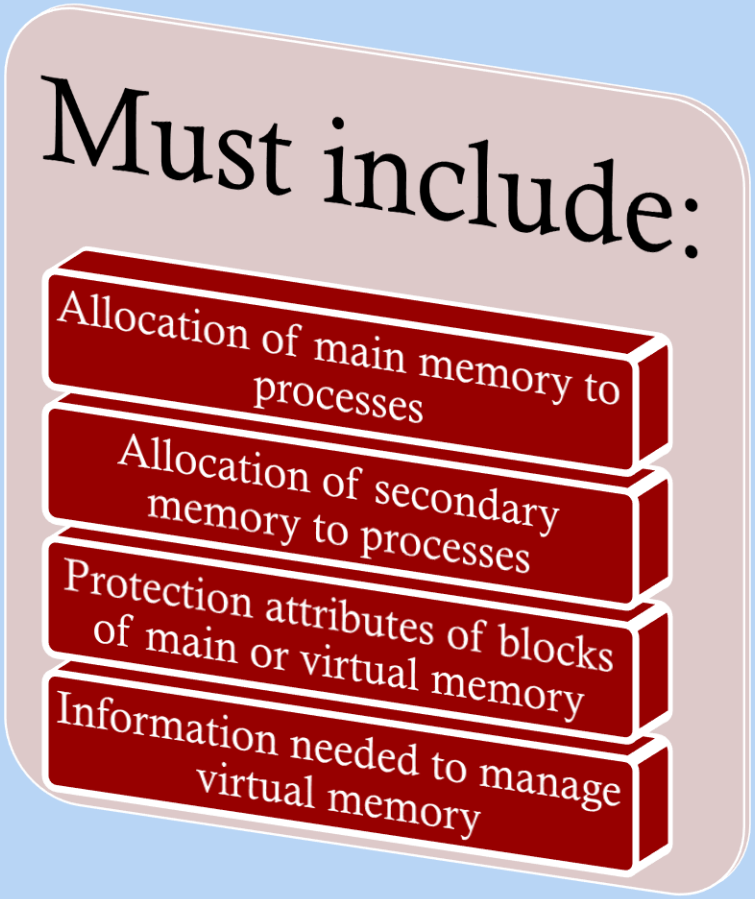



**Figure 3.11 General Structure of Operating System Control Tables**

# Memory Tables

- Used to keep track of both main (real) and secondary (virtual) memory
- Processes are maintained on secondary memory using some sort of virtual memory or simple swapping mechanism




➔ Details: **Chapters 7--8**

# I/O Tables

- Used by the OS to manage the I/O devices and channels of the computer system
- At any given time, an I/O device may be available or assigned to a particular process

If an I/O operation is in progress, the OS needs to know:

- The status of the I/O operation
- The location in main memory being used as the source or destination of the I/O transfer

➔ Details: **Chapter 11**

# File Tables

These tables provide information about:

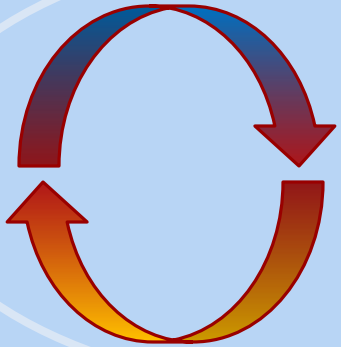
- Existence of files
- Location on secondary memory
- Current status
- Other attributes

- Information may be maintained and used by a file management system
  - In which case the OS has little or no knowledge of files
- In other operating systems, much of the detail of file management is managed by the OS itself

➔ Details: **Chapter 12**

# Process Tables

- Must be maintained to manage processes
- There must be some reference to memory, I/O, and files Tables, directly or indirectly
- The Tables themselves must be accessible by the OS and –because they are somewhere stored– are also subject to memory management



**Processes and Process Tables need memory**

Activity of memory **Management** ***is also*** **a process**

# Process Control Structures

To manage and control a process the OS must know:

- Where the process is located
- The attributes of the process that are necessary for its management

# Process Control Structures

## Process Location

- A process must include a program (or set of programs) to be executed
- A process will consist of at least sufficient memory to hold the programs and data of that process
- The execution of a program typically involves a **dynamic stack** that is used to keep track of procedure calls and parameter passing between procedures:
  ➔ **see COS341 in Study-Year 3**

## Process Attributes

- Each process has associated with it a number of attributes that are used by the OS for process control
- **The collection of program, data, stack, and attributes is referred to as the process image**
- Process image location will depend on the memory management scheme being implemented
- The **Process Tables must also indicate (*point to*) these locations: ➔ *Cross-reference to the Memory Tables***

# Table 3.4
# Typical Elements of a Process Image

**User Data**

The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.

**User Program**

The program to be executed.

**Stack**

Each process has one or more last-in-first-out (LIFO) stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.

**Process Control Block**

Data needed by the OS to control the process (see Table 3.5).

# Appreciation of the Run-Time-Stack

```
PROCEDURE: int Factorial(int n)
{
 IF (n < 0) THEN return (–1) // error: illegal input
 IF (n = 0) THEN return ( 1 )
 IF (n > 0) THEN return ( Factorial(n–1) * n ) // recursion
}
```

**Computer Science** Students: ➔ See **Algorithm 8.7** in Section 8.7 of the **COS151** introduction book

call **Factorial**(**42**)

Before the **final** result of the computation can be returned, the procedure recursively calls itself many times, whereby each call yields a preliminary intermediate result. These **intermediate results are stored in the run-time-stack, which keeps growing dynamically** while the procedure is running.

# Process Identification

- **Each process is assigned a unique numeric identifier.**
- Many of the tables controlled by the OS may use process identifiers ("***keys***") to **cross-reference process tables** ➔ See for comparison the **Database lectures** in the **2nd Year** of the Comp. Science curriculum.
- **Memory Tables** may be organized to provide a map of main memory **with an indication of which process is assigned to each region**
  - *Similar references appear in I/O tables and file tables*
- **When processes communicate with one another, the process identifier informs the OS of the destination of a particular communication**
- When processes are allowed to create other processes by spawning, identifiers indicate the parent and children of each process

# CPU State Information

Consists of the contents of processor registers

- User-visible registers
- Control and status registers
- Stack pointers

**Program status word (PSW)**

- Contains condition codes plus other status information
- EFLAGS register is an example of a PSW used by any OS running on an x86 processor

# Additional Process Control Information

- Various further information needed by the OS to control and coordinate the various active processes
- **Stored inside the PCB**

See Table 3.5 in the book

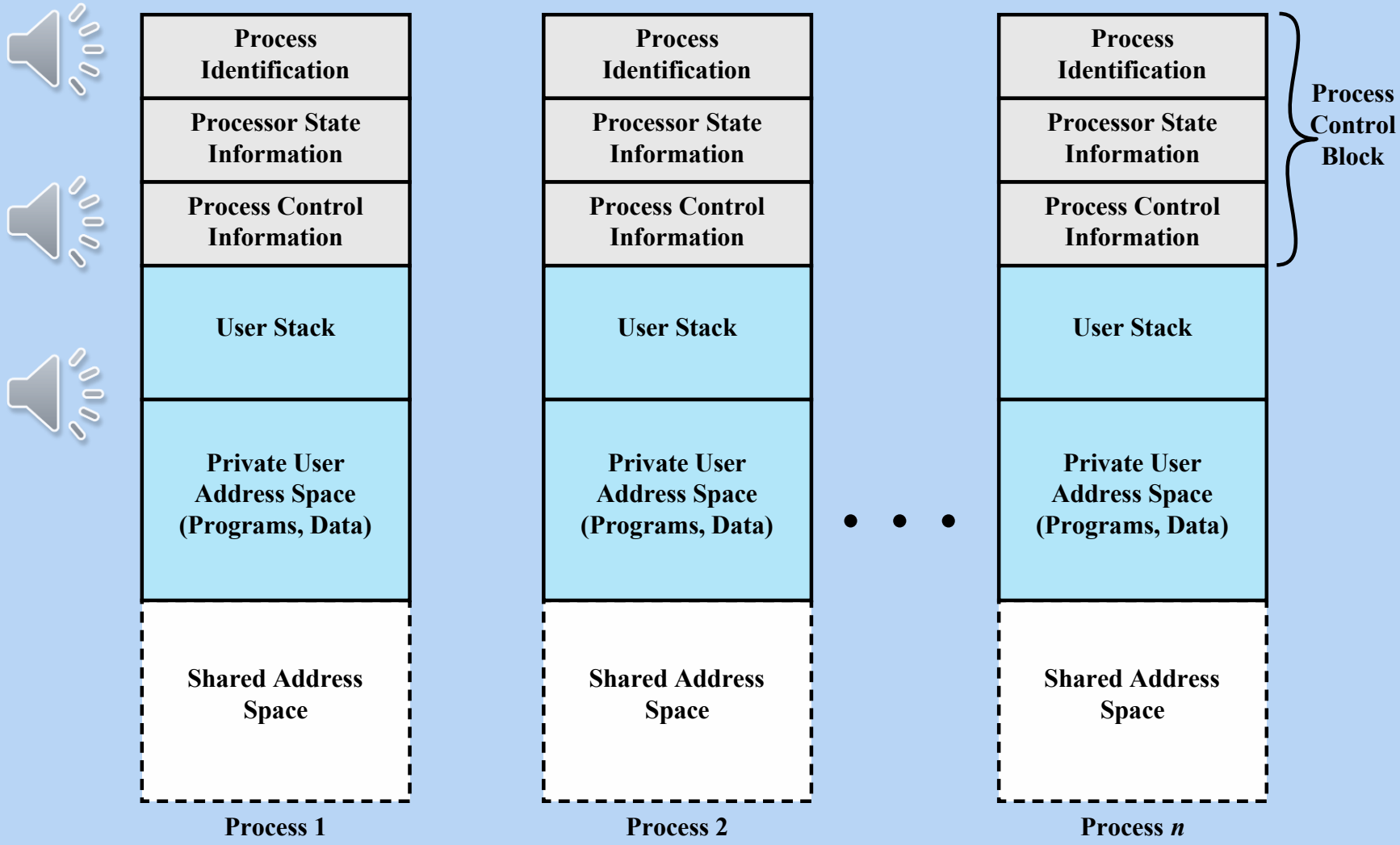



**Figure 3.13 User Processes in Virtual Memory**

**Computer Science Students:**
For further details concerning the **Implementation of Linked Lists**, see Section **11.3** of the **COS151 Introduction** Book

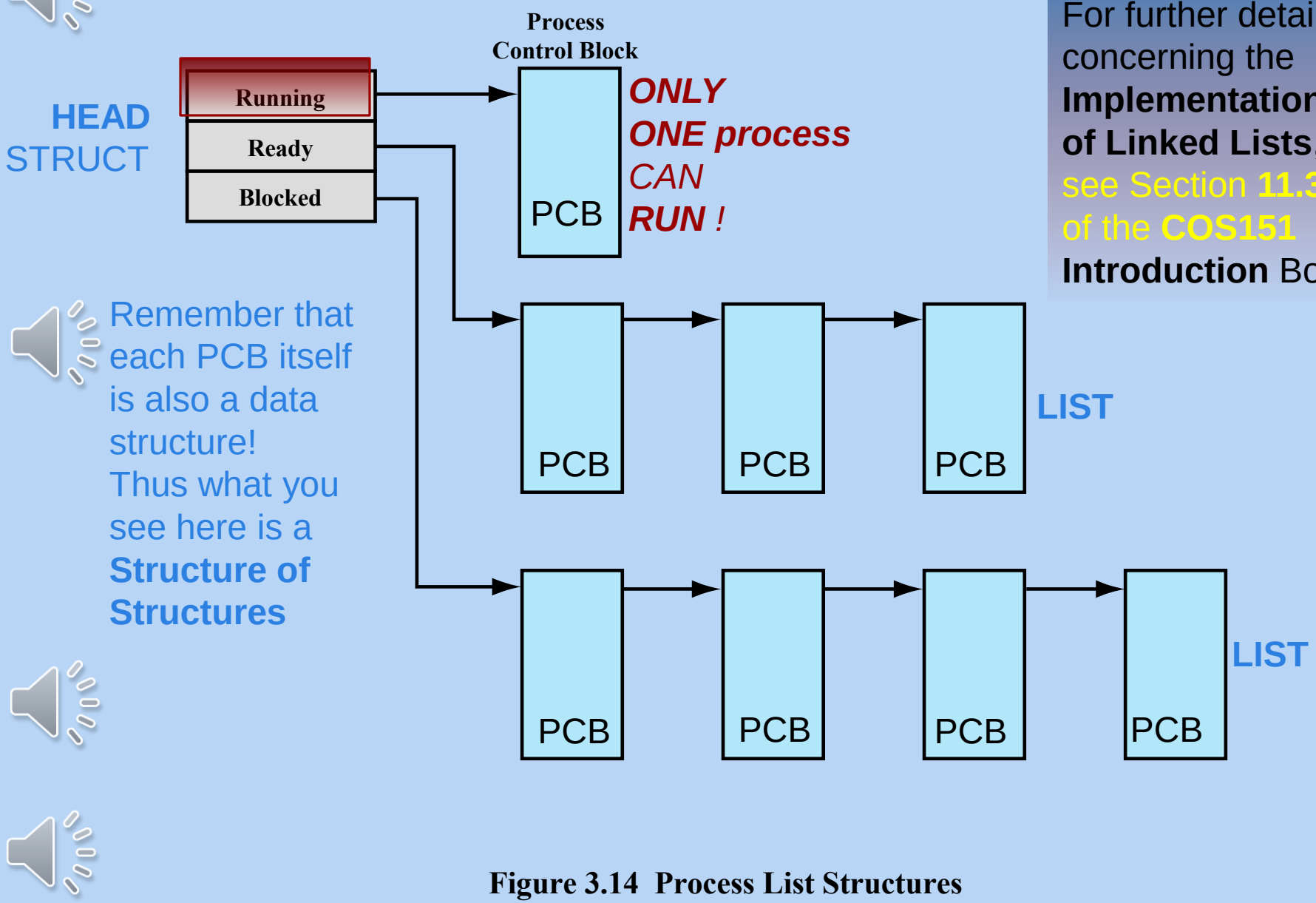



**Figure 3.14 Process List Structures**