Chapter 8
Operating System Support
Objectives and Functions

• Convenience
  ▪ Making the computer easier to use

• Efficiency
  ▪ Allowing better use of computer resources
Layers and Views of a Computer System

- End User
- Programmer
- Operating System Designer

Application Programs
Utilities
Operating System
Computer Hardware
Operating System Services

- Program creation
- Program execution
- Access to I/O devices
- Controlled access to files
- System access
- Error detection and response
- Accounting
OS as Resource Manager

- OS is responsible for managing resources of the computer
- OS is an unusual control mechanism in two respects:
  - it functions in the same way as ordinary computer software
  - it frequently relinquishes control and must depend on the processor to allow it to regain control
O/S as a Resource Manager

Computer System

Memory

Operating System Software

Programs and Data

I/O Controller

I/O Devices

Storage

OS

Programs

Data

Processor

Processor
Types of Operating System

- Batch
- Interactive
- Single program (Uni-programming)
- Multiple programs (Multi-tasking)
Early Systems

- Late 1940s to mid 1950s
- No Operating System
- Programs interact directly with hardware
- Two main problems:
  - Scheduling
  - Set-up time
Simple Batch Systems

- Resident Monitor program
- Users submit jobs to operator
- Operator batches jobs
- Monitor controls sequence of events to process batch
- When one job is finished, control returns to Monitor which reads next job
- Monitor handles scheduling
Job Control Language

• Instructions to Monitor
• Usually denoted by $ 
• e.g.
  ▪ $JOB
  ▪ $FTN
  ▪ ... Some Fortran instructions
  ▪ $LOAD
  ▪ $RUN
  ▪ ... Some data
  ▪ $END
Desirable Hardware Features

- Memory protection
  - To protect the Monitor
- Timer
  - To prevent a job monopolizing the system
- Privileged instructions
  - Only executed by Monitor
  - e.g. I/O
- Interrupts
  - Allows for relinquishing and regaining control
Overhead

• Two sacrifices:
  ▪ some main memory is used by the monitor
  ▪ some CPU time is consumed by the monitor
Multi-programmed Batch Systems

- I/O devices very slow
- When one program is waiting for I/O, another can use the CPU
Single Program
Multi-Programming with Two Programs

Program A
- Run
- Wait
- Run
- Wait

Program B
- Wait
- Run
- Wait
- Run
- Wait

Combined
- Run A
- Run B
- Wait
- Run A
- Run B
- Wait

Time
Multi-Programming with Three Programs
Time Sharing Systems

- Multi-programming allows to have many programs running at the same time
  - ... but multi-user interactivity requires more
- In each given amount of time (small!) pay attention to every user
  - Users share the time of the computer, each actually consuming only a little slice
Concept of process

• Several definitions including:
  ▪ a program in execution
  ▪ the “animated spirit” of a program
  ▪ the entity to which a processor is assigned
Scheduling

- Key to multi-programming
- Long term scheduling
- Medium term scheduling
- Short term scheduling
- I/O management
Long Term Scheduling

- Determines which programs are accepted for processing
  - i.e. controls the degree of multi-programming
- Currently, simply accept programs for processing
  - i.e. transform a program into a process
- Once accepted, a program becomes a process, to be managed by the short term scheduler
- ... or it becomes a swapped out job for the medium term scheduler
Medium Term Scheduling

- Determines which process can be entered in the central memory (i.e., swapped in)
- Part of the swapping function (later...)
- Usually based on the need to manage multi-programming
- If no virtual memory, memory management is also an issue
Short Term Scheduler

- Dispatcher
- Fine grained decisions of which process to execute next
- i.e. which process actually gets to use the processor in the next time slot
Process States
Process Control Block

- Identifier
- State
- Priority
- Program counter
- Process Memory pointers
- Context data
- I/O status
- Accounting information
### A simple example

<table>
<thead>
<tr>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service handler</td>
</tr>
<tr>
<td>scheduler</td>
</tr>
<tr>
<td>Interrupt handler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>“running”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ready”</td>
</tr>
</tbody>
</table>

| Other partitions |

---

<table>
<thead>
<tr>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service handler</td>
</tr>
<tr>
<td>scheduler</td>
</tr>
<tr>
<td>Interrupt handler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>“waiting”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>“ready”</td>
</tr>
</tbody>
</table>

| Other partitions |

---

<table>
<thead>
<tr>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service handler</td>
</tr>
<tr>
<td>scheduler</td>
</tr>
<tr>
<td>Interrupt handler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>“waiting”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>“running”</td>
</tr>
</tbody>
</table>

| Other partitions |

---

Rev. 3.2 (2009-10) by Enrico Nardelli
Key Elements of O/S

Operating System

- Service Call from Process
  - Service Call Handler
- Interrupt from Process
  - Interrupt Handler
- Interrupt from I/O

Queues:
- Long-Term Queue
- Short-Term Queue
- I/O Queues

Short-Term Scheduler

Pass Control to Process
Process Scheduling

Medium-Term Queue

Long-Term Queue

Short-Term Queue

CPU

I/O

I/O Queue

I/O

I/O Queue

End
Memory Management

• Uni-program
  ▪ Memory split into two
  ▪ One for Operating System (monitor)
  ▪ One for currently executing program

• Multi-program
  ▪ “User” part is sub-divided and shared among active processes
  ▪ Requires memory management capabilities
Swapping

- Problem: I/O is so slow compared with CPU that even in multi-programming system, CPU can be idle most of the time
- Solutions:
  - Increase main memory
    - Expensive
    - Leads to larger programs
  - Swapping
What is Swapping?

- Long term queue of processes stored on disk
- Processes “swapped” in as space becomes available
- As a process completes it is moved out of main memory
- If none of the processes in memory are ready (i.e. all I/O blocked)
  - Swap out a blocked process to medium-term queue
  - Swap in a ready process or a new process
  - But swapping is an I/O process...
Swapping

- Disk storage
- Main memory
- Operating System

Medium-term queue
Long-term queue
Completed jobs
Partitioning

• Splitting memory into sections to allocate to processes (including Operating System)

• Fixed-size partitions
  ▪ First solution: all partitions of equal size, but it wastes a lot of memory
  ▪ Better solution: partitions of unequal size.
    • Process is fitted into smallest hole that will take it (best fit)
    ▪ Still some wasted memory

• Leads to variable sized partitions
Fixed-size Partitioning

(a) Equal-size partitions

(b) Unequal-size partitions

Rev. 3.2 (2009-10) by Enrico Nardelli
Variable Sized Partitions (1)

- Allocate exactly the required memory to a process
- This leads to a hole at the end of memory, too small to use
  - Only one small hole - less waste
- When all processes are blocked, swap out a process and bring in another
- New process may be smaller than swapped out process
- Another hole
Variable Sized Partitions (2)

- Eventually have lots of holes (fragmentation)
- Solutions:
  - Coalesce - Join adjacent holes into one large hole
  - Compact - From time to time go through memory and move all hole into one free block (c.f. disk de-fragmentation)
Effect of Dynamic Partitioning

(a) Operating System 128K
896K

(b) Operating System
Process 1 320K
576K

(c) Operating System
Process 1 320K
Process 2 224K
352K
Process 3 288K
64K

(d) Operating System
Process 1 320K
Process 2 224K
Process 3 288K
64K

(e) Operating System
Process 1 320K
Process 4 224K
128K
96K
Process 3 288K
64K

(f) Operating System
Process 1 320K
Process 4 128K
96K
96K
Process 3 288K
64K

(g) Operating System
Process 2 224K
96K
96K
Process 4 128K
96K
96K
Process 3 288K
64K

(h) Operating System
Process 3 288K
64K
64K
Relocation

- No guarantee that process will load into the same place in memory
- Instructions contain addresses
  - Locations of data
  - Addresses for instructions (branching)
- Logical address - relative to beginning of program
- Physical address - actual location in memory (this time)
- Automatic conversion using base address
- Hardware feature supporting an OS requirement
Paging

- Split memory into equal sized, small chunks - page frames
- Split programs (processes) into equal sized small chunks - pages
- Allocate the required number page frames to a process
- Operating System maintains list of free frames
- A process does not require contiguous page frames
- Use page table to keep track
Logical and Physical Addresses - Paging
Virtual Memory

- Demand paging
  - Do not require all pages of a process in memory
  - Bring in pages as required

- Page fault
  - Required page is not in memory
  - Operating System must swap in required page
  - May need to swap out a page to make space
  - Select page to throw out based on recent history
Bonus

- We do not need all of a process in memory for it to run
- We can swap in pages as required
- So - we can now run processes that are bigger than total memory available!

- Main memory is called real memory
- User/programmer sees much bigger memory - virtual memory
Thrashing

- Too many processes in too little memory
- Operating System spends all its time swapping
- Little or no real work is done
- Disk light is on all the time

Solutions
- Good page replacement algorithms
- Fit more memory
- Reduce number of processes running
Some details about paging

- Where is the page table (PT) stored?
  - ...in the main memory:
    - two registers:
      - Page-table base register (PTBR)
      - Page-table length register (PTLR)
    - for each address we have 2 memory accesses
      - Usually a cache is used: translation lookaside buffer (TLB)
- What can we do if the PT is too big?
  - two-level paging: paging of the PT
- And if there are too many processes?
  - inverted page table
Idea:
an entry for each memory frame

search is expensive!!!
Page Table Structure (hash!)
Segmentation

- Paging is not (usually) visible to the programmer
- Segmentation is visible to the programmer
- Usually different segments allocated to program and data
- May be a number of program and data segments
Advantages of Segmentation

• Simplifies handling of growing data structures
• Allows programs to be altered and recompiled independently, without re-linking and re-loading
• Lends itself to sharing among processes
• Lends itself to protection
• Some systems combine segmentation with paging