

# Operating System Structure

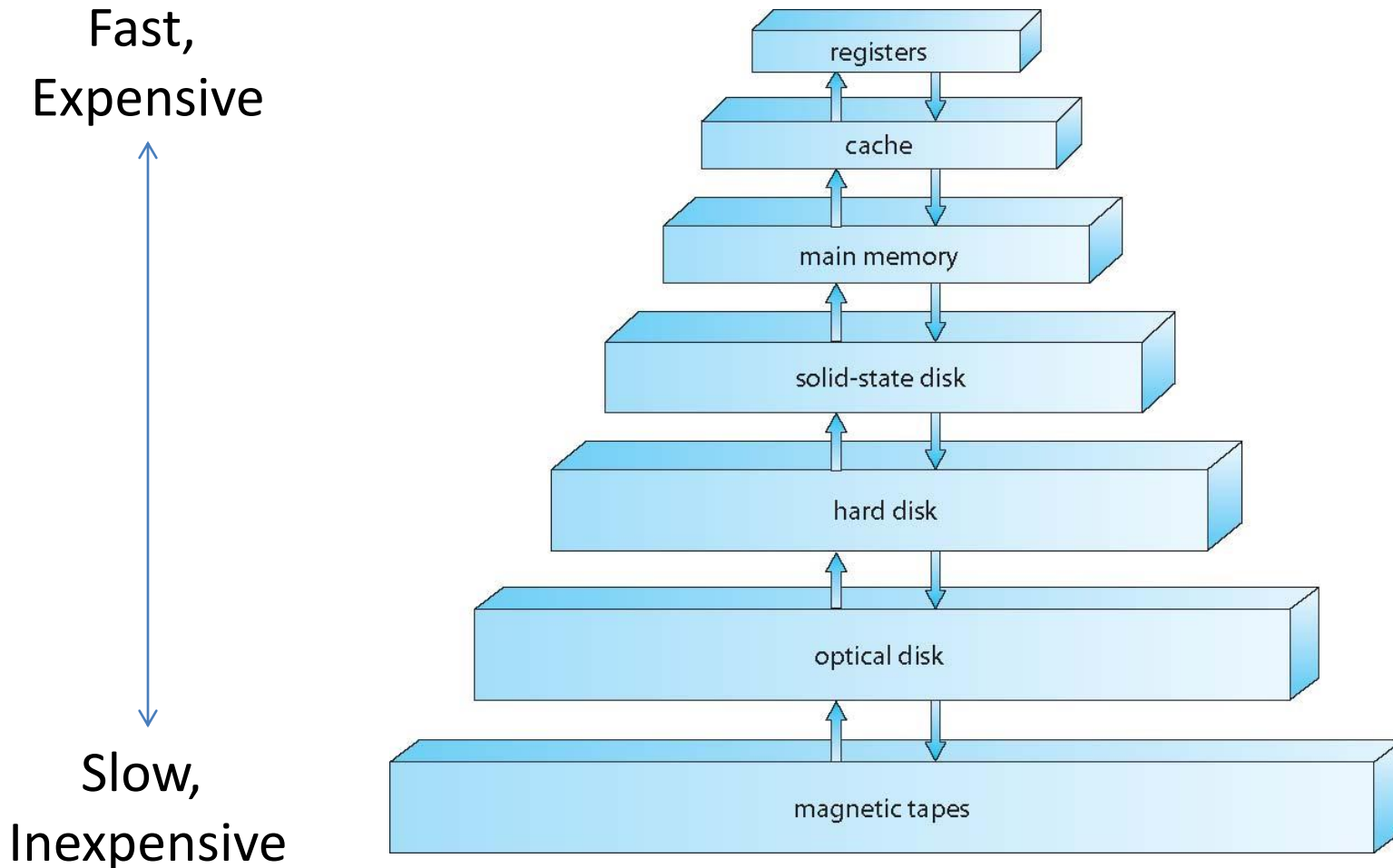
Heechul Yun

Disclaimer: some slides are adopted from the book authors' slides with permission

# Recap

- OS needs to understand architecture
  - Hardware (CPU, memory, disk) trends and their implications in OS designs
- Architecture needs to support OS
  - Interrupts and timer
  - User/kernel mode and privileged instructions
  - MMU
  - Synchronization instructions

# Recap: Memory Hierarchy



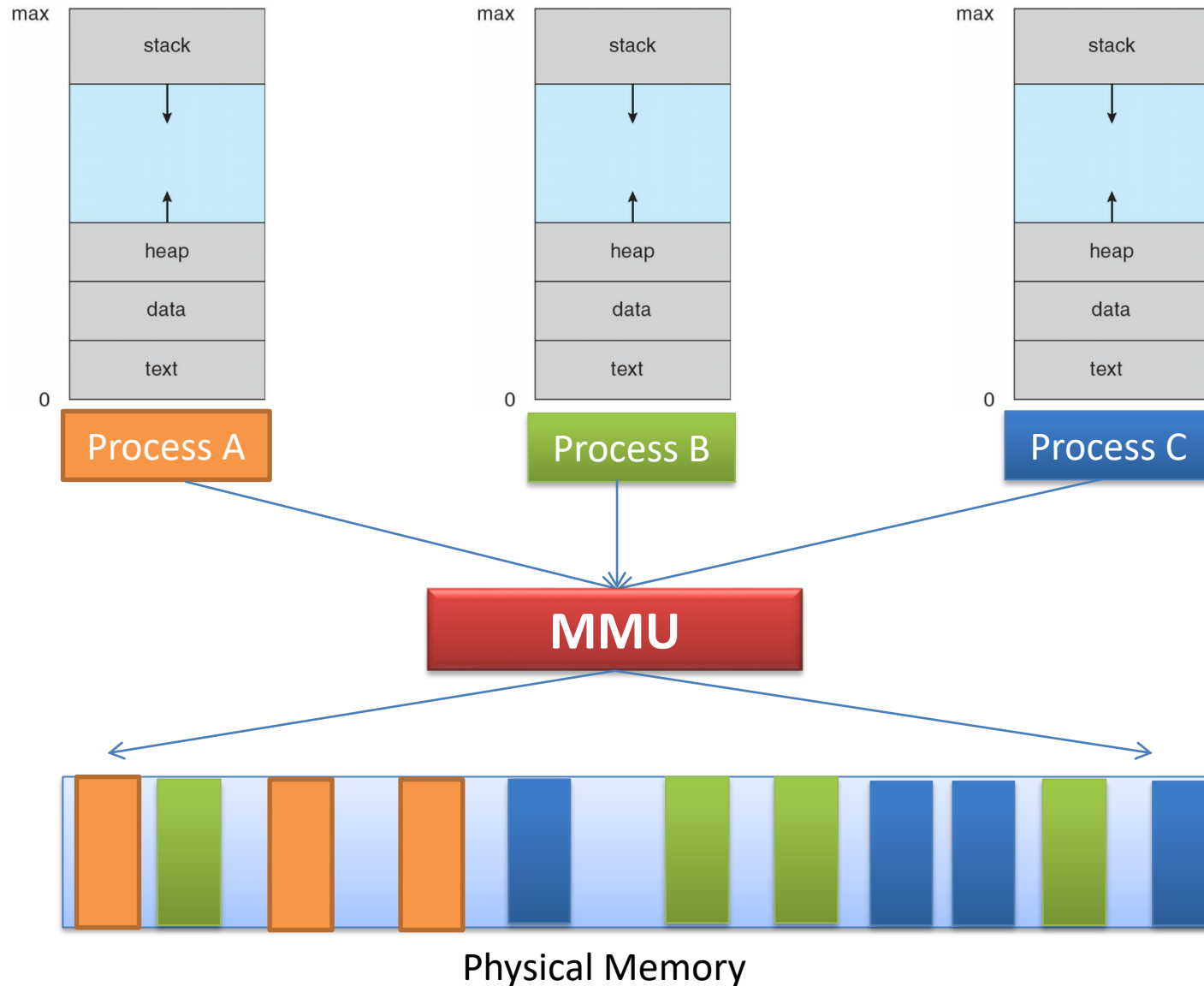
# Memory Protection

- How to protect memory among apps/kernel?
  - Applications shouldn't be allowed to access kernel's memory
  - An app shouldn't be able to access another app's memory

# Virtual Memory

- How to overcome memory space limitation?
  - Multiple apps must share limited memory space
  - But they want to use memory as if each has dedicated and big memory space
  - E.g.,) 1GB physical memory and 10 programs, each of which wants to have a linear 4GB address space

# Virtual Memory



# MMU

- Hardware unit that translates *virtual address* to *physical address*
  - Defines the boundaries of kernel/apps
  - Enable efficient use of physical memory



# Synchronization

- Synchronization problem with threads

```
Deposit(account, amount) {  
  {  
    account->balance += amount;  
  }  
}
```

**Thread 1:** *Deposit(acc, 10)*

LOAD R1, account->balance

ADD R1, amount

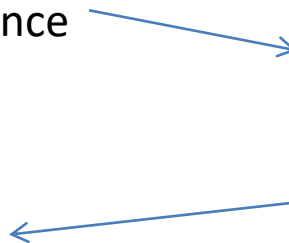
STORE R1, account->balance

**Thread 2:** *Deposit(acc, 10)*

LOAD R1, account->balance

ADD R1, amount

STORE R1, account->balance





# Synchronization Instructions

- Hardware support for synchronization
  - *TestAndSet*, *CompareAndSwap* instructions
  - Atomic load and store
  - Used to implement **lock** primitives
  - New TSX instruction → hardware transaction
- Another methods to implement locks in single-core systems
  - Disabling interrupts

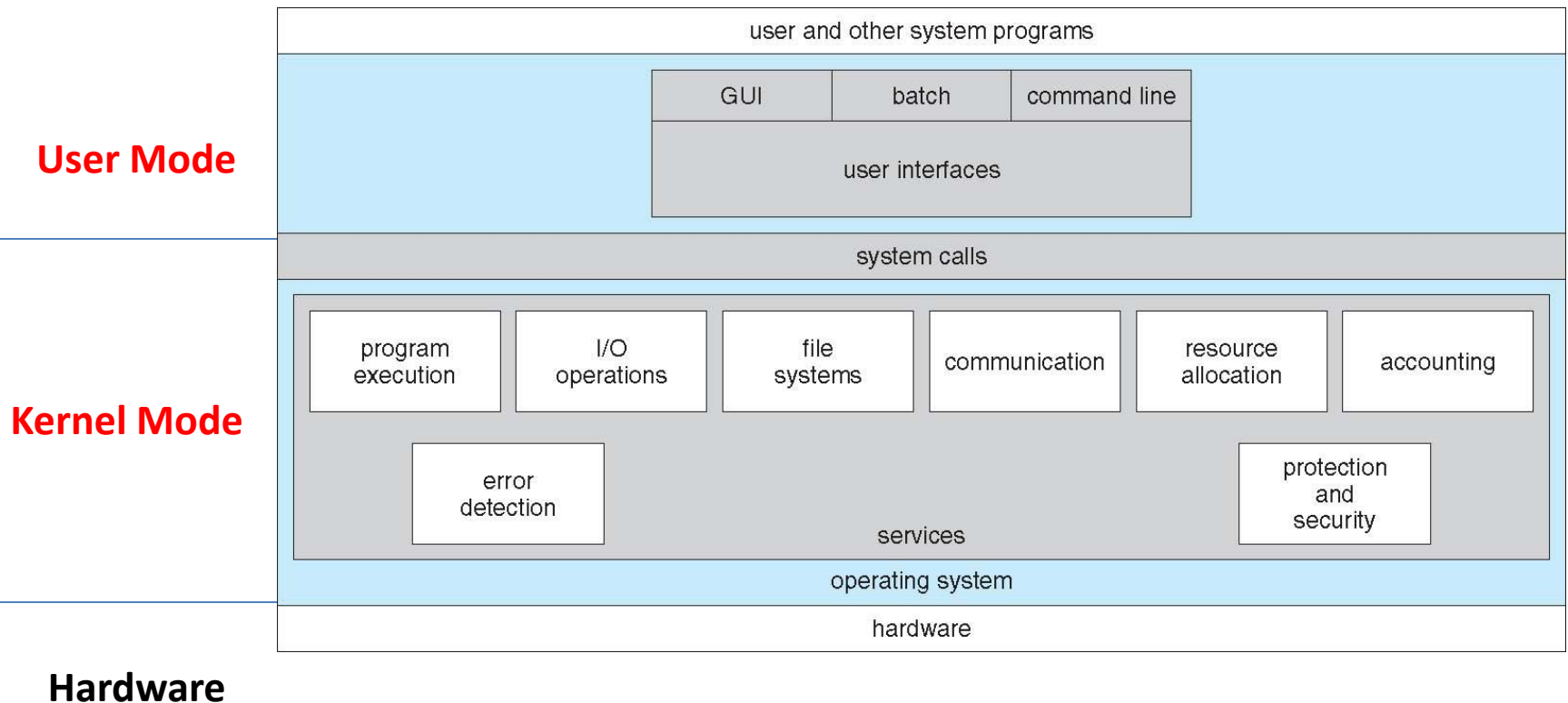
# Summary

- OS needs to understand architecture
  - Hardware (CPU, memory, disk) trends and their implications in OS designs
- Architecture needs to support OS
  - Interrupts and timer
  - User/kernel mode and privileged instructions
  - MMU
  - Synchronization instructions

# Today

- OS services
  - User's perspective
  - What are the major functionalities of an OS?
- OS interface
  - How applications interact with the OS?
- OS structure
  - What are possible structures of an OS?

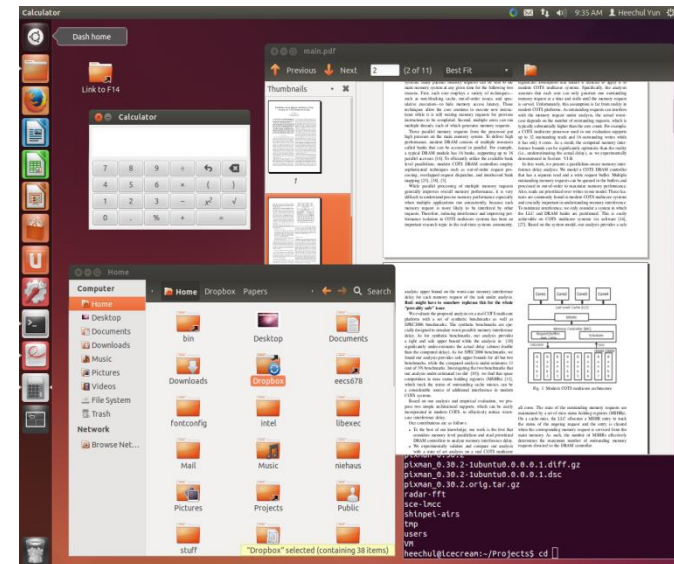
# A View of Operating System Services



# Operating System Services

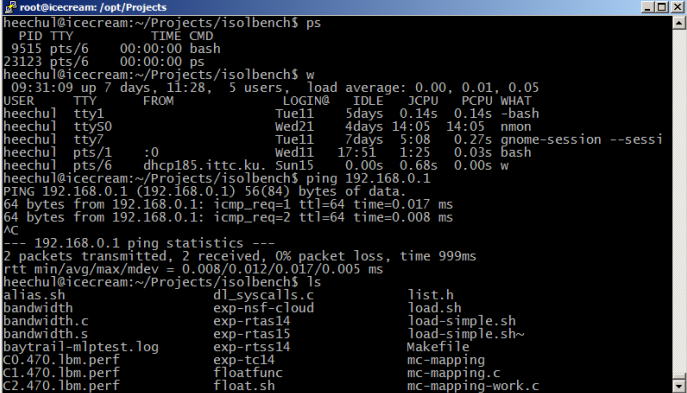
- User interface
  - Command-Line Interface (CLI) vs. Graphical User Interface (GUI)

```
root@icecream: /opt/Projects
heechul@icecream:~/Projects/isolbench$ ps
  PID TTY          TIME CMD
 9515 pts/6    00:00:00 bash
23123 pts/6    00:00:00 ps
heechul@icecream:~/Projects/isolbench$ w
 09:31:09 up 7 days, 11:28,  5 users,  load average: 0.00, 0.01, 0.05
USER      TTY      FROM             LOGIN@   IDLE   JCPU   PCPU   WHAT
heechul   tty1     Tue11            5days   0.14s  0.14s  -bash
heechul   ttyS0    Wed21            4days   14:05  14:05  nmon
heechul   tty7     Tue11            7days   5:08   0.27s  gnome-session --sessi
heechul   pts/1    :0               Wed11    17:51  1:25   0.03s  bash
heechul   pts/6    dhcp185.ittc.ku. Sun15     0.00s   0.68s  0.00s  w
heechul@icecream:~/Projects/isolbench$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data:
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=0.017 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=0.008 ms
AC
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.008/0.012/0.017/0.005 ms
heechul@icecream:~/Projects/isolbench$ ls
alias.sh          dl_syscalls.c      list.h
bandwidth         exp-nsf Cloud      load.sh
bandwidth.c       exp-rtas14         load-simple.sh
bandwidth.s       exp-rtas15         load-simple.sh~
baytrail-mpptest.log exp-rtss14         Makefile
c0.470.lbm.perf   exp-tc14           mc-mapping
c1.470.lbm.perf   floatfunc          mc-mapping.c
c2.470.lbm.perf   float.sh           mc-mapping-work.c
```



# Command-Line Interface (CLI)

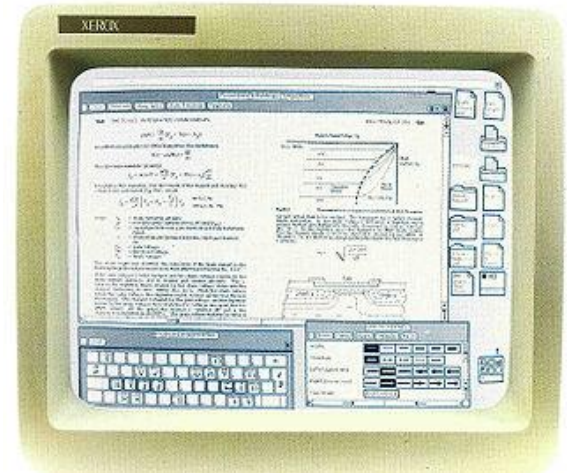
- Command-line interpreter (shell)
  - Many flavors: bash, csh, ksh, tcsh, ...
  - Usually not part of the kernel, but an essential system program
- Allow users to enter text commands
  - Some commands are built-in
    - E.g., alias, echo, read, source, ...
  - Some are external programs
    - E.g., ls, find, grep, ...
- Pros and Cons.
  - + Easy to implement, use less resources, easy to access remotely
  - + Easy to **automate**
    - E.g., `$ grep bandwidth /tmp/test.txt | awk '{ print $2 }'`
  - Difficult to learn



```
root@icecream: /opt/Projects
heechul@icecream:~/Projects/isolbench$ ps
PID TTY TIME CMD
9515 pts/6 00:00:00 bash
23123 pts/6 00:00:00 ps
heechul@icecream:~/Projects/isolbench$ w
09:31:09 up 7 days, 11:28, 5 users, load average: 0.00, 0.01, 0.05
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
heechul tty1 Tue11 5days 0.14s 0.14s -bash
heechul tty50 Wed21 4days 14:03 14:05 nmon
heechul tty7 Tue11 7days 5:08 0.27s gnome-session --sessi
heechul pts/1 :0 Wed11 17:51 1:25 0.03s bash
heechul pts/6 dhcp185.ittc.ku. Sun15 0.00s 0.68s 0.00s w
heechul@icecream:~/Projects/isolbench$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data:
64 bytes from 192.168.0.1: icmp_req=1 ttl=64 time=0.017 ms
64 bytes from 192.168.0.1: icmp_req=2 ttl=64 time=0.008 ms
AC
--- 192.168.0.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.008/0.012/0.017/0.005 ms
heechul@icecream:~/Projects/isolbench$ ls
alias.sh dl_syscalls.c list.h
bandwidth exp-nsf-cloud load.sh
bandwidth.c exp-rtas14 load-simple.sh
bandwidth.sh exp-rtas15 load-simple.sh~
baytrail-mptest.log exp-rtss14 Makefile
C0.470.1bm.perf exp-rtcl4 mc-mapping
C1.470.1bm.perf floatfunc mc-mapping.c
C2.470.1bm.perf float.sh mc-mapping-work.c
```

# Graphic User Interface (GUI)

- GUI
  - Mouse, keyboard, monitor
  - Invented at Xerox PARC, then adopted to Mac, Window,...
- Pros and Cons
  - + Easy to use
  - Use more h/w resources
- Many systems support both CLI and GUI



The first commercial GUI from Xerox Star workstation. (source: Wikipedia)

# Operating System Services

- File-system service
  - Read/write /create/delete/search files and directories
  - See file information (e.g., file size, creation time, access time, ...)
  - Permission management (read only, read/write, ...)
- Communications
  - Share information between processes in the same computer (Inter-process communication - IPC) or between computers over a network (TCP/IP)



# Operating System Services

- Resource allocation
  - CPU cycles, main memory space, file space, I/O devices
- Accounting
  - Keeping track of who uses what for how much
- Security
  - Login, administrators vs. normal users vs. guests

# Operating System Services

- Protection
  - Prevent memory corruption between multiple user programs and between user programs and the kernel
  - Detect and report errors
    - Divide by zero, access violation, hardware faults, ...

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

***      gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.
```

# Recap

- OS services
  - CPU scheduling
  - Memory management
  - Filesystem
  - Communication
  - Protection & security
  - Device drivers

# System Calls

- Programming interface to the services provided by the OS
- Typically written in a high-level language (C)
- Most programmers do not directly use system calls
  - They use more high level APIs (i.e., **libraries**)
  - But **system programmers** (or you) do use system calls
- Two most popular system call APIs
  - **Win32 API** for Windows
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X)

# Example

- Copy the contents of one file to another file

```
int main(int argc, char *argv[])
{
    int src_fd, dst_fd; char buf[80]; int len;

    src_fd = open(argv[1], O_RDONLY);
    dst_fd = open(argv[2], O_WRONLY|O_CREAT|O_TRUNC);

    while ((len = read(src_fd, buf, 80)) > 0) {
        write(dst_fd, buf, len);
    }

    printf("Done\n");
    return 0;
}
```

# Example

- Copy the contents of one file to another file

```
int main(int argc, char *argv[])
{
    int src_fd, dst_fd; char buf[80]; int len;

    src_fd = open(argv[1], O_RDONLY);
    dst_fd = open(argv[2], O_WRONLY|O_CREAT|O_TRUNC);

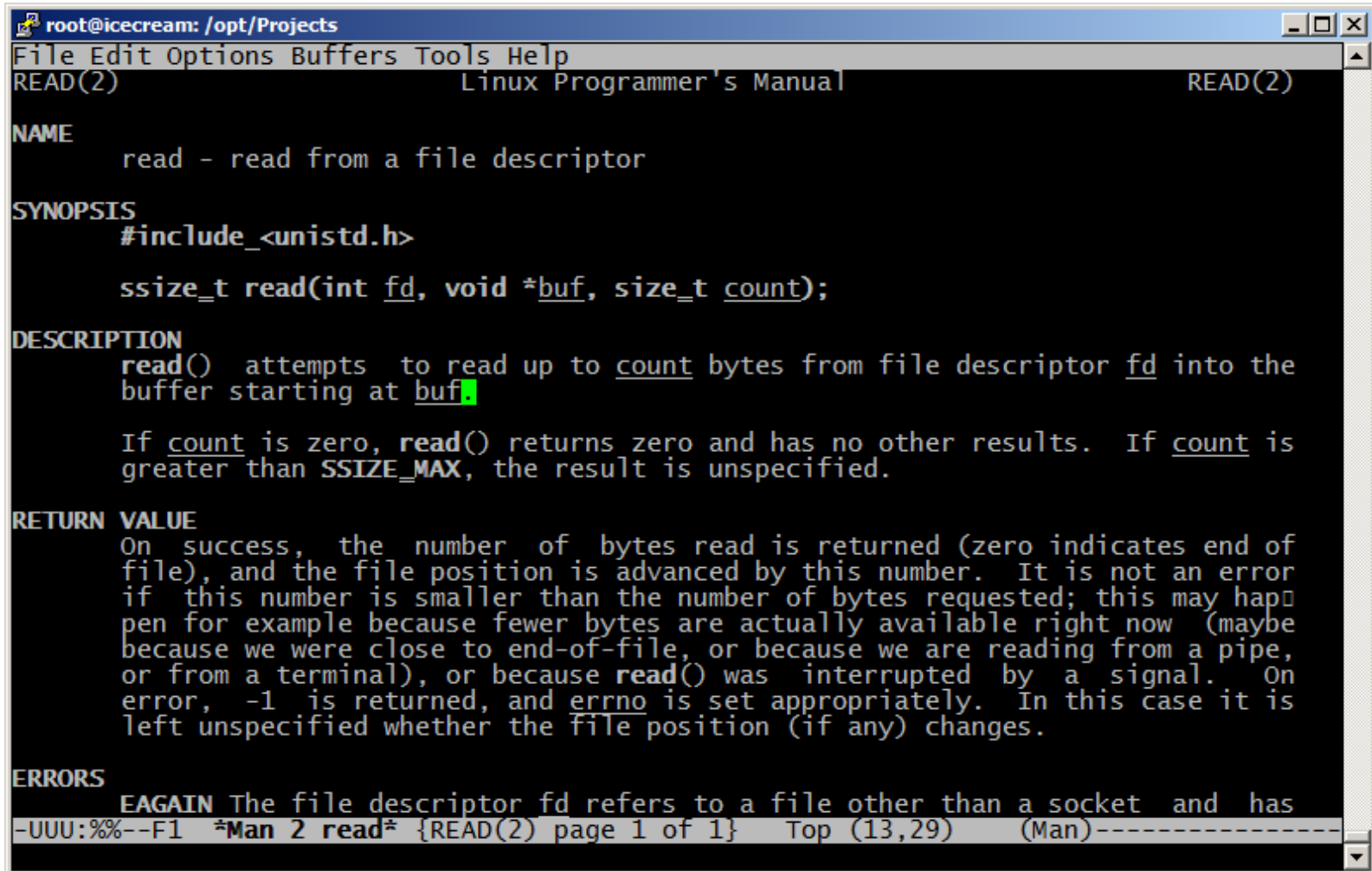
    while ((len = read(src_fd, buf, 80)) > 0) {
        write(dst_fd, buf, len);
    }

    printf("Done\n");
    return 0;
}
```

Syscalls: **open**, **read**, **wrtie**  
Non-syscall: **printf**

# System Call API Description

- \$ man 2 read



```
root@icecream: /opt/Projects
File Edit Options Buffers Tools Help
READ(2)                                Linux Programmer's Manual                                READ(2)

NAME
    read - read from a file descriptor

SYNOPSIS
    #include <unistd.h>

    ssize_t read(int fd, void *buf, size_t count);

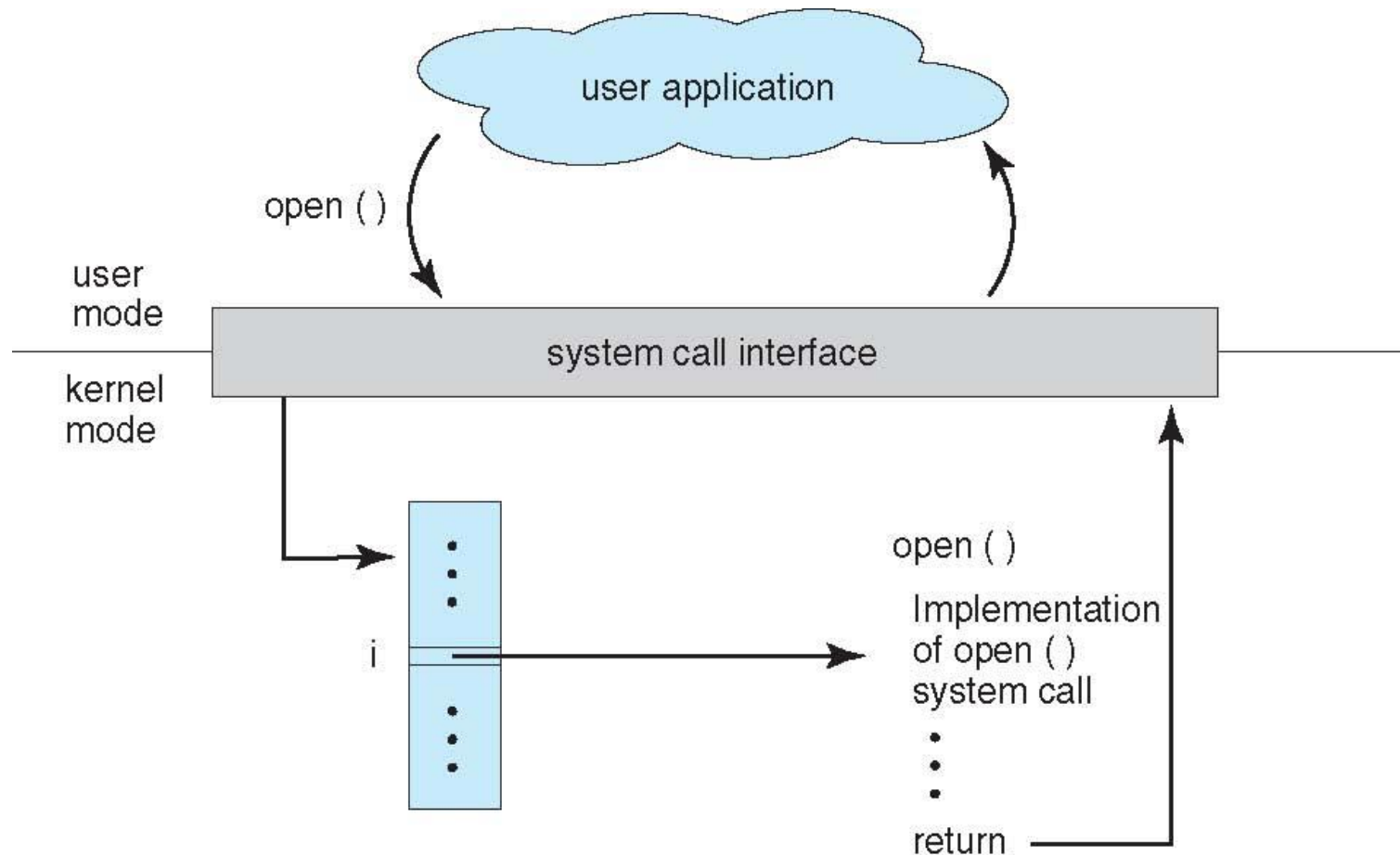
DESCRIPTION
    read() attempts to read up to count bytes from file descriptor fd into the
    buffer starting at buf.

    If count is zero, read() returns zero and has no other results. If count is
    greater than SSIZE_MAX, the result is unspecified.

RETURN VALUE
    On success, the number of bytes read is returned (zero indicates end of
    file), and the file position is advanced by this number. It is not an error
    if this number is smaller than the number of bytes requested; this may hap-
    pen for example because fewer bytes are actually available right now (maybe
    because we were close to end-of-file, or because we are reading from a pipe,
    or from a terminal), or because read() was interrupted by a signal. On
    error, -1 is returned, and errno is set appropriately. In this case it is
    left unspecified whether the file position (if any) changes.

ERRORS
    EAGAIN The file descriptor fd refers to a file other than a socket and has
    -UUU:%%--F1 *Man 2 read* {READ(2) page 1 of 1} Top (13,29) (Man)-----
```

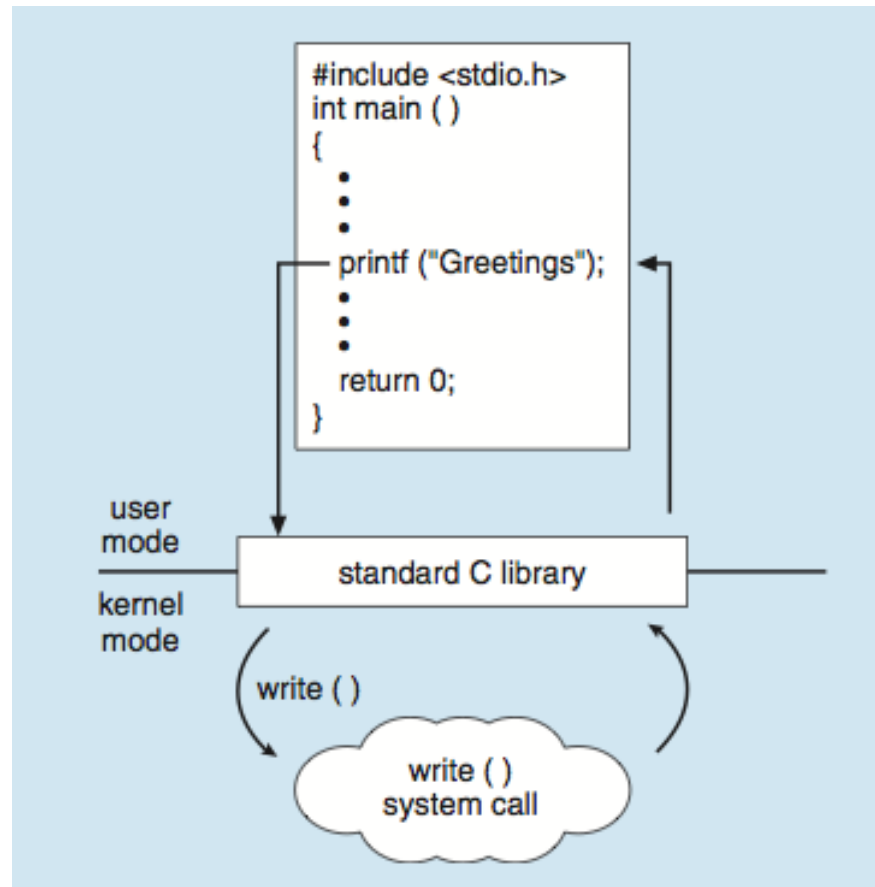
# API - System Call - OS





# Standard C Library Example

- C program invoking `printf()` library call, which calls `write()` system call



# Types of System Calls

- Process control
  - Create/terminate process, get/set process attributes, wait for time/event, allocate and free memory
- File management
  - create, delete, open, close, read, write, reposition
  - get and set file attributes
- Device management
  - request device, release device, read, write, reposition, get device attributes, set device attributes
- Communications
  - create, delete communication, send, receive messages
- Protection
  - Control access to resources, get/set permissions, allow and deny user access

# Examples of Windows and Unix System Calls

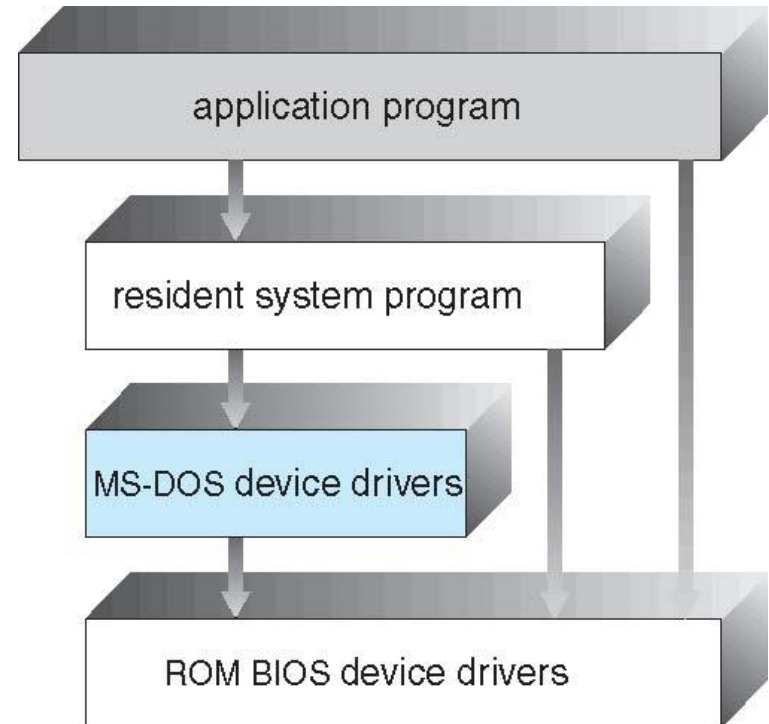
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

# Operating System Structure

- Simple structure – MS-DOS
- **Monolithic kernel – UNIX**
- Microkernel – Mach

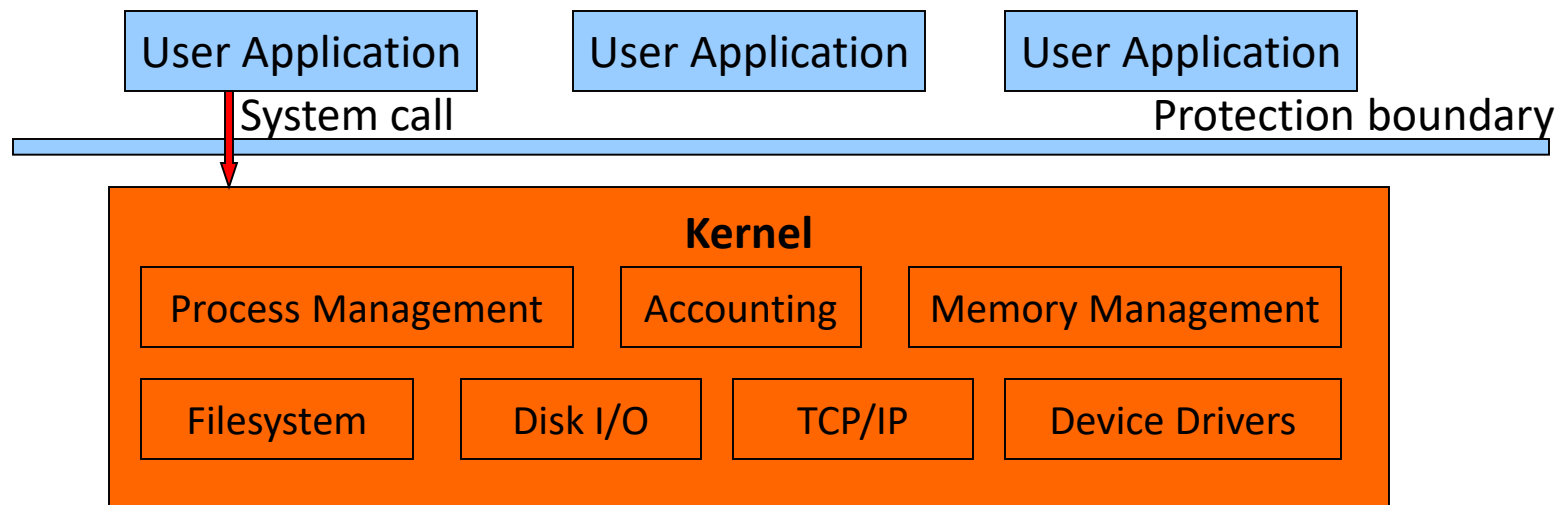
# MS-DOS Structure

- Written to provide the most functionality in the **least space**
- Minimal functionalities
- Not divided into modules
- Although MS-DOS has some structure, its interfaces and levels of *functionality are not well separated*



# UNIX: Monolithic Kernel

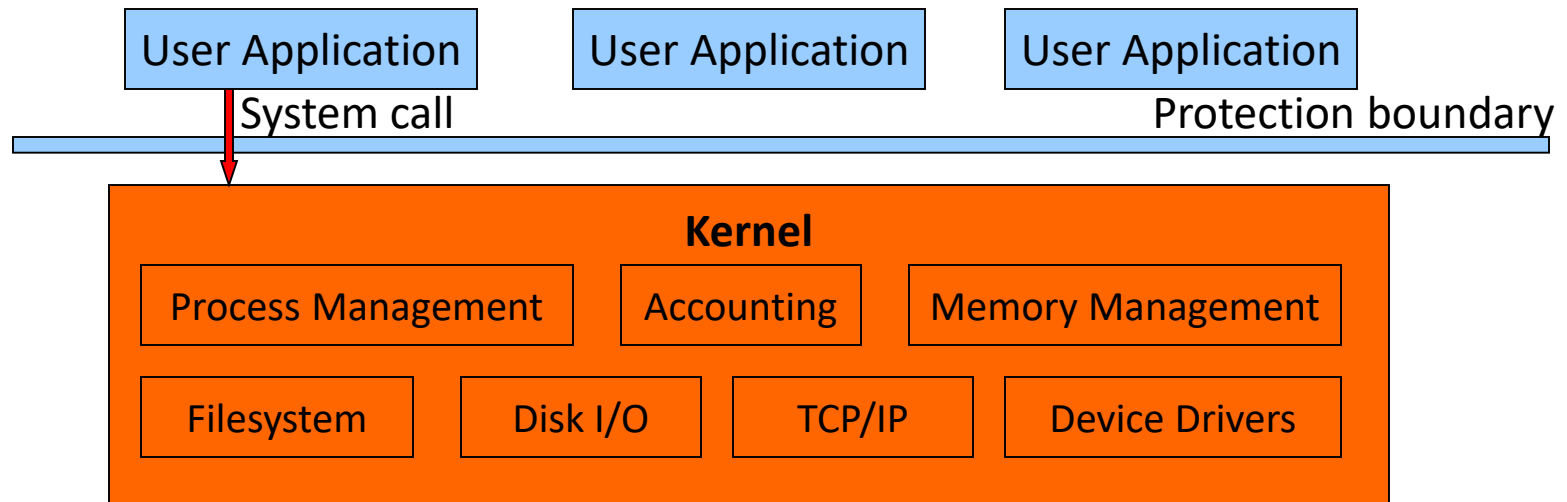
- Implements CPU scheduling, memory management, filesystems, and other OS modules all in a single big chunk



- Pros and Cons
  - + Overhead is low
  - + Data sharing among the modules is easy
  - Too big. (device drivers!!!)
  - A bug in one part of the kernel can crash the entire system

# Loadable Kernel Module

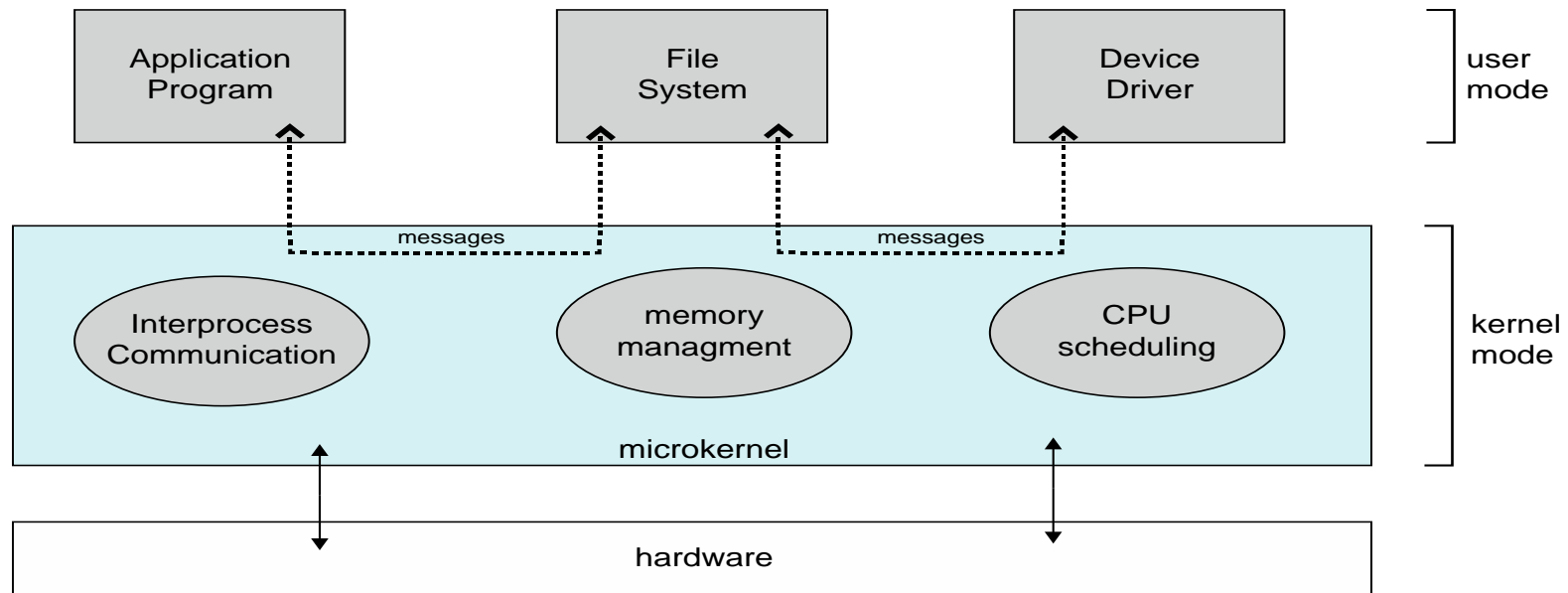
- Dynamically load/unload new kernel code
  - Linux and most today's OSes support this



- Pros and Cons
  - + Don't need to have every driver in the kernel.
  - + Easy to extend the kernel (just like micro kernel. See next)
  - A bug in a module can crash the entire system

# Microkernel

- Moves as much from the kernel into user space
- Communicate among kernels and user via **message passing**



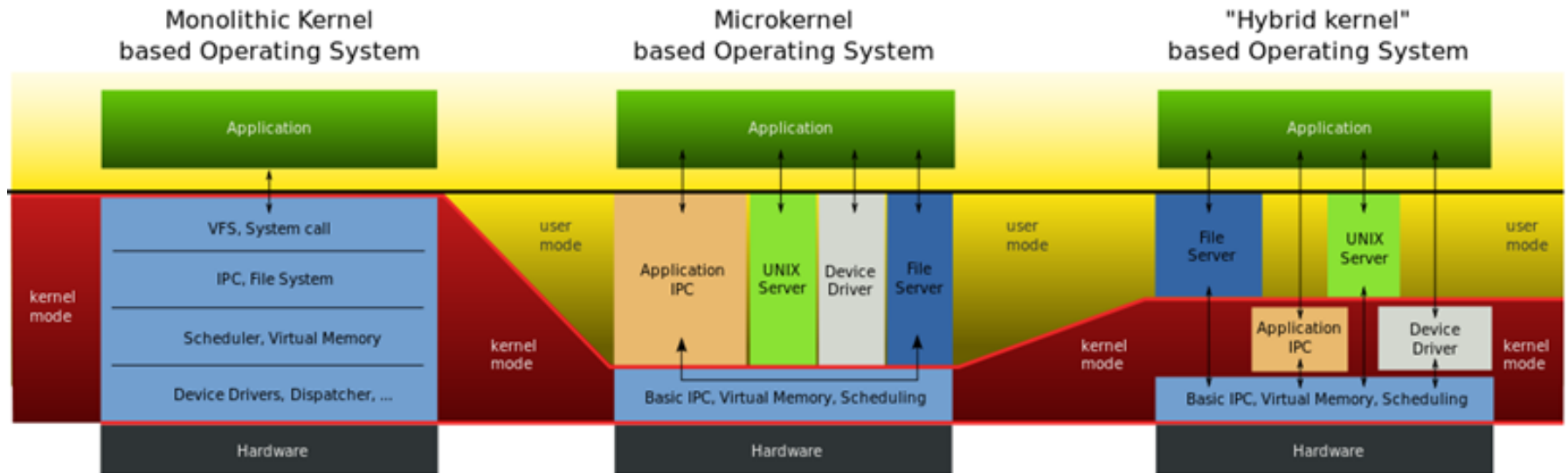
- Pros and Cons
  - + Easy to extend (user level driver)
  - + More reliable (**less code** is running in kernel mode)
  - **Performance overhead** of user space to kernel space communication



# Hybrid Structure

- Most modern operating systems are actually not one pure model
  - Hybrid combines multiple approaches to address performance, security, usability needs
  - Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality
  - Windows mostly monolithic, plus microkernel for different subsystem *personalities*
- Apple Mac OS X
  - hybrid, layered
  - Below is kernel consisting of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called kernel extensions)

# OS Structure Comparison



Source: [http://en.wikipedia.org/wiki/Monolithic\\_kernel](http://en.wikipedia.org/wiki/Monolithic_kernel)