

CS 5523 Lecture 11: The Operating System Layer

- Questions on laboratory 2
- Discuss questions from Chapter 4
- Basic OS terminology
- Layer functions: OS vs network vs middleware
- OS requirements
- How an OS executes an application
- Introduction to processes and threads

Discussion questions from CDK Chapter 4

CDK [4.3]

The programs in Figure 4.3 and Figure 4.4 are available on `cdk3.net/ipc`. Use them to make a test kit to determine the conditions in which datagrams are sometimes dropped (UDP).

Hint: the client program should be able to vary the number of messages sent and their size; the server should detect when a message from a particular client is missed.

Discussion Questions from CDK Chapter 4

[CDK 4.10]

Write an algorithm in pseudocode to describe the serialization procedure described in Section 4.3.2. The algorithm should show when handles are defined or substituted for classes and instances. Describe the serialized form that your algorithm would produce when serializing an instance of the following class `Couple`.

```
class Couple implements Serializable{
    private Person one;
    private Person two;
    public Couple(Person a, Person b) {
        one = a;
        two = b;
    }
}
```

Discussion Questions from CDK Chapter 4

[CDK 4.11]

Write an algorithm in pseudocode to describe deserialization of the serialized form produced by the algorithm defined in Exercise 4.10. Hint: use reflection to create a class from its name, to create a constructor from its parameter types and to create a new instance of an object from the constructor and the argument values.

Discussion Questions from CDK Chapter 4

[CDK 4.12]

Define a class whose instances represent remote object references. It should contain information similar to that shown in Figure 4.10 and should provide access methods needed by the request-reply protocol. Explain how each of the access methods will be used by that protocol. Give a justification for the type chosen for the instance variable containing information about the interface of the remote object.

Basic terminology

Operating system:

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

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

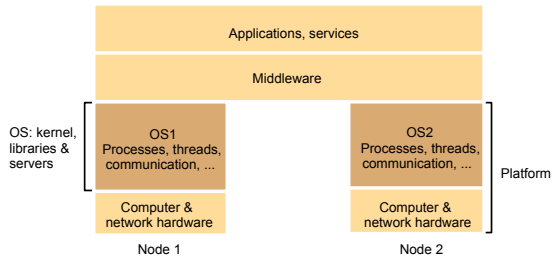
- have networking capability to access remote resources
- retain autonomy in managing own resources
- remote resource access not always transparent
- separate system image on each node

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 Condra, Dillman and Kriebel. 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:

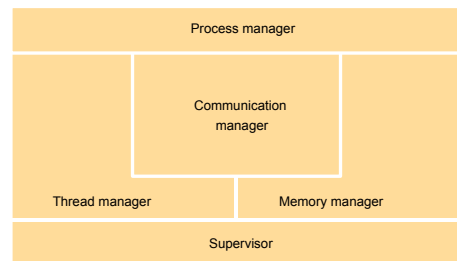
- runs over the operating system a user process
- provides the glue for resources accessed across the network
- provides remote invocation, name management, etc.

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

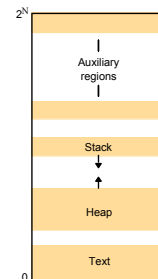


Instructor's Guide for Condra, Dillman and Kriebel. Distributed Systems: Concepts and Design. Edn. 3
© Addison-Wesley Publishers 2000

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



Instructor's Guide for Condra, Dillman and Kriebel. Distributed Systems: Concepts and Design. Edn. 3
© Addison-Wesley Publishers 2000

What happens when an application is executed?

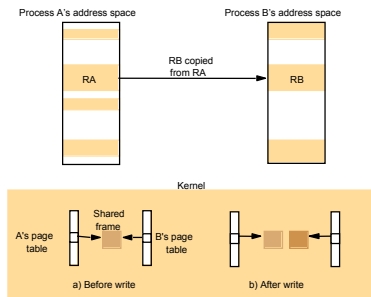
- How is the running program created?
- How does it run?
- How are its resources deallocated when it is finished?
- What information is in the state of a process?
- What is a context switch?
- When do context switches happen?

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).

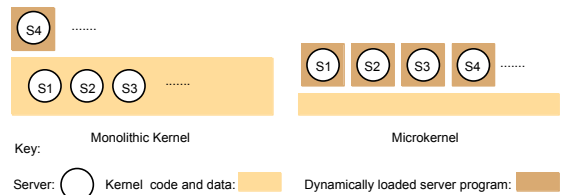
What does the Unix exec call do?

Figure 6.4
Copy-on-write



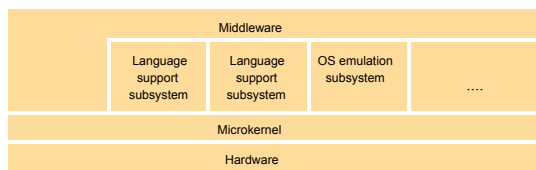
Instructor's Guide for Cookson, Deffensee and Knudsen. Distributed Systems: Concepts and Design. Edn. 3
© Addison-Wesley Publishers 2000

Figure 6.15
Monolithic kernel and microkernel



Instructor's Guide for Cookson, Deffensee and Knudsen. 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 Cookson, Deffensee and Knudsen. 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