Process Description and Control

Chapter 3
Requirements of an Operating System

- Interleave the execution of multiple processes to maximize processor utilization while providing reasonable response time
- Allocate resources to processes
- Support interprocess communication and user creation of processes
Manage Execution of Applications

- Resources made available to multiple applications
- Processor is switched among multiple applications
- The processor and I/O devices can be used efficiently
Process

• A program in execution
• An instance of a program running on a computer
• The entity that can be assigned to and executed on a processor
• A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system instructions
Process Elements

- Identifier
- State
- Priority
- Program counter
- Memory pointers
- Context data
- I/O status information
- Accounting information
Process Control Block

- Contains the process elements
- Created and manage by the operating system
- Allows support for multiple processes
Process Control Block

Figure 3.1 Simplified Process Control Block
Trace of Process

• Sequence of instruction that execute for a process
• Dispatcher switches the processor from one process to another
Example Execution

Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13
Trace of Processes

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>8000</td>
<td>12000</td>
</tr>
<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>

(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

Process B is waiting for I/O
Figure 3.3 Traces of Processes of Figure 3.2

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 3.4 Combined Trace of Processes of Figure 3.2

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
Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running

(a) State transition diagram
Not-Running Process in a Queue

(b) Queuing diagram
## Process Creation

### Table 3.1 Reasons for Process Creation

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New batch job</td>
<td>The operating system is provided with a batch job control stream, usually on tape or disk. When the operating system is prepared to take on new work, it will read the next sequence of job control commands.</td>
</tr>
<tr>
<td>Interactive logon</td>
<td>A user at a terminal logs on to the system.</td>
</tr>
<tr>
<td>Created by OS to provide a service</td>
<td>The operating system can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).</td>
</tr>
<tr>
<td>Spawned by existing process</td>
<td>For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.</td>
</tr>
</tbody>
</table>
# Process Termination

**Table 3.2 Reasons for Process Termination**

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal completion</td>
<td>The process executes an OS service call to indicate that it has completed running.</td>
</tr>
<tr>
<td>Time limit exceeded</td>
<td>The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time (&quot;wall clock time&quot;), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input.</td>
</tr>
<tr>
<td>Memory unavailable</td>
<td>The process requires more memory than the system can provide.</td>
</tr>
<tr>
<td>Bounds violation</td>
<td>The process tries to access a memory location that it is not allowed to access.</td>
</tr>
<tr>
<td>Protection error</td>
<td>The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file.</td>
</tr>
<tr>
<td>Arithmetic error</td>
<td>The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate.</td>
</tr>
</tbody>
</table>
## Process Termination

Table 3.2  Reasons for Process Termination

<table>
<thead>
<tr>
<th>Reason</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time overrun</td>
<td>The process has waited longer than a specified maximum for a certain event to occur.</td>
</tr>
<tr>
<td>I/O failure</td>
<td>An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).</td>
</tr>
<tr>
<td>Invalid instruction</td>
<td>The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).</td>
</tr>
<tr>
<td>Privileged instruction</td>
<td>The process attempts to use an instruction reserved for the operating system.</td>
</tr>
<tr>
<td>Data misuse</td>
<td>A piece of data is of the wrong type or is not initialized.</td>
</tr>
<tr>
<td>Operator or OS intervention</td>
<td>For some reason, the operator or the operating system has terminated the process (for example, if a deadlock exists).</td>
</tr>
<tr>
<td>Parent termination</td>
<td>When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.</td>
</tr>
<tr>
<td>Parent request</td>
<td>A parent process typically has the authority to terminate any of its offspring.</td>
</tr>
</tbody>
</table>
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
Five-State Process Model

![Diagram of the Five-State Process Model]

*Figure 3.6  Five-State Process Model*
Let’s look at the execution again.

What are the state transitions of the processes?

![Diagram of process traces]

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

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 Traces of Processes of Figure 3.2

Figure 3.4 Combined Trace of Processes of Figure 3.2
Process States

![Diagram showing process states]

**Figure 3.7 Process States for Trace of Figure 3.4**
Using Two Queues

Is the Blocked Queue a FIFO queue?
What is the problem with the Blocked Queue?
Multiple Blocked Queues

Event 1 Occurs
Event 2 Occurs
Event $n$ Occurs

Event 1 Queue
Event 2 Queue
Event $n$ Queue

Admit
Ready Queue
Dispatch
Timeout
Release

Event 1 Wait
Event 2 Wait
Event $n$ Wait

(b) Multiple blocked 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
  – Ready/Suspend
One Suspend State

(a) With One Suspend State
Two Suspend States

(b) With Two Suspend States

Figure 3.9 Process State Transition Diagram with 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>