

## CS 5523 Lecture 12: The Operating System Layer

---

- *Basic OS terminology*
- *Operating System vs Network Operating System vs Distributed System vs Middleware*
- *Operating Systems Requirements*
- *How an OS Works to Execute an Application*
- *Introduction to Processes and Threads*

### Basic terminology

---

#### *Operating system:*

- *provides an abstraction of underlying resources*
- *manages and protects these resources*

#### *Network operating systems (e.g. Unix or NT):*

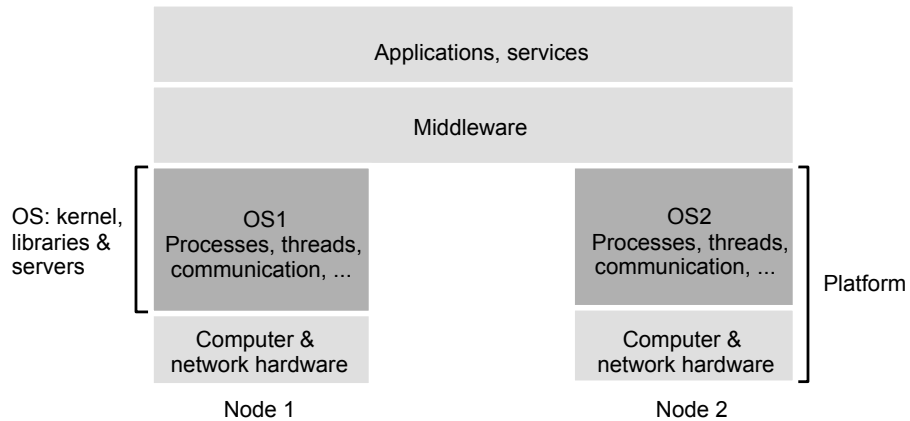
- *have networking capability to access remote resources*
- *retain autonomy in managing own resources*
- *remote resource access not always transparent*
- *separate system image on each node*

#### *Distributed operating system:*

- *single system image across multiple nodes*
- *resource access completely transparent*

*Why are there no distributed operating systems in general use?*

Figure 6.1  
System layers



Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
© Addison-Wesley Publishers 2000

## Operating systems versus middleware

### *Operating system:*

- ▮ *provides an abstraction of underlying resources*
- ▮ *manages and protects these resources*

### *Middleware:*

- ▮ *runs over the operating system a user process*
- ▮ *provides the glue for resources accessed across the network*
- ▮ *provides remote invocation, name management, etc.*

*We're going to do Unix with CORBA. An alternative would be NT or Windows 2000 with COM*

## Operating systems requirements

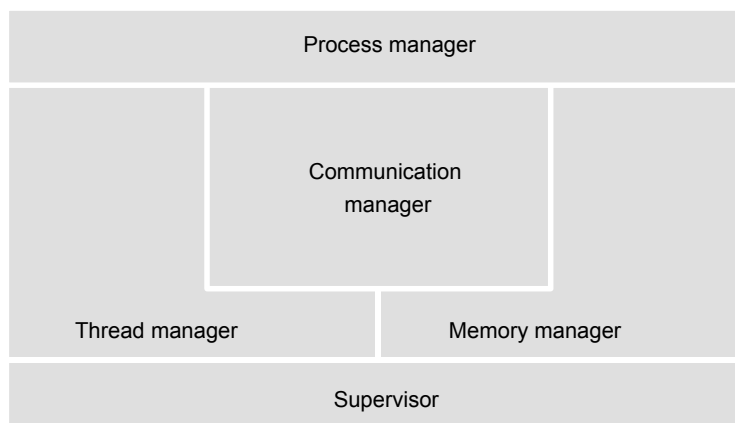
---

- *Encapsulation*
- *Protection*
- *Concurrent processing*
- *Communication*
- *Scheduling*

*Explain what each of these requirements means and why it is necessary. Give some examples of each idea in Unix*

Figure 6.2  
Core OS functionality

---



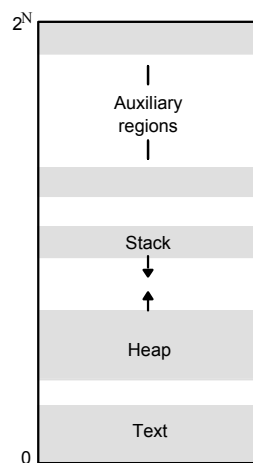
## Review the basic operation of an operating system:

---

- *What is a system call?*
- *What is the difference between a trap and an interrupt?*
- *How do interrupts and traps interact with the normal instruction cycle of a CPU?*
- *What is the difference between supervisor mode and user mode?*
- *How does address space and virtual memory support protection?*

Figure 6.3  
Address space

---



## What happens when an application is executed?

---

- *How is the running program created?*
- *How does it run?*
- *How are its resources deallocated when it is finished?*
- *What information is in the state of a process?*
- *What is a context switch?*
- *When do context switches happen?*

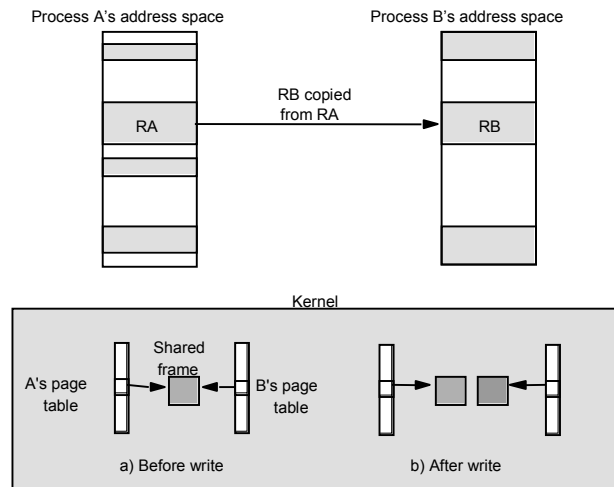
## Process creation:

---

- *What is the target host?*  
*Should process be created locally or according to some other policy such as load balancing on a cluster?*
- *How is the execution environment created?*  
*Address space can be created according to a statically specified format.*  
*Address space can be created relative to a pre-existing execution environment (Unix fork).*

*What does the Unix exec call do?*

Figure 6.4  
Copy-on-write

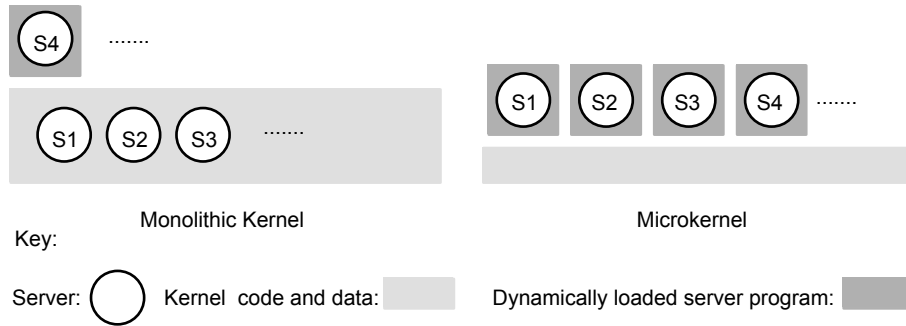


Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
© Addison-Wesley Publishers 2000

## Using multiple processes for Laboratory 1:

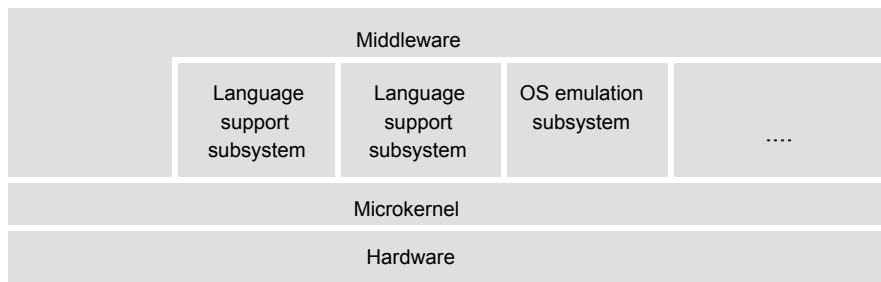
- *What is the role of the parent?*
- *How will end-of-session be detected?*
- *How should the log file be handled?*

Figure 6.15  
Monolithic kernel and microkernel



Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
© Addison-Wesley Publishers 2000

Figure 6.16  
The role of the microkernel



The microkernel supports middleware via subsystems

Instructor's Guide for Coulouris, Dollimore and Kindberg Distributed Systems: Concepts and Design Edn. 3  
© Addison-Wesley Publishers 2000

For next time:

---

- *Read CDK Chapter 6.4*
- *Read Stevens I Chapter 23.1-23.3*