Introduction to Process in Computing Systems

Programs and Processes

- One way to describe the hardware of a computer system is to say that it provides a framework for executing programs and storing files.
- A *file* is a collection of data that is usually stored on disk, although some files are stored on tape.
  - UNIX treats peripherals as special files, so that terminals, printers, and other devices are accessible in the same way as disk-based files.
- A *program* is a collection of bytes representing code and data that are stored in a file.
- When a program is started, it is loaded from disk into the main memory (RAM).
- When a program is running, it is called a *process.* Most processes read and write data from files.

Processes

- The concept of process
  - A *program in execution*
  - The “animated spirit” of a program
  - The entity that can be assigned to and executed on a processor
- The *life of a process* is bounded by its creation and termination.
- In a typical computer system, it may be running thousands of processes at the same time.

Processes

- In order to be executed by the CPU, the code and data of a process must be located in the main memory.

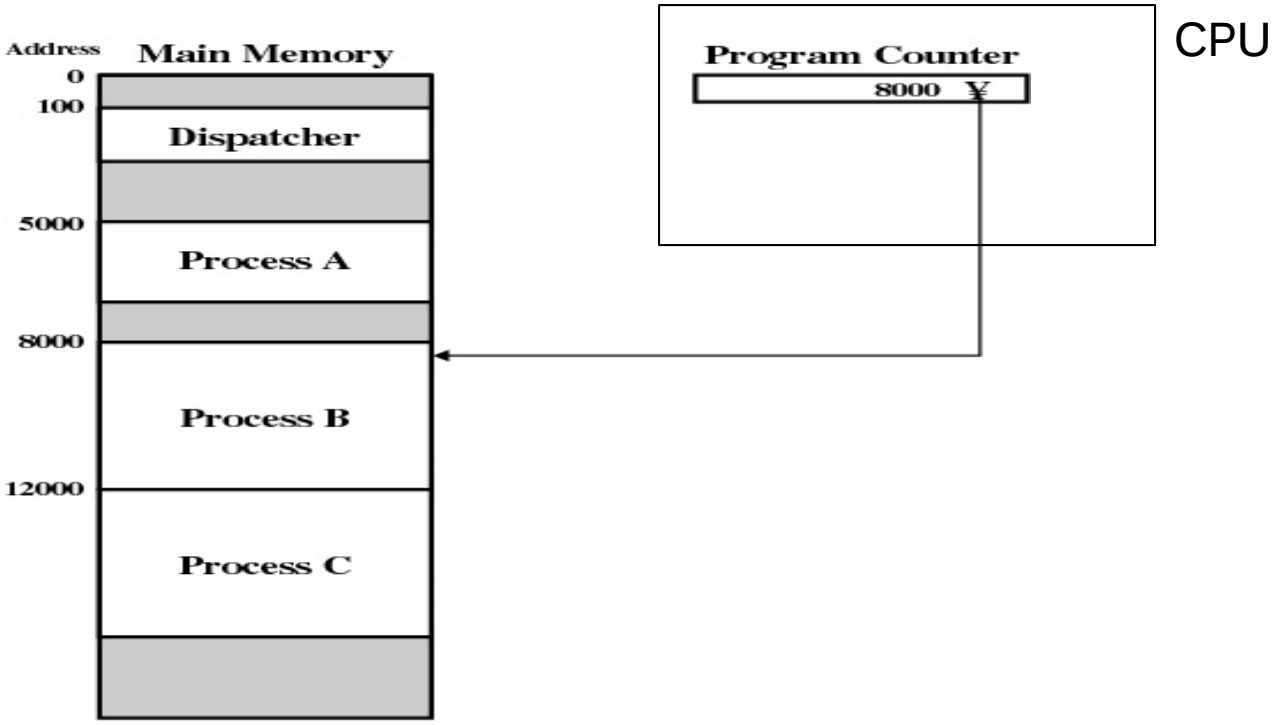


Figure 3.1 Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13

Show Processes in Unix

- The Unix command “ps” can display the current processes in the computer.

```
cuse93:> ps -a                    ... display all processes
PID   TT       S  TIME  COMMAND
360   console  S  0:00  /usr/lib/saf/ttymon
2434 pts/2     O  0:00  -tcsh
2431 pts/2     O  0:00  ps –a

cuse93:> ps -au                   ... display all processes with user info
USER   PID %CPU %MEM  SZ   RSS    TT       S  START     TIME  COMMAND
root   360   0.0   0.1  2987 1504  console  S  Feb 17     0:00  /usr/lib….
wlam  2434  0.2   0.1  3302  2354  pts/2    O 17:34:21 0:00  -tcsh
wlam  2431  0.1   0.2  3421 1456  pts/2    O 17:34:21 0:00  ps -au
```

Major Components of Operating Systems (OS)

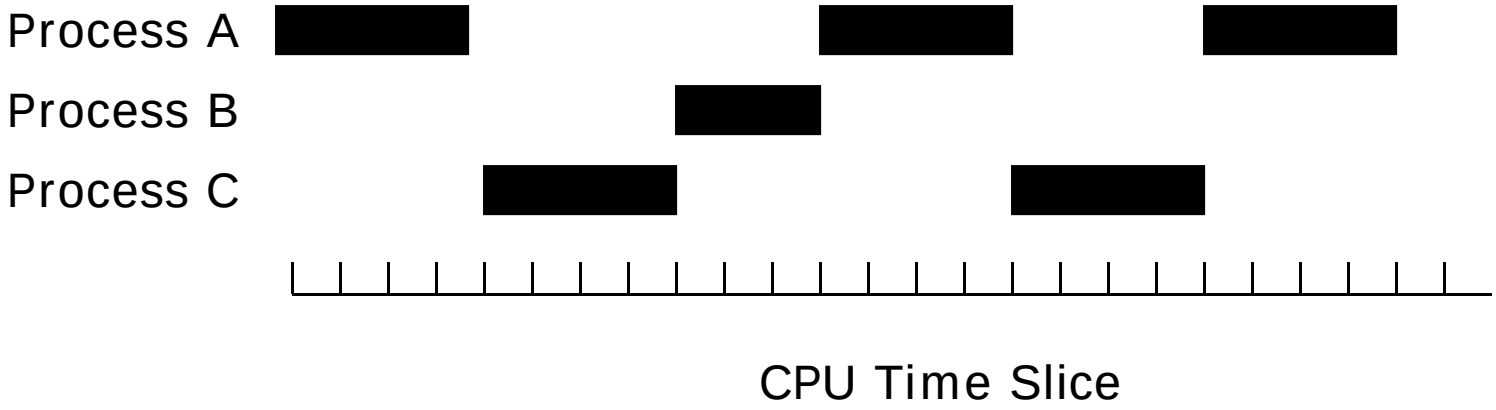
- Process management
- Resource management
  - CPU
  - Memory
  - Device
- File system

Process Management

- One major operating system (OS) function is the sharing of limited resources among competing processes.
  - Limited resources in a typical computer system include CPUs, memory, disk space, and peripherals such as printers.
- OS shares *CPUs* among processes
  - By dividing each second of CPU time into equal-sized “slices” (typically 1/10 second) and then allocating them to processes on the basis of a priority scheme.

Process Management

- The operating system usually interleaves the execution of all processes to maximize processor utilization while providing reasonable response time.



- One component in the operating system is called *dispatcher* which takes care of the interleaving of the execution of the processes.

Trace of a Process (cont'd)

(a) Trace of process A	(b) Trace of process B	(c) Trace of process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004	8004 (wait for input e.g. keyboard)	12004
5005	:	12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011
:		:

5000 = Starting address of program of process A
8000 = Starting address of program of process B
12000 = Starting address of program of process C

Figure 3.2 Traces of Processes of Figure 3.1

Interrupts

- A mechanism by which different processes may interrupt the normal processing of the processor.
- The classes of Interrupts
  - *Timer*: Generated by a timer within the processor.
  - *Input/Output (I/O)*: Generated by an input/output device.
- Interrupt is an interruption of the normal sequence of execution of the running process.
- After interrupt is completed, the normal program execution is resumed.
- Interrupts are provided primarily to improve processing efficiency.
  - I/O time is substantially slower than CPU processing
  - Avoid CPU waiting for slow I/O devices

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Time out
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
----------	Time out
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Time out
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Time out

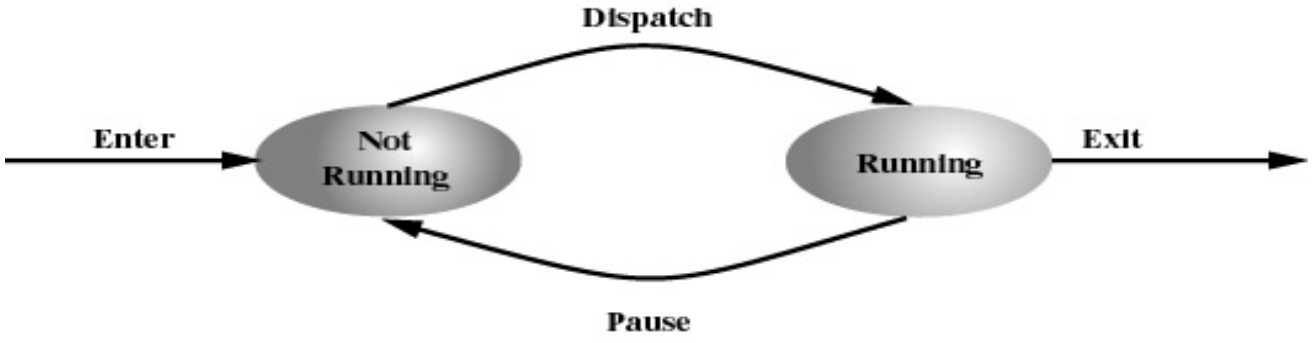
100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

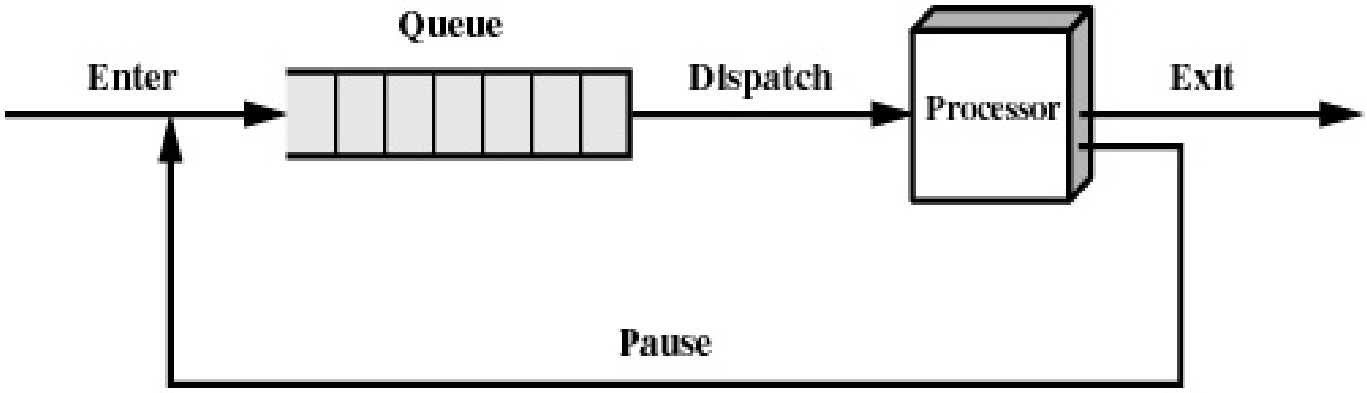
Figure 3.3 Combined Trace of Processes of Figure 3.1

A Two-State Process Model

- A *process model* is used to describe the behavior what we want the processes to exhibit.



(a) State transition diagram



(b) Queuing diagram

Two-State Process Model

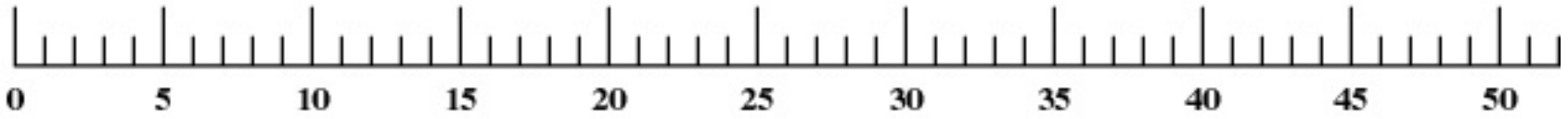
A Two-State Process Model (cont'd)

- Each process must be represented in some way so that OS can keep track of it.
  - current state
  - location in memory
- Each queue item may be:
  - a pointer to a particular process
  - a data block representing a process
- *Process creation* (*Process Spawning*):
  - OS may create a process by itself
    e.g. a user logs on to the system
  - OS may create a process on behalf of an application
    e.g. requests a file to be printed
  - Application may create process
    e.g. a Web browser creates a process to download a file
- *Process termination: T*ermination request by application, error and fault conditions.

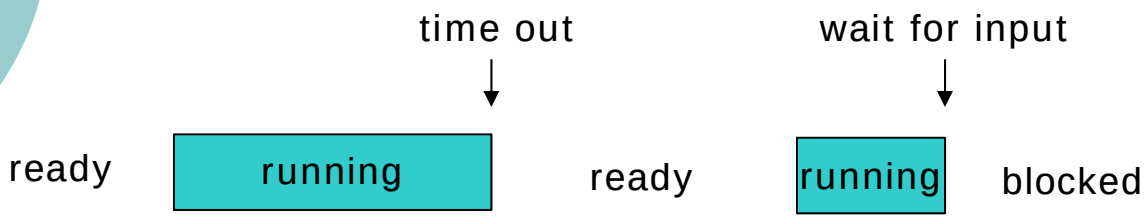
A Five-State Process Model

ready

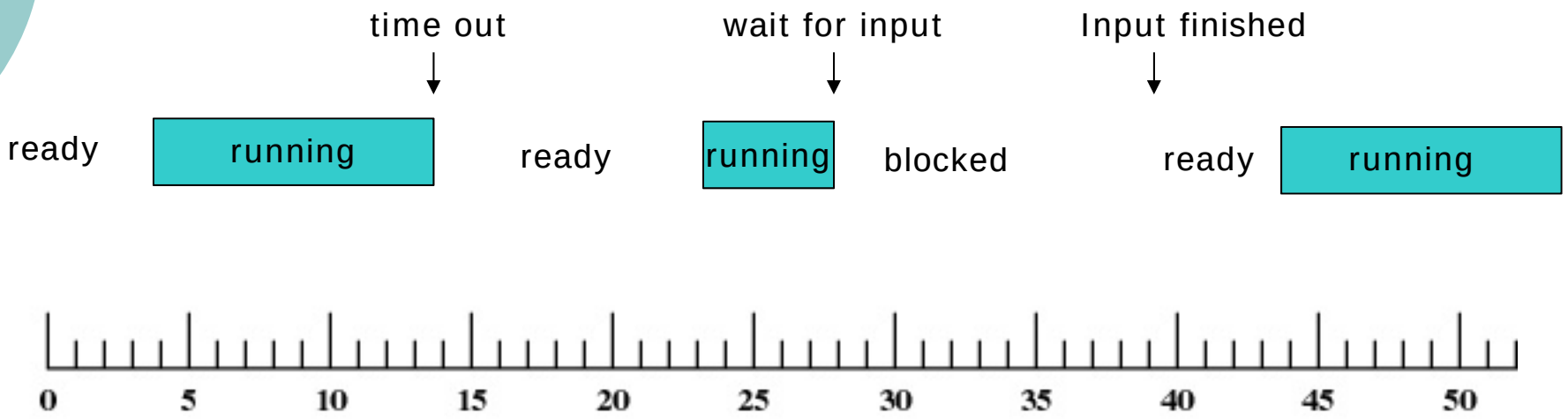
running



A Five-State Process Model



A Five-State Process Model



A Five-State Process Model

- The 2-state model does not consider that some processes in the queue may *not* be ready to execute e.g. waiting for I/O, blocked
- I/O operation is much slower than CPU computation

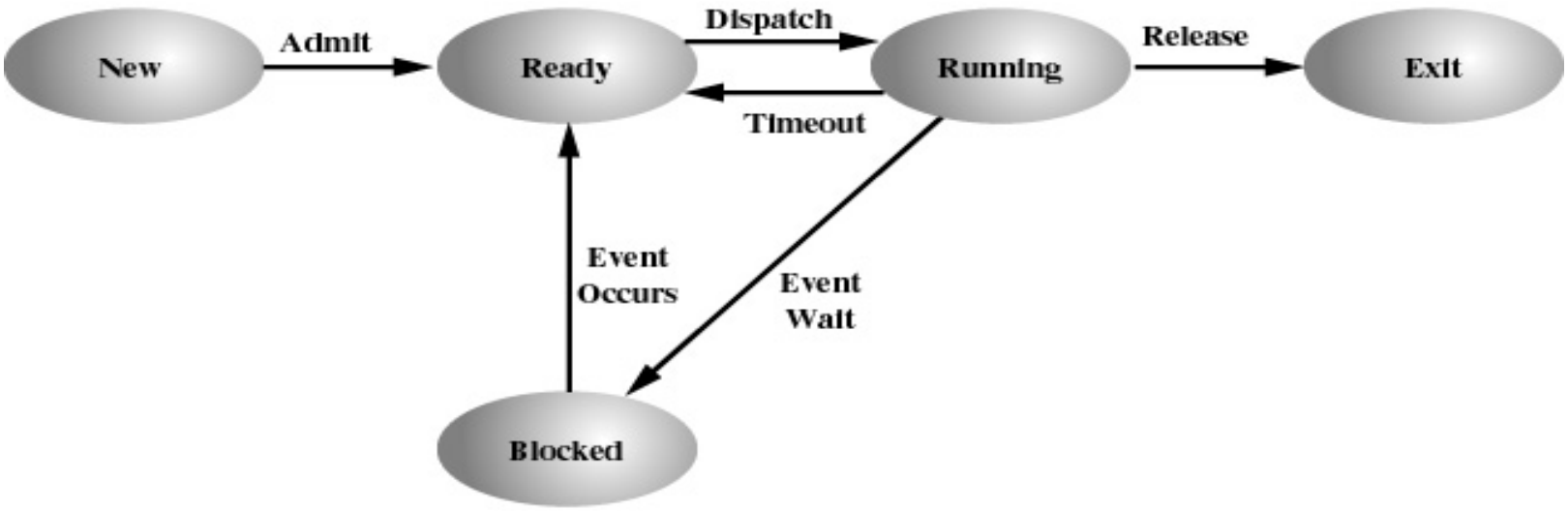


Figure 3.5 Five-State Process Model

A Five-State Process Model (cont'd)

- *Running*: The process is currently being executed
- *Ready*: The process is prepared to execute when given the opportunity
- *Blocked*: The process cannot execute until some event (e.g. I/O read/write) occurs
- *New*: The process has just been created but has not yet been admitted to the pool of executable processes by the OS.
- *Exit*: The process has been released from the pool of executable processes by the OS.

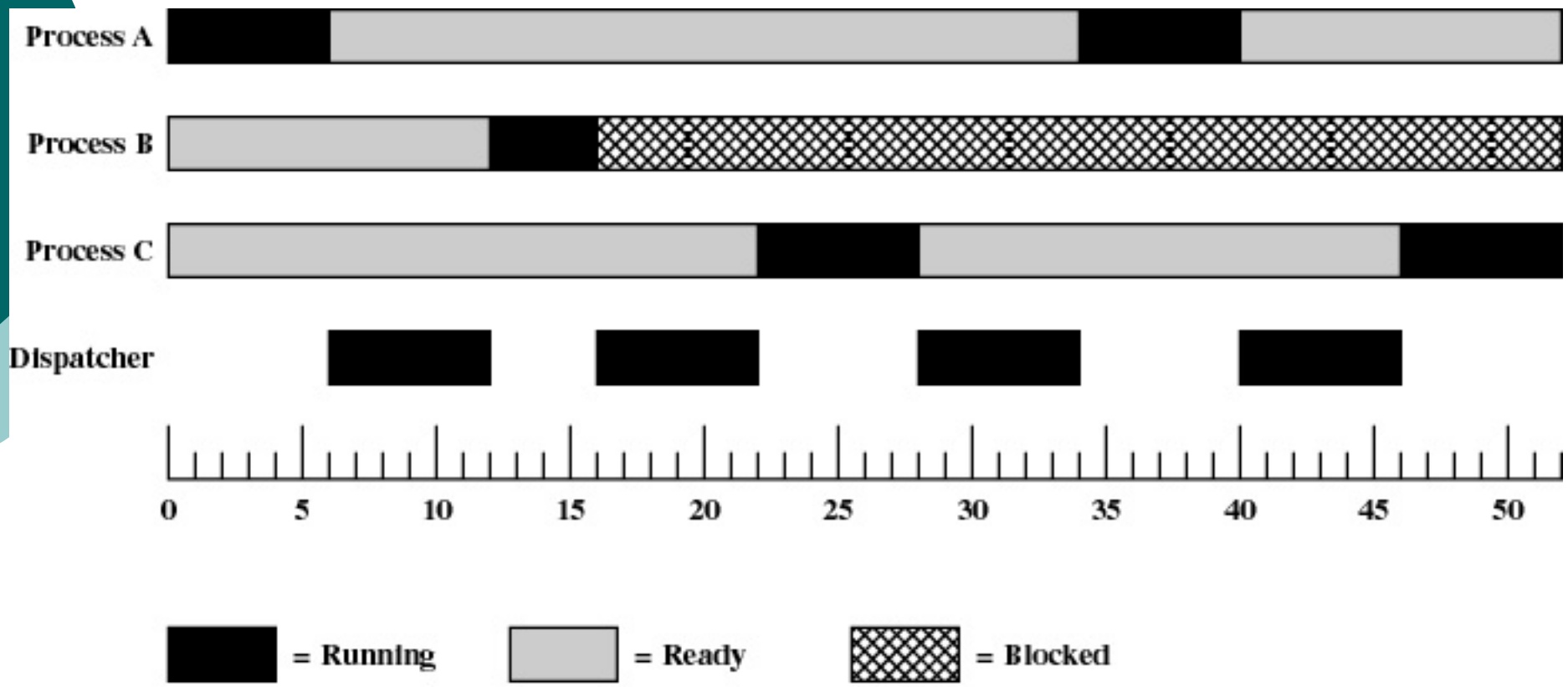
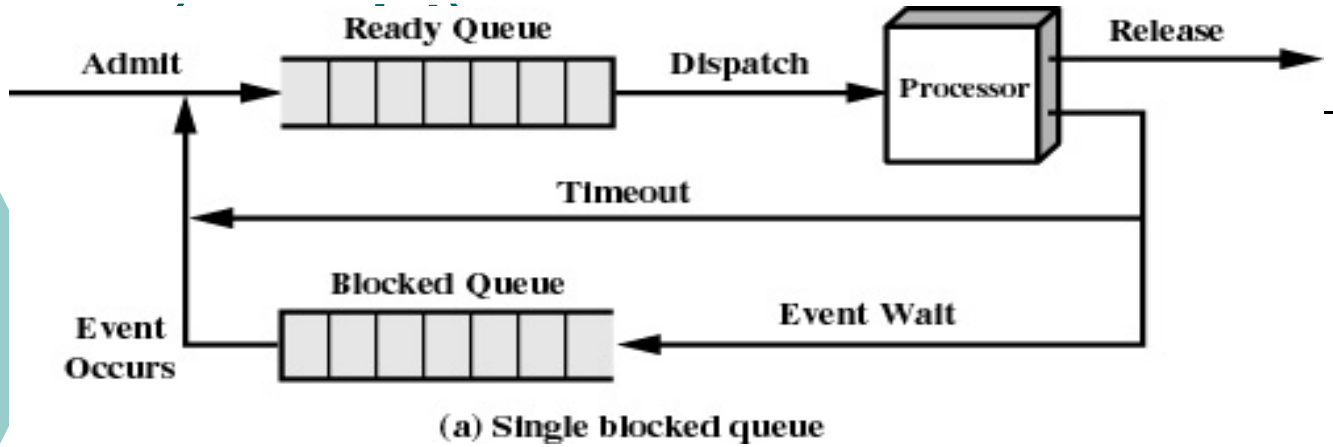
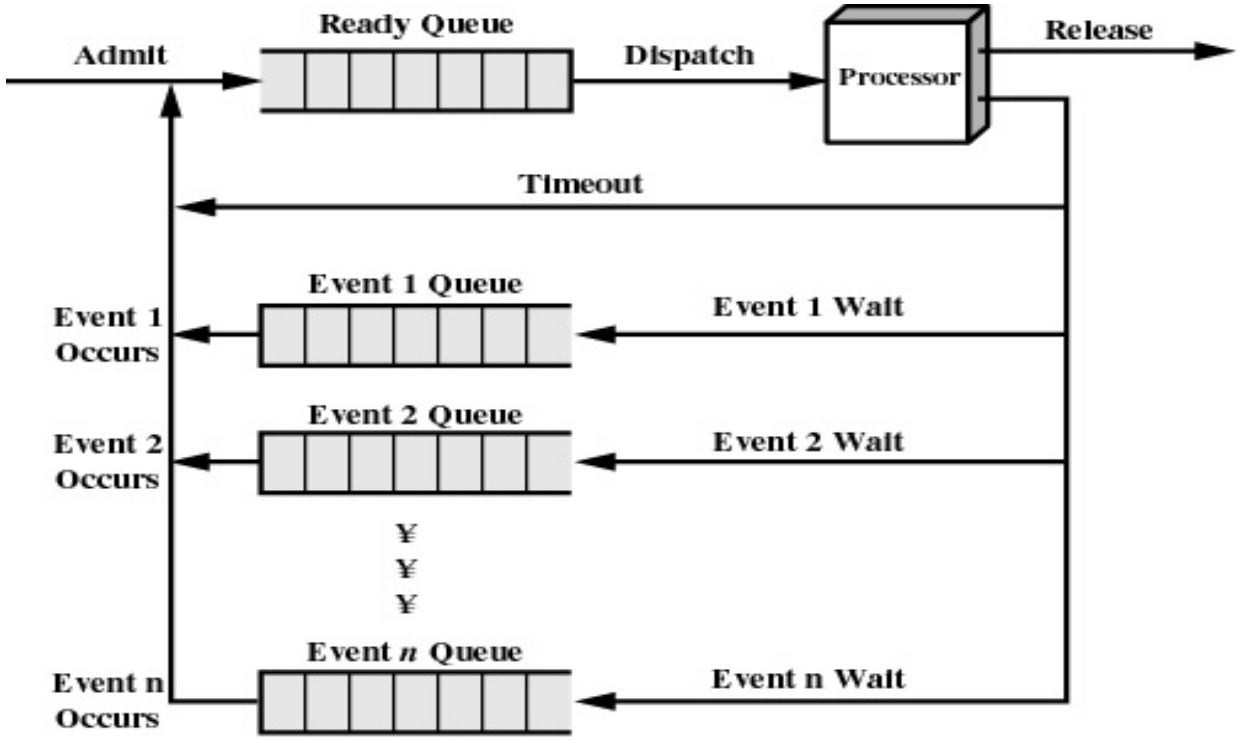


Figure 3.6 Process States for Trace of Figure 3.3

A Five-State Process Model



(a) Single blocked queue



(b) Multiple blocked queues

Queuing Model

Processes and Resources

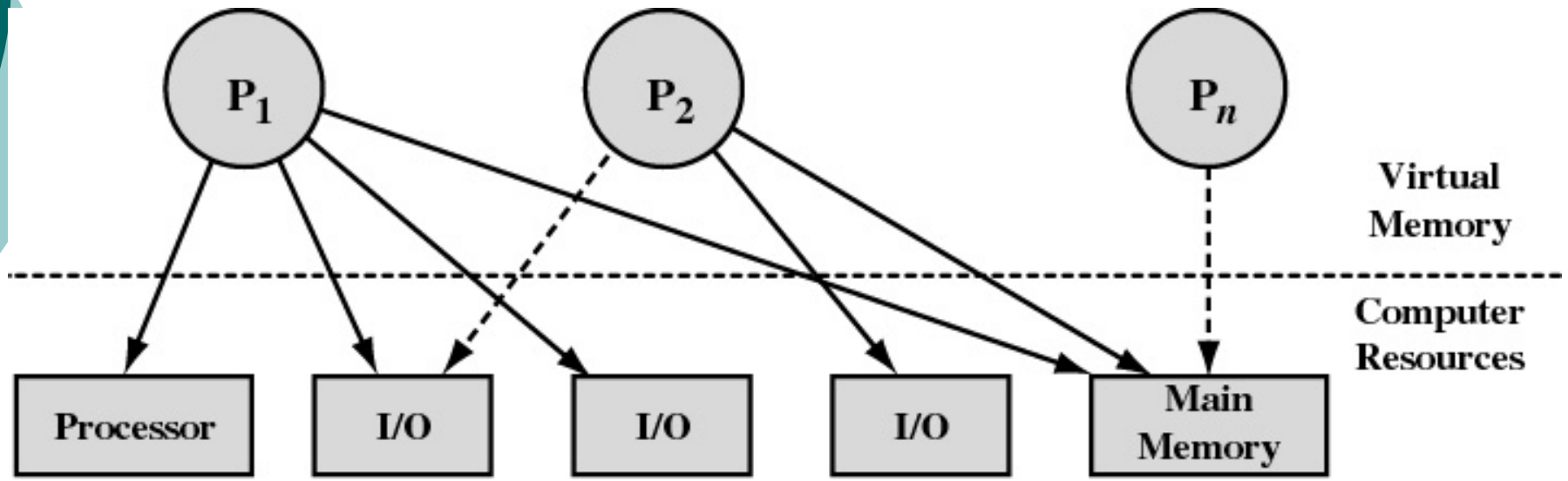


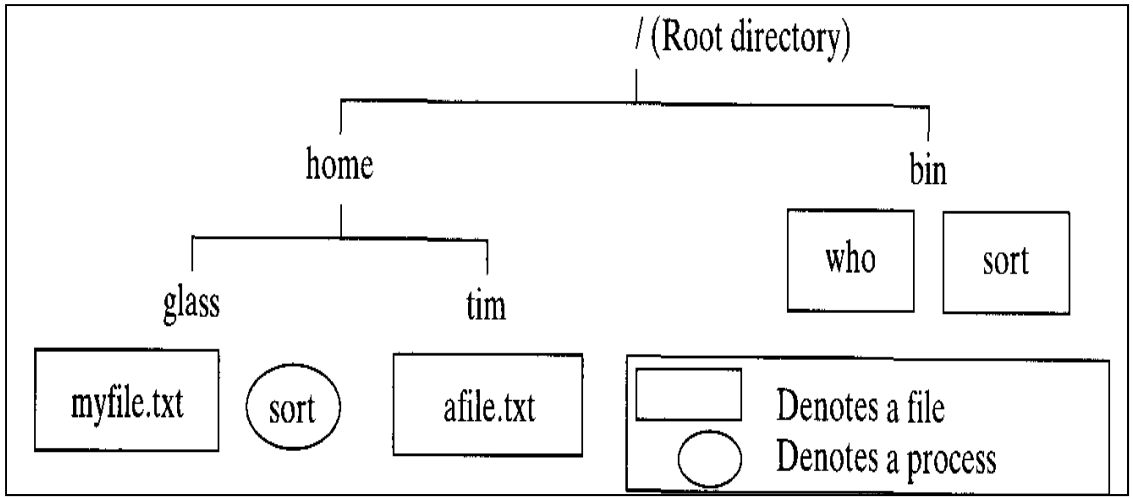
Figure 3.9 Processes and Resources (resource allocation at one snapshot in time)

OS Functions

- OS controls events within the computer system.
- Main functions of OS:
  - schedules and dispatches processes for execution by the processor
  - allocates resources to processes
  - responds to requests by user programs for basic services.
- OS manages the use of system resources by processes

Owner of Processes and Files

- Processes and files have an *owner* and may be protected against unauthorized access.
- UNIX supports a hierarchical directory structure.
- Files and processes have a “location” within the directory hierarchy. A process may change its own location or the location of a file.
- The figure below is an illustration of a tiny UNIX directory hierarchy that contains four files and a process running the “sort” utility.



Sharing Resources

- OS shares *memory* among processes
  - By dividing the main memory up into thousands of equal-sized "chunks" of memory and then allocating them to processes
  - The chunks of a process are called *pages* and chunks of memory are called *frames*

Sharing Memory - Paging

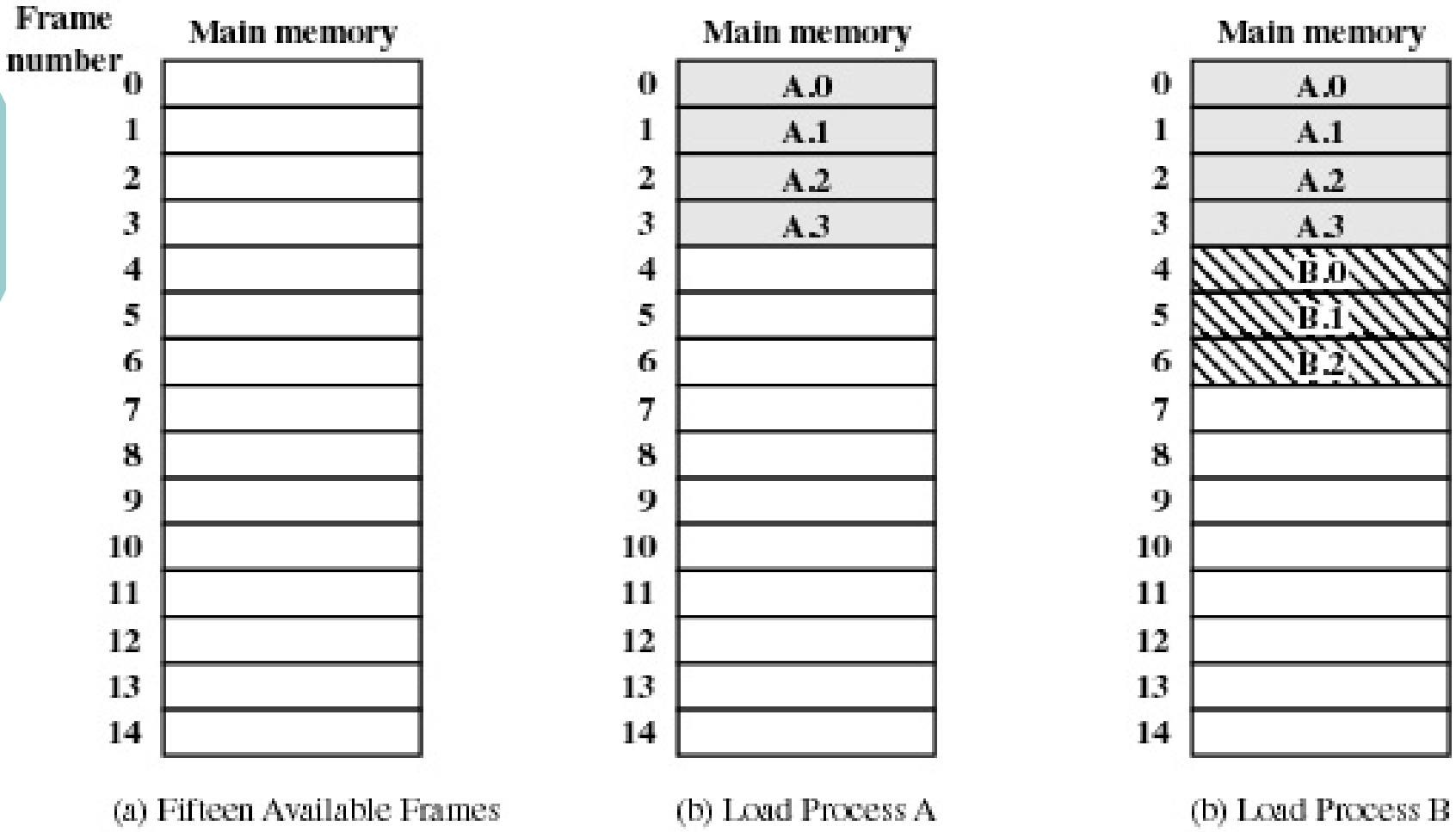


Figure 7.9 Assignment of Process Pages to Free Frames

Sharing Memory - Paging

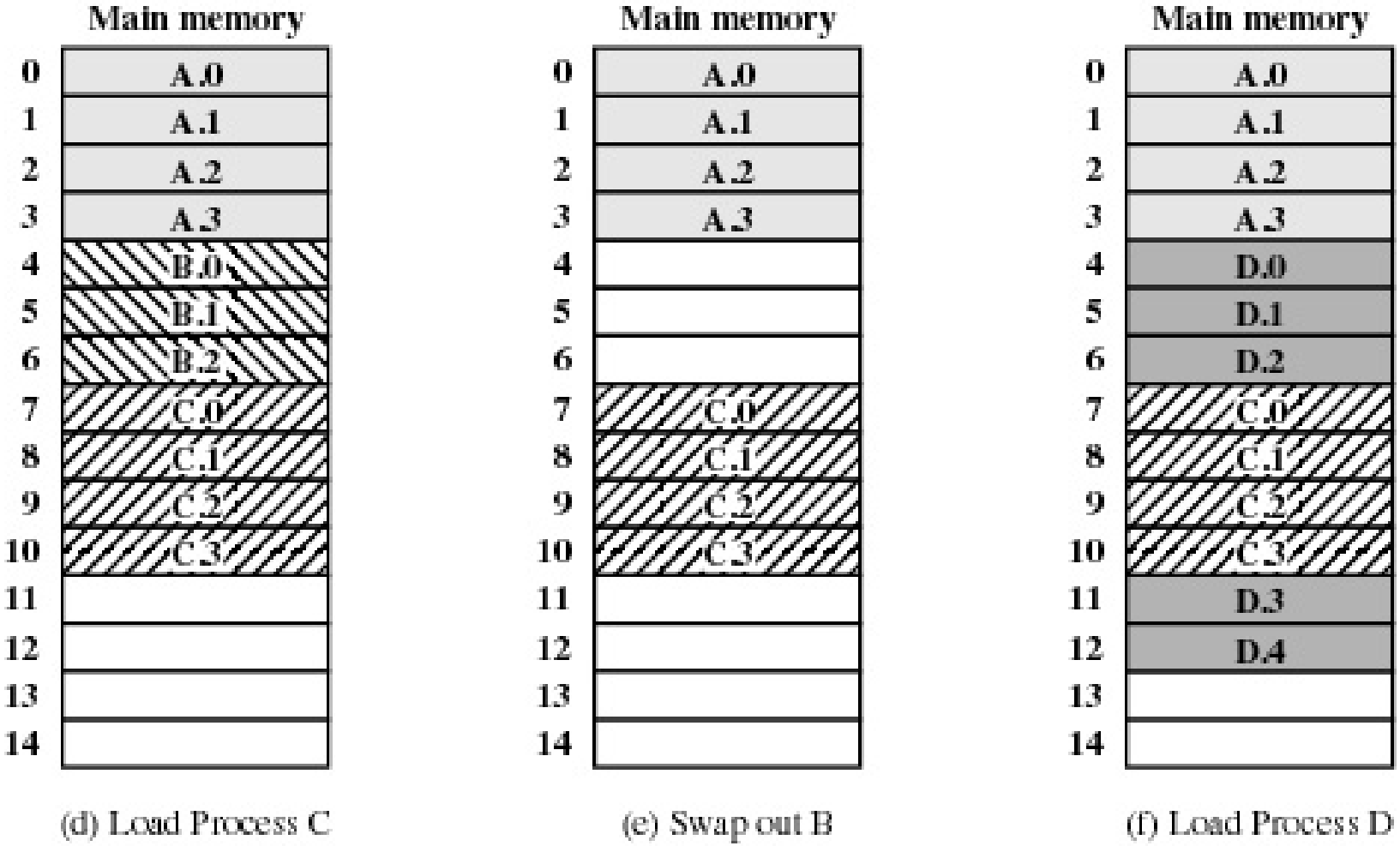


Figure 7.9 Assignment of Process Pages to Free Frames

Sharing Memory – Page Tables for Paging

- Operating system maintains a page table for each process
  - contains the frame location for each page in the process
  - memory address consist of a page number and offset within the page

Sharing Memory – Page Tables for Paging

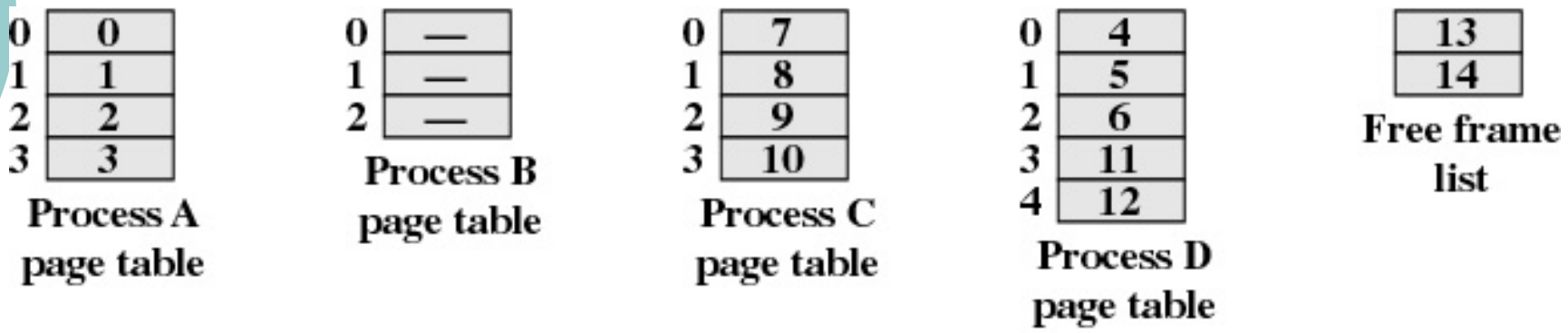


Figure 7.10 Data Structures for the Example of Figure 7.9 at Time Epoch (f)

Sharing Resources

- OS shares *disk space* among users
  - by dividing the disks into thousands of equal-sized “blocks” and then allocating them to users according to a quota system.
  - A single file is built out of one or more blocks.