

بسم الله الرحمن الرحيم

«سیستم عامل»

۱

جلسه ۴: پردازش

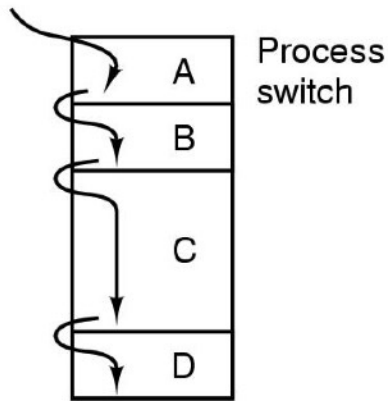
The Process Concept

The Process Concept

- **Process - a program in execution**
 - ❖ **Program**
 - description of how to perform an activity
 - instructions and static data values
 - ❖ **Process**
 - a snapshot of a program in execution
 - memory (program instructions, static and dynamic data values)
 - CPU state (registers, PC, SP, etc)
 - operating system state (open files, accounting statistics etc)

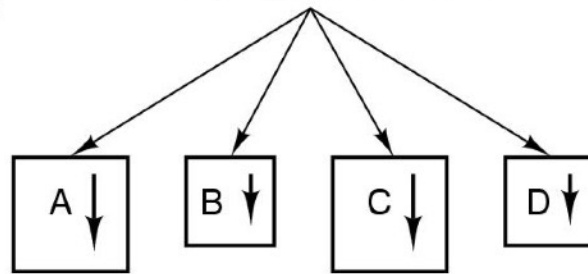
Why use the process abstraction?

One program counter

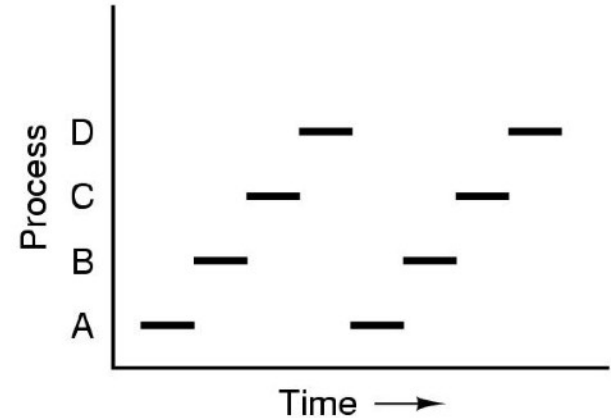


(a)

Four program counters



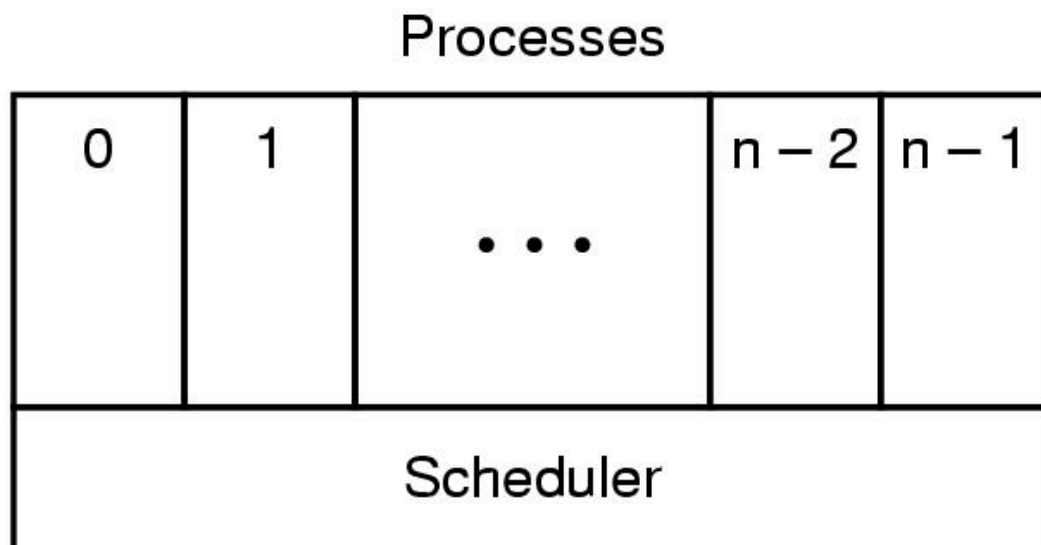
(b)



(c)

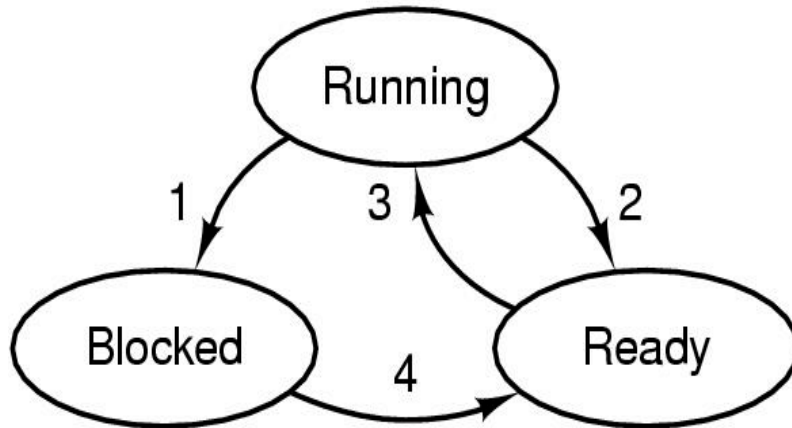
- ❑ **Multiprogramming of four programs in the same address space**
- ❑ **Conceptual model of 4 independent, sequential processes**
- ❑ **Only one program active at any instant**

The role of the scheduler



- ❑ **Lowest layer of process-structured OS**
 - ❖ handles interrupts & scheduling of processes
- ❑ **Sequential processes only exist above that layer**

Process states



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

- **Possible process states**
 - ❖ running
 - ❖ blocked
 - ❖ ready

How do processes get created?

Principal events that cause process creation

- ❑ System initialization
- ❑ Initiation of a batch job
- ❑ User request to create a new process
- ❑ Execution of a process creation system call from another process

Process hierarchies

- ❑ **Parent creates a child process,**
 - ❖ special system calls for communicating with and waiting for child processes
 - ❖ each process is assigned a unique identifying number or process ID (PID)
- ❑ **Child processes can create their own child processes**
 - ❖ Forms a hierarchy
 - ❖ UNIX calls this a "process group"

Process creation in UNIX

- ❑ **All processes have a unique process id**
 - ❖ `getpid()`, `getppid()` system calls allow processes to get their information
- ❑ **Process creation**
 - ❖ `fork()` system call creates a copy of a process and returns in both processes (parent and child), but with a different return value
 - ❖ `exec()` replaces an address space with a new program
- ❑ **Process termination, signaling**
 - ❖ `signal()`, `kill()` system calls allow a process to be terminated or have specific signals sent to it

Example: process creation in UNIX

csch (pid = 22)

```
...  
  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
  
...
```

Process creation in UNIX example

csh (pid = 22)

```
...  
pid = fork()  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
...
```

Process creation in UNIX example

csch (pid = 22)

```
...  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
...
```

csch (pid = 24)

```
...  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
...
```

Process creation in UNIX example

csch (pid = 22)

```
...  
  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
  
...
```

csch (pid = 24)

```
...  
  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
  
...
```

Process creation in UNIX example

csh (pid = 22)

```
...  
  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
  
...
```

csh (pid = 24)

```
...  
  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
else {  
    // parent  
    wait();  
}  
  
...
```

Process creation in UNIX example

csh (pid = 22)

```
...  
  
pid = fork()  
  
if (pid == 0) {  
    // child...  
    ...  
    exec();  
}  
  
else {  
    // parent  
    wait();  
}  
  
...
```

ls (pid = 24)

```
//ls program  
main() {  
    //look up dir  
    ...  
}
```

Process creation (fork)

- ❑ Fork creates a new process by copying the calling process
- ❑ The new process has its own
 - ❖ memory address space (copied from parent)
 - Instructions
 - Data
 - Stack
 - ❖ Register set (copied from parent)
 - ❖ Process table entry in the OS

Killing a process

- ❑ Sending kill signal to kernel
- ❑ Killing a process does not kill its descendants
 - ❑