Operating Systems

Process Description and Control

Chapter 3
Outline

• Process States
• Process Description
• Process Control
Major Requirements of an Operating System

- Interleave the execution of several processes to maximize processor utilization while providing reasonable response time
- Allocate resources to processes
- Support interprocess communication and user creation of processes
Process

- Also called a task
- Execution of an individual program
- Can be traced
  - list the sequence of instructions that execute for that process (*process trace*)
Figure 3.1  Snapshot of Example Execution (Figure 3.3) at Instruction Cycle 13
<table>
<thead>
<tr>
<th>5000</th>
<th>8000</th>
<th>12000</th>
</tr>
</thead>
<tbody>
<tr>
<td>5001</td>
<td>8001</td>
<td>12001</td>
</tr>
<tr>
<td>5002</td>
<td>8002</td>
<td>12002</td>
</tr>
<tr>
<td>5003</td>
<td>8003</td>
<td>12003</td>
</tr>
<tr>
<td>5004</td>
<td></td>
<td>12004</td>
</tr>
<tr>
<td>5005</td>
<td></td>
<td>12005</td>
</tr>
<tr>
<td>5006</td>
<td></td>
<td>12006</td>
</tr>
<tr>
<td>5007</td>
<td></td>
<td>12007</td>
</tr>
<tr>
<td>5008</td>
<td></td>
<td>12008</td>
</tr>
<tr>
<td>5009</td>
<td></td>
<td>12009</td>
</tr>
<tr>
<td>5010</td>
<td></td>
<td>12010</td>
</tr>
<tr>
<td>5011</td>
<td></td>
<td>12011</td>
</tr>
</tbody>
</table>

**I/O operation**

(a) Trace of Process A       (b) Trace of Process B       (c) Trace of Process C

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
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5001</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5005</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>8000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>8002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8003</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>12000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>12001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>12002</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>12003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>12004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>12005</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time out

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>5006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>5007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>5008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>5009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>5010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>5011</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Time out

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>105</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>12006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>12007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>12008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>12009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>12010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>12011</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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
Two-State Process Model

• Process may be in one of two states
  – Running
  – Not-running
Not-Running Process in a Queue

(b) Queuing diagram
Process Creation

- Submission of a batch job
- User logs on
- Created to provide a service such as printing
- Process creates another process
Process Termination

- Batch job issues *Halt* instruction
- User logs off
- Quit an application
- Error and fault conditions
Reasons for Process Termination

- Normal completion
- Time limit exceeded
- Memory unavailable
- Bounds violation
- Protection error
  - example write to read-only file
- Arithmetic error
- Time overrun
  - process waited longer than a specified maximum for an event
Reasons for Process Termination

- I/O failure
- Invalid instruction
  - happens when try to execute data
- Privileged instruction
- Data misuse
- Operating system intervention
  - such as when deadlock occurs
- Parent terminates so child processes terminate
- Parent request
Processes

- **Not-running**
  - ready to execute

- **Blocked**
  - waiting for I/O

- Dispatcher cannot just select the process that has been in the queue the longest because it may be blocked
A Five-State Model

- Running
- Ready
- Blocked
- New
- Exit
Figure 3.5 Five-State Process Model
Figure 3.6  Process States for Trace of Figure 3.3
Using Two Queues

- When an event occurs, all processes in the blocked queue that are waiting on that event are moved to the ready queue.

- If dispatching of processes dictated by a priority scheme, can have a number of ready queues, one for each priority level.
Why multiple event queues?
Suspended Processes

- Processor is faster than I/O so all processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- *Blocked state becomes* suspend state when swapped to disk
- Two new states
  - Blocked, suspend (awaiting an event)
  - Ready, suspend (available for execution when loaded into main memory)
One Suspend State

(a) With One Suspend State
Two Suspend States
## Reasons for Process Suspension

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swapping</td>
<td>The operating system needs to release sufficient main memory to bring in a process that is ready to execute.</td>
</tr>
<tr>
<td>Other OS reason</td>
<td>The operating system may suspend a background or utility process or a process that is suspected of causing a problem.</td>
</tr>
<tr>
<td>Interactive user request</td>
<td>A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.</td>
</tr>
<tr>
<td>Timing</td>
<td>A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.</td>
</tr>
<tr>
<td>Parent process request</td>
<td>A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendents.</td>
</tr>
</tbody>
</table>
Process Description

- Operating System control structures
- Process control structures
OS manages the use of system resources by processes

$P_1$: running, part in main memory, has control of 2 I/O devices

$P_2$: in main memory, blocked waiting for an I/O device (allocated to $P_1$)

$P_n$: swapped out (suspended)

Figure 3.9 Processes and Resources (resource allocation at one snapshot in time)
Operating System Control Structures

- Information about the current status of each process and resource
- Tables are constructed for each entity the operating system manages
Figure 3.10 General Structure of Operating System Control Tables
Memory Tables

- Allocation of main memory to processes
- Allocation of secondary memory to processes
- Protection attributes for access to shared memory regions
- Information needed to manage virtual memory

© Dr. Ayman Abdel-Hamid, OS
I/O Tables

• I/O device is available or assigned
• Status of I/O operation
• Location in main memory being used as the source or destination of the I/O transfer
File Tables

• Existence of files
• Location on secondary memory
• Current Status
• Attributes
• Sometimes this information is maintained by a file-management system
Process Table

• Where process is located
• Attributes necessary for its management
  – Process ID
  – Process state
  – Location in memory
Process Control Structures

- Process Location
- Process Control Block
Process Location

- Process includes set of programs to be executed
  - Data locations for local and global variables
  - Any defined constants
  - Stack

- Process control block (process descriptor)
  - Collection of attributes used for process control

- Process image
  - Collection of program, data, stack, and attributes
Process Control Block

- See Table 3.5 on page 129
- Process identification
  - Identifiers
    - Numeric identifiers that may be stored with the process control block include
      - Identifier of this process (might be index into primary process table)
      - Identifier of the process that created this process (parent process)
      - User identifier responsible for the job
- Processor state information
- Process control information
• Processor State Information
  – Contents of processor registers while a process is running
    • User-visible registers
    • Control and status registers
    • Stack pointers
  – User-Visible Registers
    • A user-visible register is one that may be referenced by means of the machine language that the processor executes. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.
• Processor State Information
  – Control and Status Registers

These are a variety of processor registers that are employed to control the operation of the processor. These include

• *Program counter:* Contains the address of the next instruction to be fetched

• *Condition codes:* Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)

• *Status information:* Includes interrupt enabled/disabled flags, execution mode
Process Control Block 4/10

• Processor State Information
  – Stack Pointers
    • Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls. The stack pointer points to the top of the stack.
• Processor State Information
  – Program status word (PSW)
    • A register or a set of registers
    • Typically contains condition codes and other status information
    • Example: the EFLAGS register on Pentium machines
Process Control Block 6/10
Pentium II EFLAGS Register

<table>
<thead>
<tr>
<th>31</th>
<th>21</th>
<th>16</th>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
<td>V</td>
<td>V</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>I</td>
<td>P</td>
<td></td>
</tr>
</tbody>
</table>

ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

**Figure 3.11** Pentium II EFLAGS Register

© Dr. Ayman Abdel-Hamid, OS 39
• Process Control Information
  – Scheduling and State Information
    This is information that is needed by the operating system to perform its scheduling function. Typical items of information:
    • **Process state**: defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).
    • **Priority**: One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable)
    • **Scheduling-related information**: This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
    • **Event**: Identity of event the process is awaiting before it can be resumed
Process Control Block 8/10

• Process Control Information
  – Data Structuring
    • A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent-child (creator-created) relationship with another process. The process control block may contain pointers to other processes to support these structures.
• Process Control Information
  – Interprocess Communication
    • Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.
  – Process Privileges
    • Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.
Process Control Block

- Process Control Information
  - Memory Management
    - This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.
  - Resource Ownership and Utilization
    - Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.
Figure 3.12  User Processes in Virtual Memory

© Dr. Ayman Abdel-Hamid, OS
Process Control

• Modes of Execution
• Process Creation
• Process Switching
• Execution of the Operating System

© Dr. Ayman Abdel-Hamid, OS
Modes of Execution

• User mode
  – Less-privileged mode
  – User programs typically execute in this mode
• System mode, control mode, or kernel mode
  – More-privileged mode
  – Kernel of the operating system
• A bit in PSW indicates the execution mode
• A user makes a call to an OS service → the mode is set to kernel mode (typically, by executing an instruction that changes the mode)
Process Creation

- Assign a unique process identifier
- Allocate space for the process
- Initialize process control block
- Set up appropriate linkages
  - Ex: add new process to linked list used for scheduling queue
- Create or expand other data structures
  - Ex: maintain an accounting file
When to Switch a Process 1/2

- A process switch may occur any time the OS gains control from the currently running process.

- Interrupts
  - Clock interrupt
    - process has executed for the maximum allowable time slice
  - I/O interrupt
  - Memory fault
    - memory address is in virtual memory so it must be brought into main memory
When to Switch a Process 2/2

- **Trap**
  - error occurred
  - may cause process to be moved to Exit state
- **Supervisor call**
  - such as file open
  - Generally, the use of a system call results in placing the user process in the Blocked state
Mode Switching 1/2

• If an interrupt is pending, the process does the following
  – Saves the context of the current program being executed
  – Sets the program counter to the starting address of an interrupt-handler program
  – *Switches from user mode to kernel mode* since the interrupt processing code may include privileged instructions
Mode Switching 2/2

• In most OS, the occurrence of an interrupt does not necessarily mean a process switch

• After interrupt handler has executed, the currently running process might resume execution
  – Save processor state information when interrupt occurs
  – Restore information when control is returned to the program that was in progress
Change of Process State 1/2

Steps in a full process switch

• Save context of processor including program counter and other registers
• Update the process control block of the process that is currently running
• Move process control block to appropriate queue - ready, blocked
• Select another process for execution
Steps in a full process switch (cont.)

- Update the process control block of the process selected
- Update memory-management data structures
- Restore context of the selected process
Execution of the Operating System

Is the OS a process?

(a) Separate kernel

(b) OS functions execute within user processes

(c) OS functions execute as separate processes

© Dr. Ayman Abdel-Hamid, OS
Execution of the Operating System

- **Non-process Kernel (older OS)**
  - execute kernel outside of any process
  - operating system code is executed as a separate entity that operates in privileged mode

- **Execution Within User Processes (smaller machines: PCs and workstations)**
  - operating system software within context of a user process
  - process executes in privileged mode when executing operating system code
Contains OS code and data shared by all user processes

Figure 3.15  Process Image: Operating System Executes Within User Space
Execution of the Operating System

- Process-Based Operating System
  - major kernel functions are separate processes
  - Useful in multi-processor or multi-computer environment (some of the OS services can be shipped out to dedicated processors, improving performance)