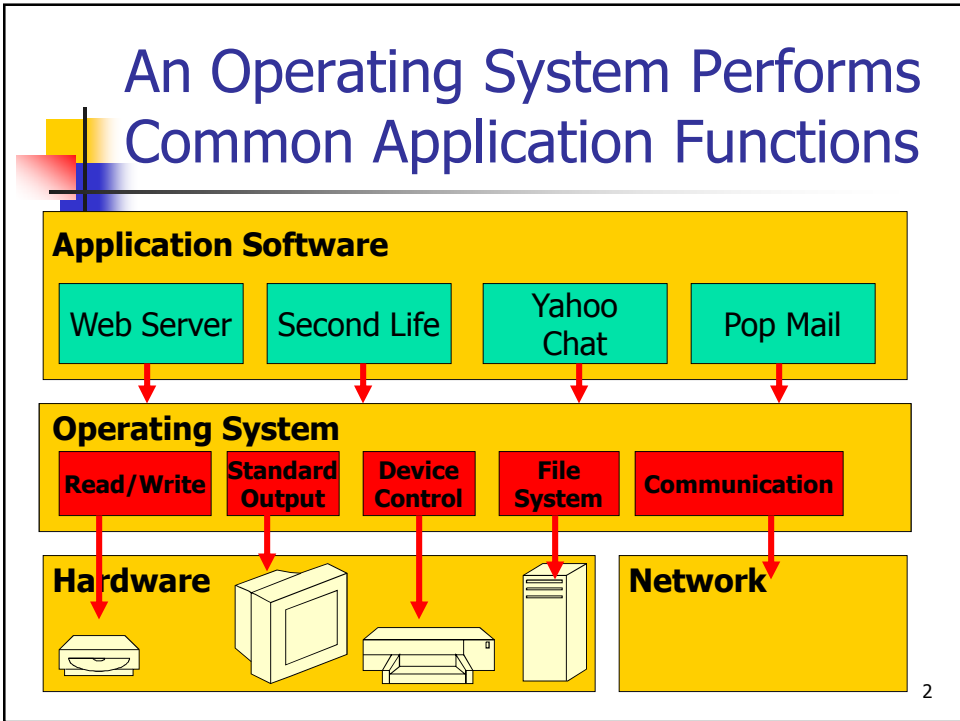
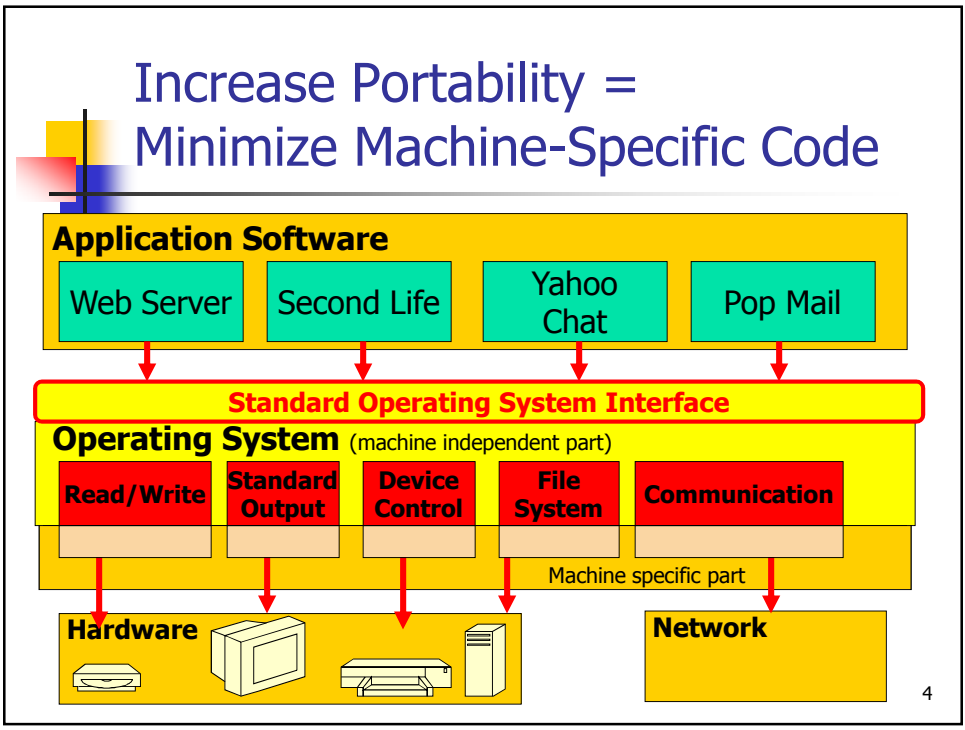
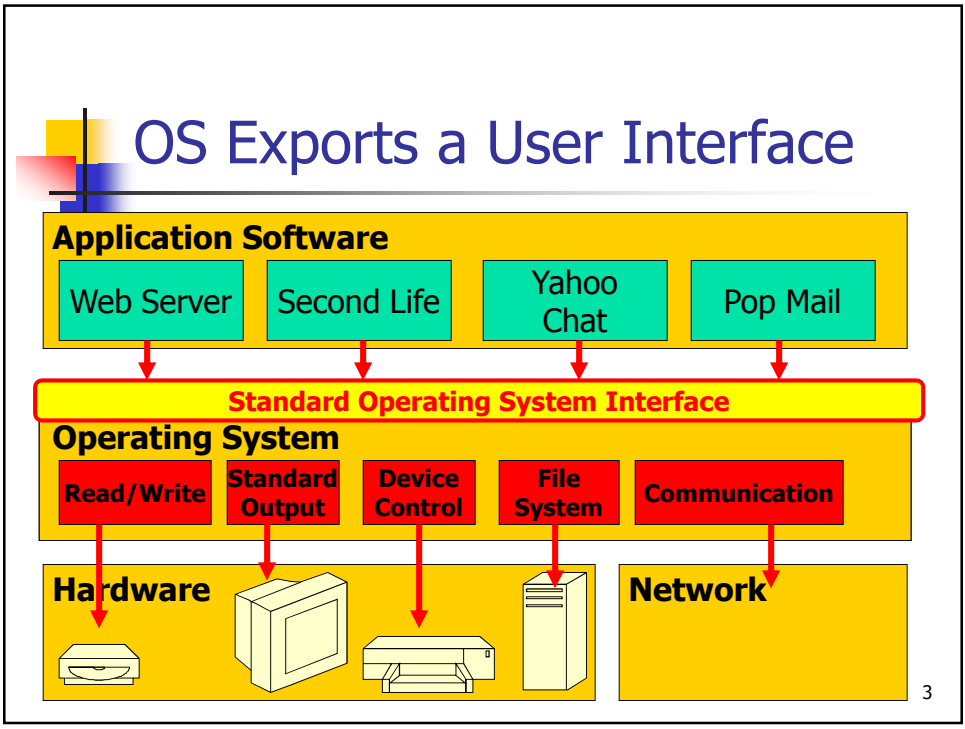


Operating Systems Overview

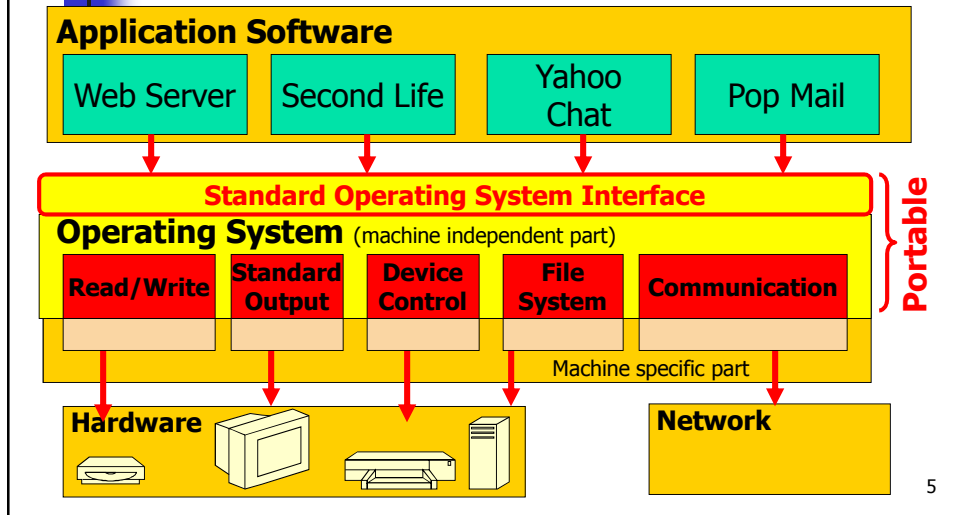
Intro, Sys Calls, Threads, etc.
Tarek Abdelzaher

1

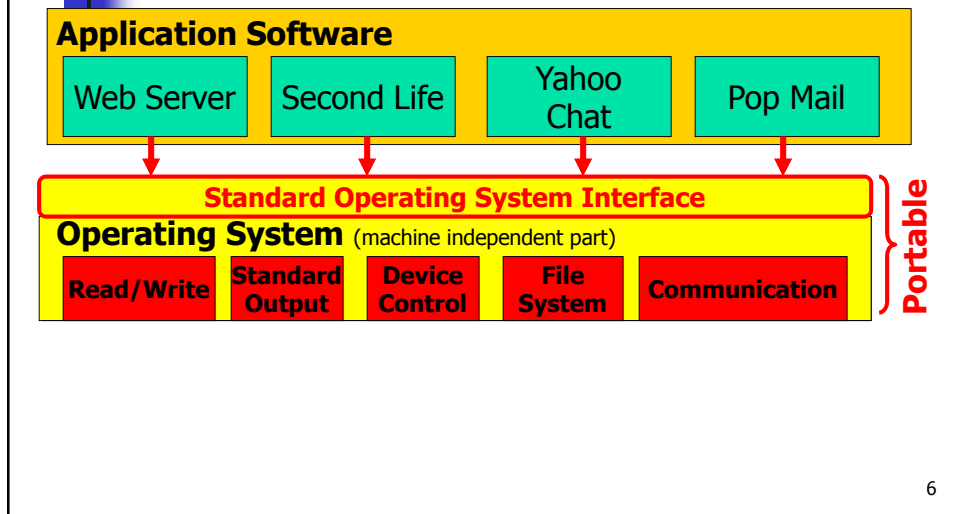




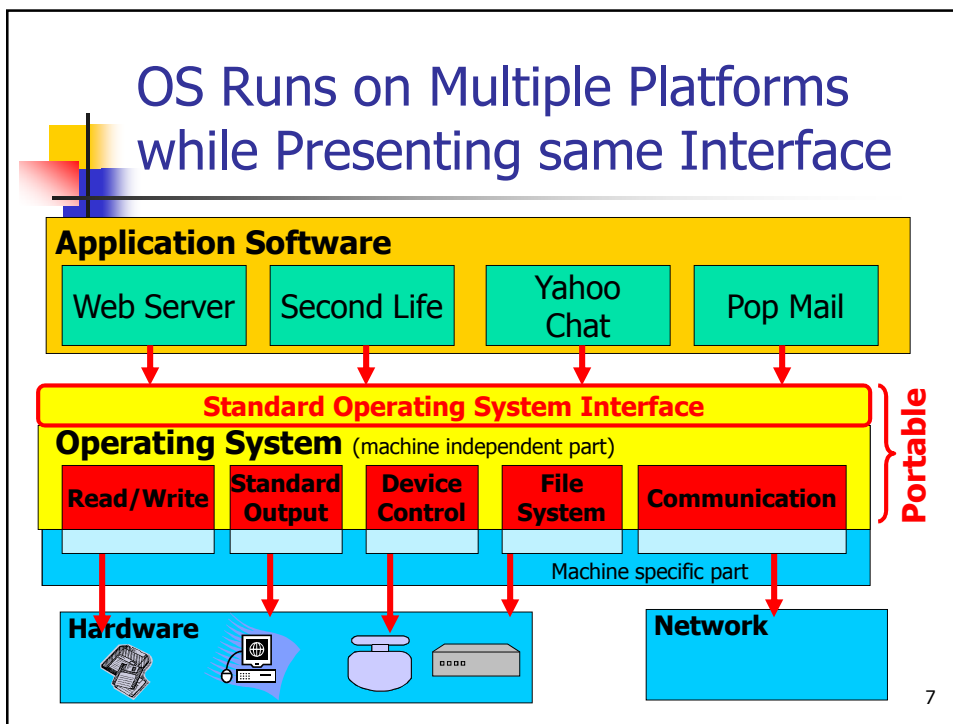
Increase Portability = Minimize Machine-Specific Code



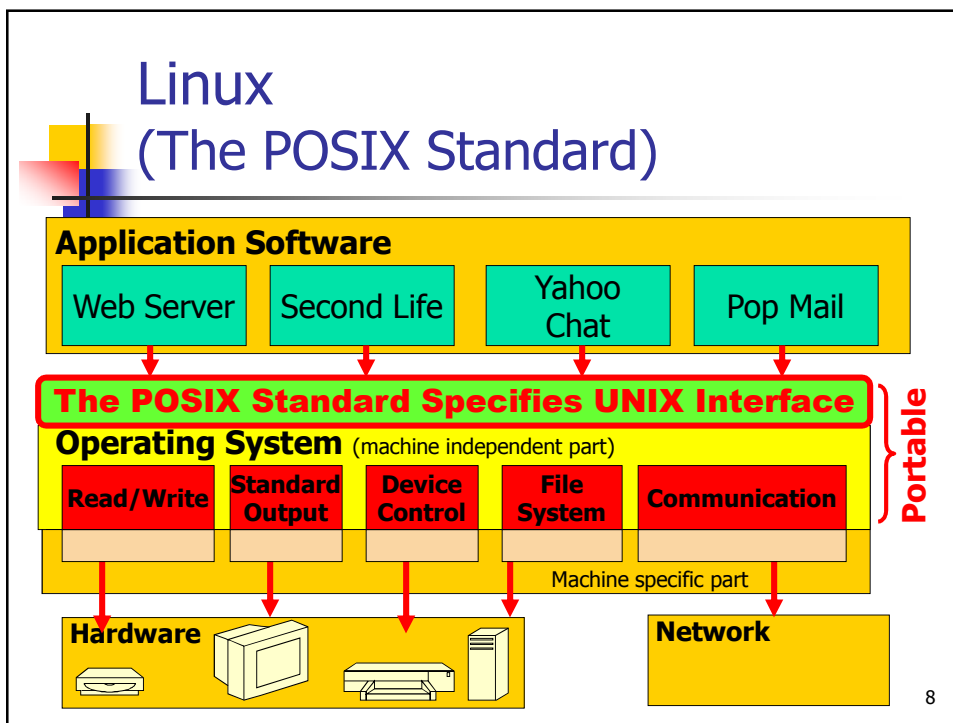
OS Runs on Multiple Platforms while Presenting same Interface



OS Runs on Multiple Platforms while Presenting same Interface



Linux (The POSIX Standard)

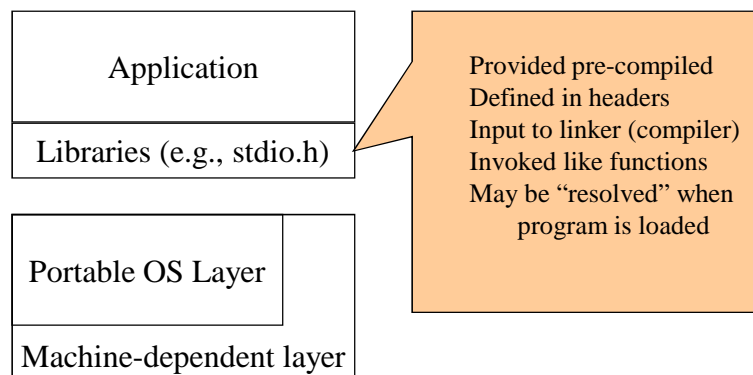


OS Goals

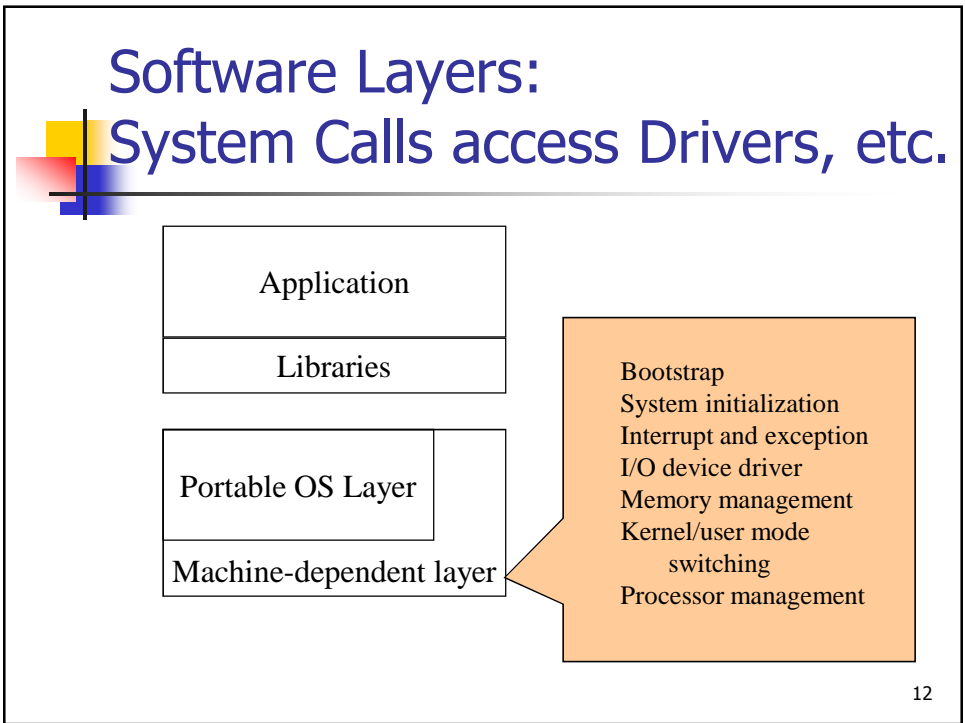
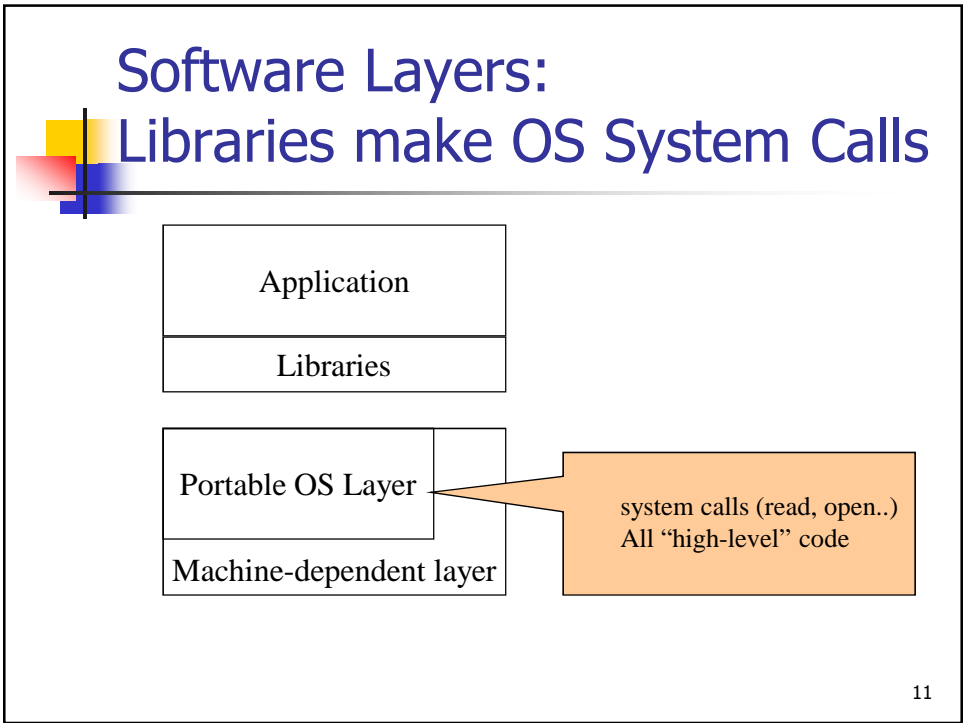
- 1) Provide an interface for applications to do common functions.


9

Software Layers: Applications call Libraries



10

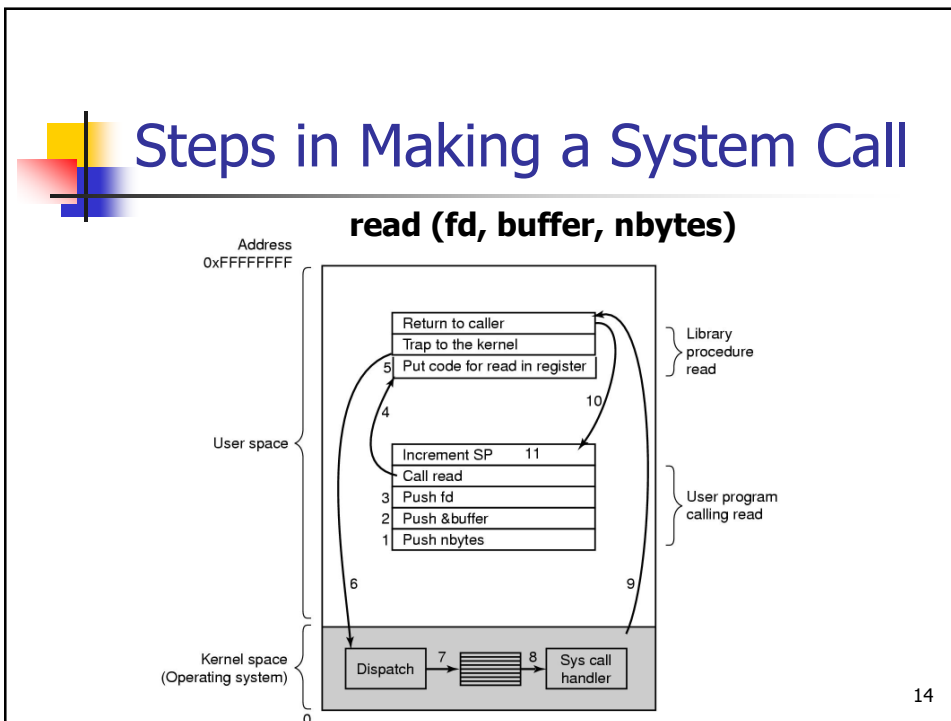




MP1

- Familiarize yourself with installing/compiling an OS and changing its interface (System Calls)

13





Some System Calls For Process Management

Process management

Call	Description
pid = fork()	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

15



Some System Calls For File Management

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

16




Some System Calls For Directory Management

Directory and file system management

Call	Description
s = mkdir(name, mode)	Create a new directory
s = rmdir(name)	Remove an empty directory
s = link(name1, name2)	Create a new entry, name2, pointing to name1
s = unlink(name)	Remove a directory entry
s = mount(special, name, flag)	Mount a file system
s = umount(special)	Unmount a file system

17




Some System Calls For Miscellaneous Tasks

Miscellaneous

Call	Description
s = chdir(dirname)	Change the working directory
s = chmod(name, mode)	Change a file's protection bits
s = kill(pid, signal)	Send a signal to a process
seconds = time(&seconds)	Get the elapsed time since Jan. 1, 1970


18



OS Goals

- Provide an interface for applications to do common functions.

19



OS Goals

- Provide an interface for applications to do common functions.
- Manage resources

20

Managed Resources - CPU

The diagram shows a computer system architecture. At the top, there is a horizontal line representing a bus. Below this bus, several components are connected: a yellow box labeled 'CPU', a white box labeled 'Memory', a white box labeled 'Video controller' connected to a 'MONITOR' icon, a white box labeled 'Keyboard controller' connected to a 'Keyboard' icon, a white box labeled 'Floppy disk controller' connected to a 'Floppy disk drive' icon, and a white box labeled 'Hard disk controller' connected to a 'Hard disk drive' icon. The 'CPU' box is highlighted in yellow, indicating it is the managed resource.

Process/thread scheduling, context switching, etc.

21

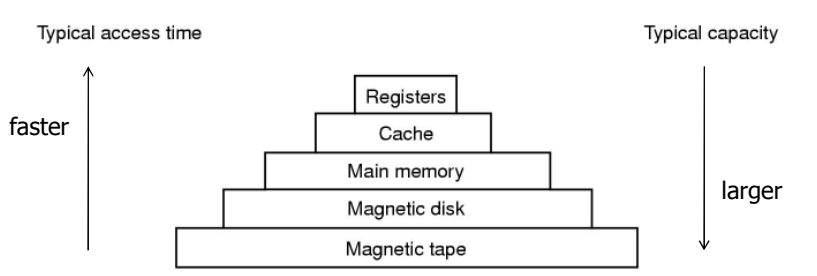
Managed Resource - Memory

The diagram shows the same computer system architecture as above. In this version, the 'Memory' box is highlighted in yellow, indicating it is the managed resource.

Paging, swapping, virtual memory management, etc.

22

Memory Hierarchy

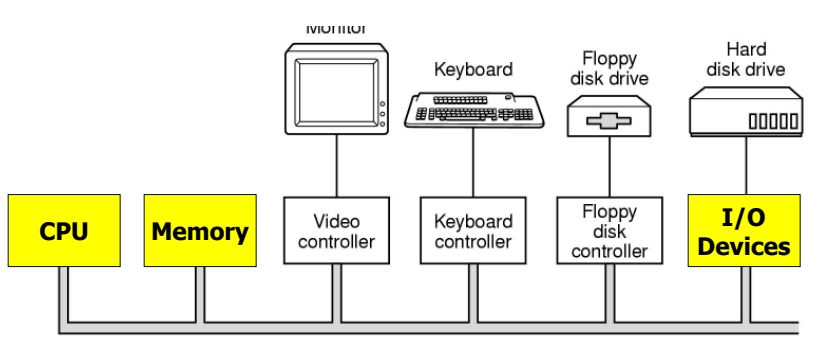


The diagram shows a pyramid representing the memory hierarchy. From top to bottom, the levels are: Registers, Cache, Main memory, Magnetic disk, and Magnetic tape. To the left of the pyramid, an upward-pointing arrow is labeled 'faster' and 'Typical access time'. To the right, a downward-pointing arrow is labeled 'larger' and 'Typical capacity'.

- Make it look like it's as large as the disk and as fast as the cache

23


Managed Resource – I/O



The diagram illustrates the I/O system architecture. At the top, there are icons for a Monitor, Keyboard, Floppy disk drive, and Hard disk drive. Below these are their respective controllers: Video controller, Keyboard controller, Floppy disk controller, and I/O Devices. The CPU and Memory are shown as yellow boxes on the left. All components are connected to a common horizontal bus at the bottom.

Drivers, interrupts, timers, files, etc.


24



OS Goals

- Provide an interface for applications to do common functions.
- Manage resources


25



OS Goals

- Provide an interface for applications to do common functions.
- Manage resources
- Structure concurrent application execution


26



Threads and Processes

Per process items	Per thread items
Address space	Program counter
Global variables	Registers
Open files	Stack
Child processes	State
Pending alarms	
Signals and signal handlers	
Accounting information	

27



Things Suitable for threading

- Block for potentially long waits
- Use many CPU cycles
- Respond to asynchronous events
- Execute functions of different importance
- Execute parallel code

28

Thread Usage: Word Processor

The diagram illustrates a single-threaded architecture for a word processor. A central box labeled 'Kernel' contains a circle with three wavy lines representing a single thread. This thread is connected to a 'Keyboard' on the left and a 'Disk' on the right. Above the kernel box, there is a large block of placeholder text consisting of several columns of small, illegible text.

- What if it is single-threaded?

29

Thread Usage: Web Server

The diagram shows a 'Web server process' box divided into 'User space' and 'Kernel space'. In the 'User space', there is a circle containing a 'Dispatcher thread' and several 'Worker threads'. A 'Web page cache' is represented as a rectangular block within this circle. In the 'Kernel space', there is a 'Kernel' label. A 'Network connection' is shown as a line entering from the bottom left, passing through the kernel space into the dispatcher thread in the user space.

30

Common Multi-thread Software Architectures

■ **Manager/worker**

- a single thread, the *manager* assigns work to other threads, the *workers*. Typically, the manager handles all input and parcels out work to the other tasks

■ **Pipeline**

- a task is broken into a series of sub-operations, each of which is handled by a different thread. An automobile assembly line best describes this model

■ **Peer**

- similar to the manager/worker model, but after the main thread creates other threads, it participates in the work.³¹

User-level Threads

■ **Advantages**

- Fast Context Switching:
 - User level threads are implemented using **user level thread libraries**, rather than system calls, hence no call to OS and no interrupts to kernel
 - When a thread is finished running for the moment, it can call **thread_yield**. This instruction (a) saves the thread information in the thread table, and (b) calls the thread scheduler to pick another thread to run.
 - The procedure that saves the local thread state and the scheduler are **local procedures**, hence no trap to kernel, no context switch, no memory switch, and this makes the **thread scheduling very fast**.
- Customized Scheduling

32



Kernel-level Threads

- Kernel can schedule threads in addition to processes.
- Multiple threads of a process can run simultaneously on multiple CPUs.
- Synchronization more efficient than for processes (but less than for user-level threads).
- Kernel-level threads can make blocking I/O calls without blocking other threads of same process

33