Computer Systems
Lecture 6

Overview

- Operating systems: Examples
- Operating systems: Functions
- Onion ring model
- Interaction with operating systems
- OS services
- Computer networks
- Client-server computing

Operating systems: examples

- OS/360 for IBM System/360, 1960s.
- Unix, 1970s.
- MS-DOS for IBM PC and Mac OS for Apple Macintosh, 1980s.
- Windows 95, 98, NT, 1990s.
  - NT served as the basis for Microsoft's desktop operating system line starting in 2001.
- Apple rebuilt their operating systems on top of a Unix core as Mac OS X, released in 2001.
- Linux, BSD Unix…

Operating systems: examples (cont.)

- Operating systems have undergone more than 40 years of evolution,

"…but surprisingly little has changed at the technical core. Only the introduction of **windowing interfaces** really distinguishes."

[Wil01]

- Do you agree with the remark?
- Why is this phenomenon?

Functions

- Functions of OS - to interpret and carry out commands issued by:
  - Users of the computer or,
  - Application programs running on the computer.

Purposes

- Two fundamental purposes:
  - Management:
    - To control and operate hardware in an efficient way.
  - Provide functionalities:
    - To allow the user **efficient access** to the facilities of the machine.
    - To allow the user **fair access** to the facilities of the machine.
    - To allow the user **protected access** to the facilities of the machine.
    - **Interaction** with the user.

Purposes – Onion ring model

- To realise the purposes of an operating system, the software system is often arranged in multiple layers.

- Why multiple layers?

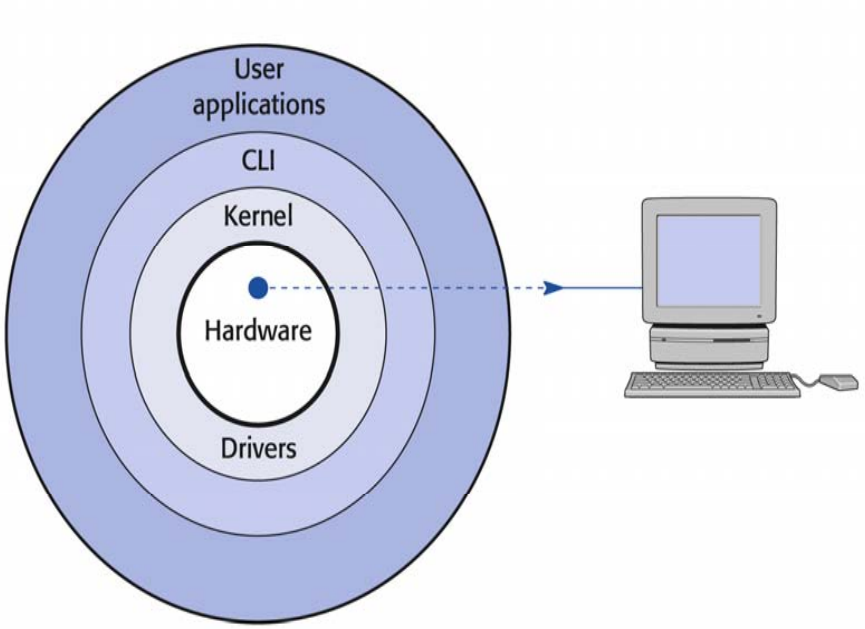


Fig. 2.14 Layers of software wrapping the hardware.

Purposes – Onion ring model (cont.)

- We use a computer through the operating system.

  (**gatekeeper**)
- It is often the operating system that governs the overall efficiency of a computer.

  (**butler**)

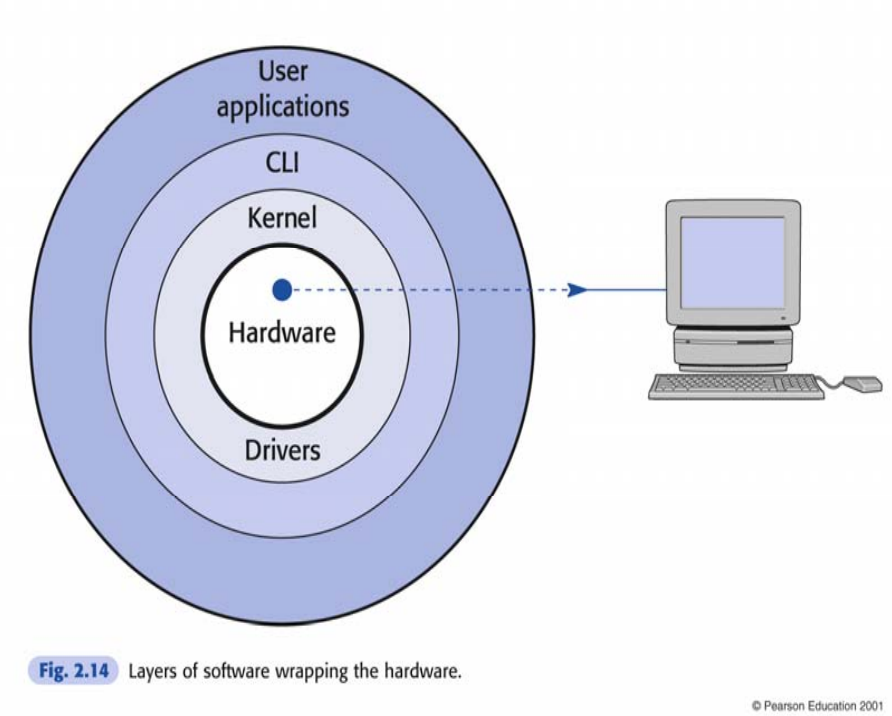


Fig. 2.14 Layers of software wrapping the hardware.

Purposes – Onion ring model (cont.)

- Core of operating system: dealing directly with the hardware.

  - **Kernel**: device drivers, memory allocator…

- CLI: provide user accessibilities to the system.

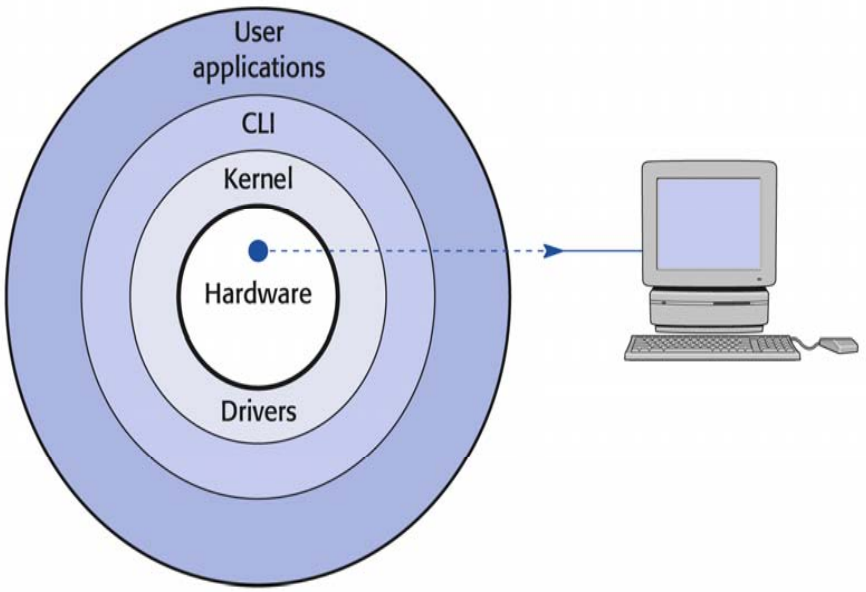


Fig. 2.14 Layers of software wrapping the hardware.

Modern operating Systems

- Many modern operating systems also:
  - Allow for the simultaneous processing of multiple programs.
    - DOS
      - To run a second program you have to wait the current one finishes. (Efficiency?)
      - Background spooling provides minimal degree of concurrency
    - Windows: Programs can run simultaneously, if they are not too resource consuming. (Same with Unix and Linux.) (**Multi-tasking**)
  - Support for multiple users. (**Multi-user**)

Interaction with operating system

- CLI (command line interpreter.)
  - DOS: type a command in a command line.
  - Unix/Linux: shell scripts (sequences of instructions).
  - Windows/Mac OS X: click with mouse on icons.
- Function calls from within user programs (API).
  - See, for example, the C code segment below.

```
#include <stdio.h>
…
void main(){
   printf(“Hello world!”);
}
```

Example

- Consider the situation, when you are doing your Java exercise.
  - To compile your Java program you need to run `javac` compiler
  - So, you type `javac MyProc.java` and press Return key.
  - What happens next?

Example (cont.)

- Observations:
  - `javac` is a program to be executed.
  - Any program should be in the main memory (RAM) to be executed.
- Thus, `javac` should be loaded into the main memory from the disk and then be executed.
- Question: who, or what does it?
  - Answer: The operating system.

One more example from Linux

- In Linux you type `ls` (similar to `dir` in DOS) command and press Return key.
  - The command lists the files in the current directory.
- When `ls` is running it needs information re the files in the current directory.
- Question: Who, or what provides such information?
  - Answer: OS. `ls` program makes requests to the appropriate system program to this end.

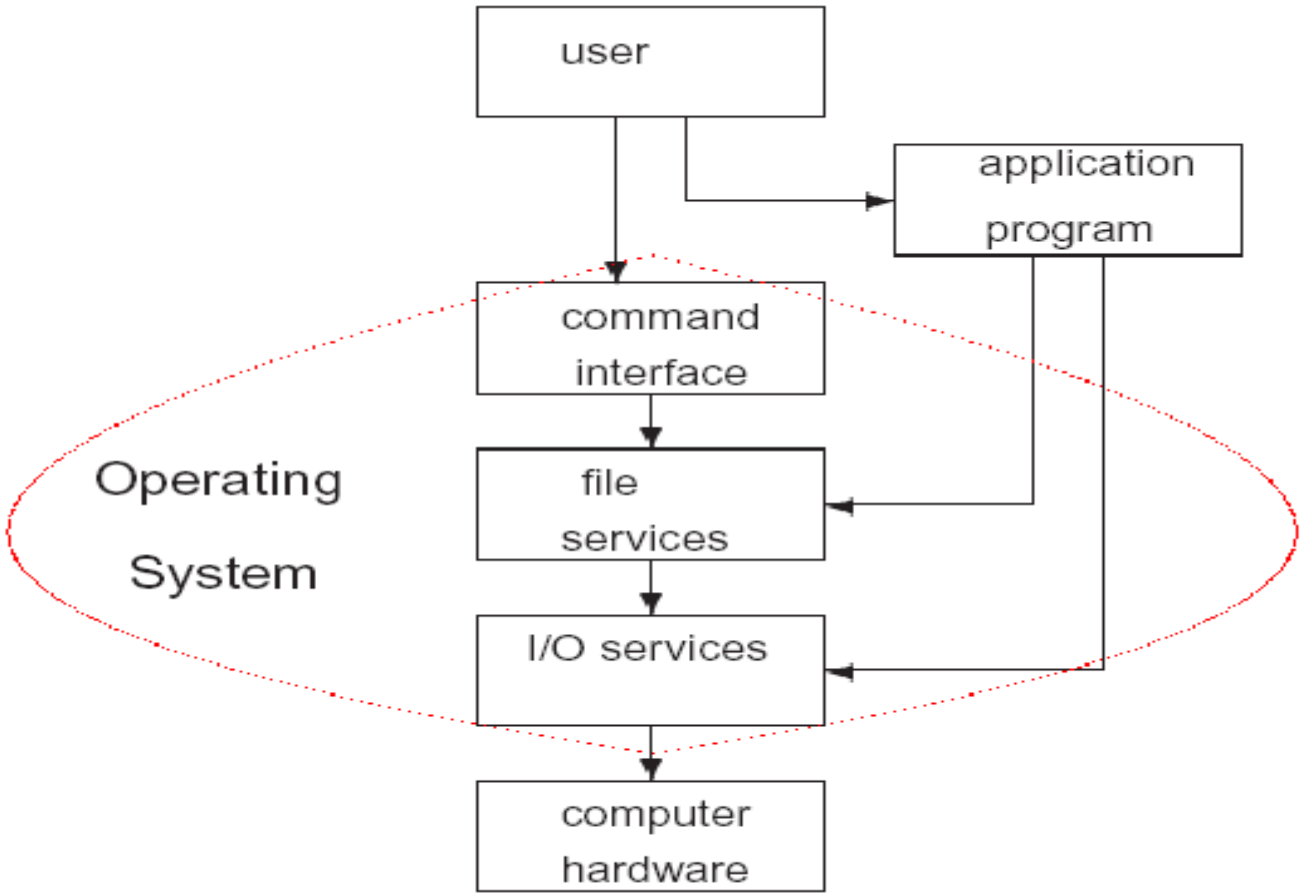
Complexity of OS operation

- Example: To accept a command from a user OS must:
  - Accept the input keystrokes from the keyboard.
  - Interpret this keystrokes as a command.
  - Determine the location of the file with a program for this command.
  - Read the appropriate blocks of file from (secondary storage) device into main memory.
  - Set up context for the program to be executed.
  - Transfer control to the program being executed.
  - Resume control when the program is finished.

Complexity of OS operation (cont.)

- If multiple programs are executing simultaneously, OS must include also:
  - Program to allocate memory and other resources to each program. (**memory manager**)
  - Program to allocate CPU time to each program (**scheduler**: more details later on).
  - Program to maintain integrity of each program. (**security kernel**)
  - Etc.
  - Can you see the complexity of tasks handled by OS?

Simplified picture of OS services



Computer Networks

- Perhaps the most far-reaching changes ever produced to von Neumann's original blueprint.
  - What changes?
- Operating system usually provides access to network facilities. (via networking API, e.g. socket interface)
- Computer network is an interconnected collection of autonomous computers to facilitate fast information exchange.

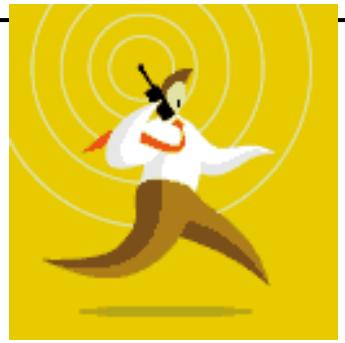
Why computer networks?

- To increase computing power.
  - Distributed computing projects: SETI@home, Folding@home.
- To share valuable resources (printers, large disks facilities, programs, data bases, etc).
  - The central printing system at XJTLU.
  - The central filing system at XJTLU.
- To organize convenient interactions of users working at their own computers, often from different locations.
  - Think about online games!
- Information at your fingertips…

Connectivity

- Connecting devices for communication
  - voice, data, multimedia
  - foundation for information age
- E-mail
  - send and receive messages over a local area network or a large network, including the Internet

Connectivity (continued)



- Telecommuting
  - Working at home or on the road
  - Communicating with the office through phone, fax, and/or computer
- On-line shopping, E- or M-commerce

Browsers and the World Wide Web

- Microsoft Internet Explorer and Netscape Navigator
  - two dominant Web browsers

Client-server computing

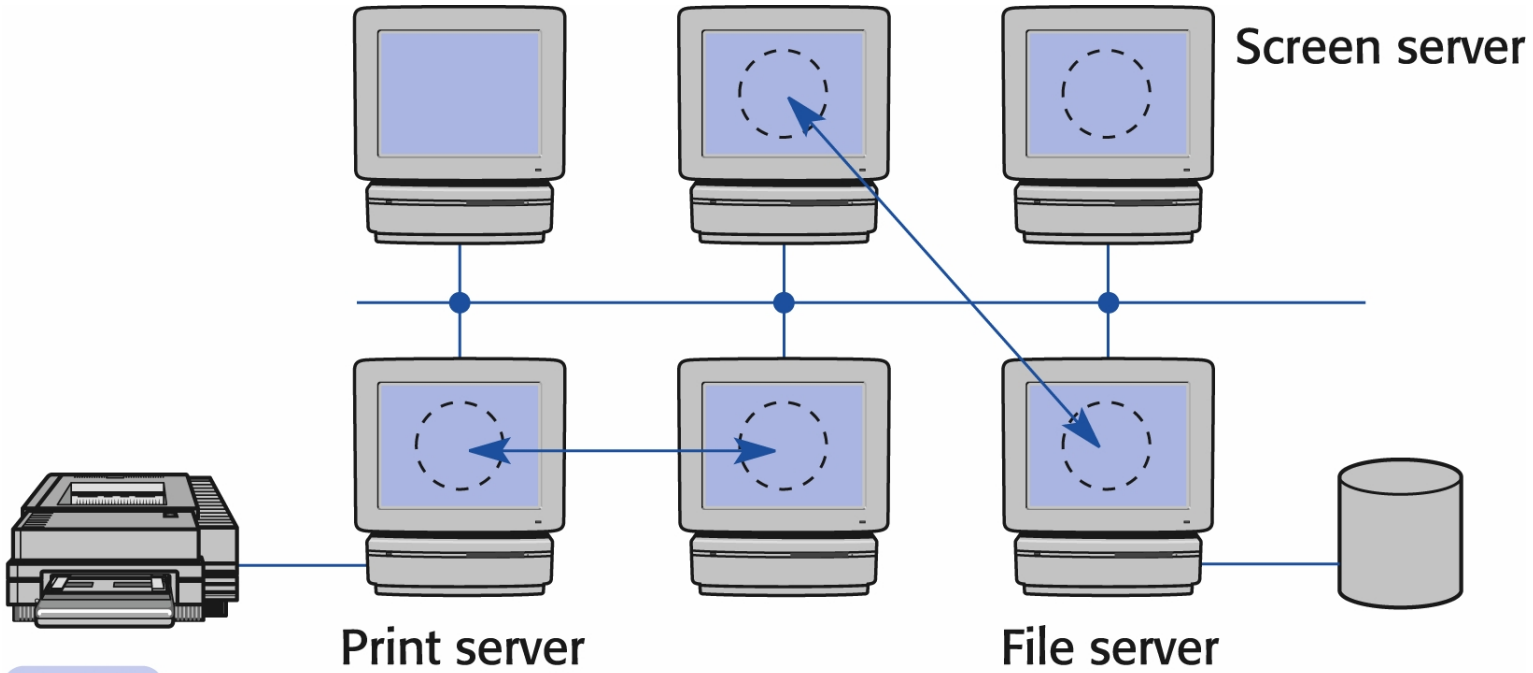


Fig. 2.15 Client–server couplets working across a network.

- **Client**: The originator of a request.
- **Server**: The supplier of the service.

Client-server interaction

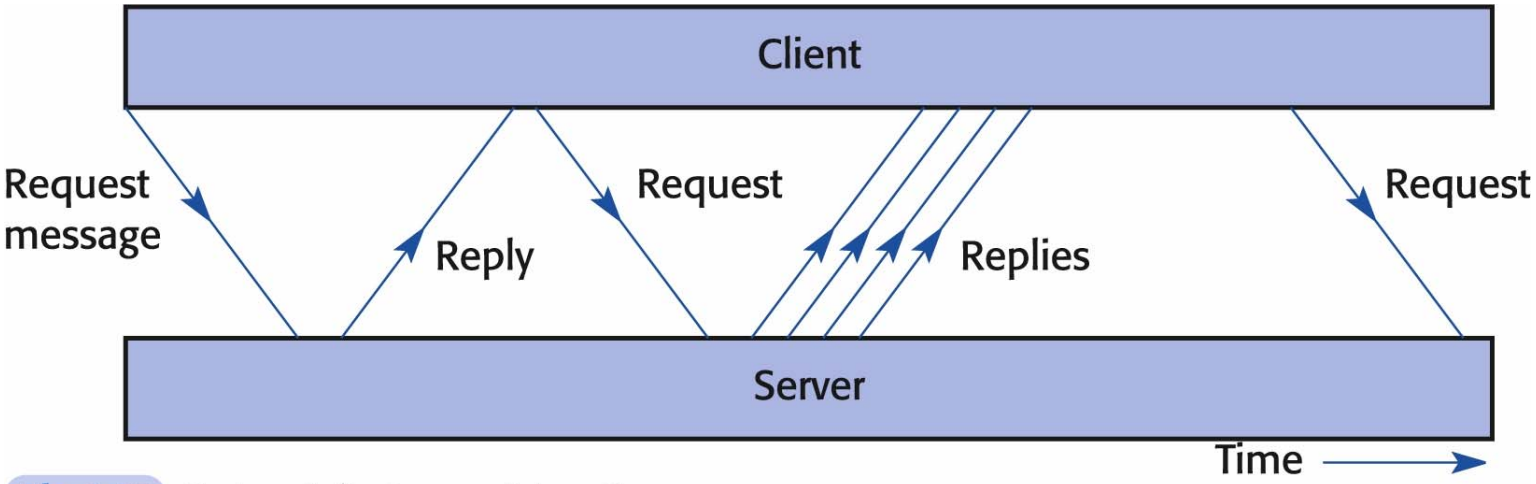


Fig. 2.16 Timing of client–server interaction.

- Client starts the interaction by sending a request message to the server.
- Server responds by sending replies back…
- Look at the status bar of your web browser when opening a web page from a remote site.

Exercise

- Demonstrate the concept of client-server computing using Web as an example.
  - Identify key entities under the Web
  - Explain how services are rendered via client-server computing
  - Pinpoint potential bottleneck under this Web scenario

Q&A

- What is the most important development during the past 40 years evolution of OS?
- OS needs to provide fair & protected access to system resources, why?
- Explain ‘multi-tasking’.
- Mention 3 functions that need to be provided when an OS is to support ‘multi-tasking’.
- Explain ‘multi-user’.

Q&A

- How does operating system provides access to network facilities?
- Telecommuting or online shopping are enabled through the introduction of ?

Readings

- [Wil06] Chapter 2, sections 2.9 - 2.10.