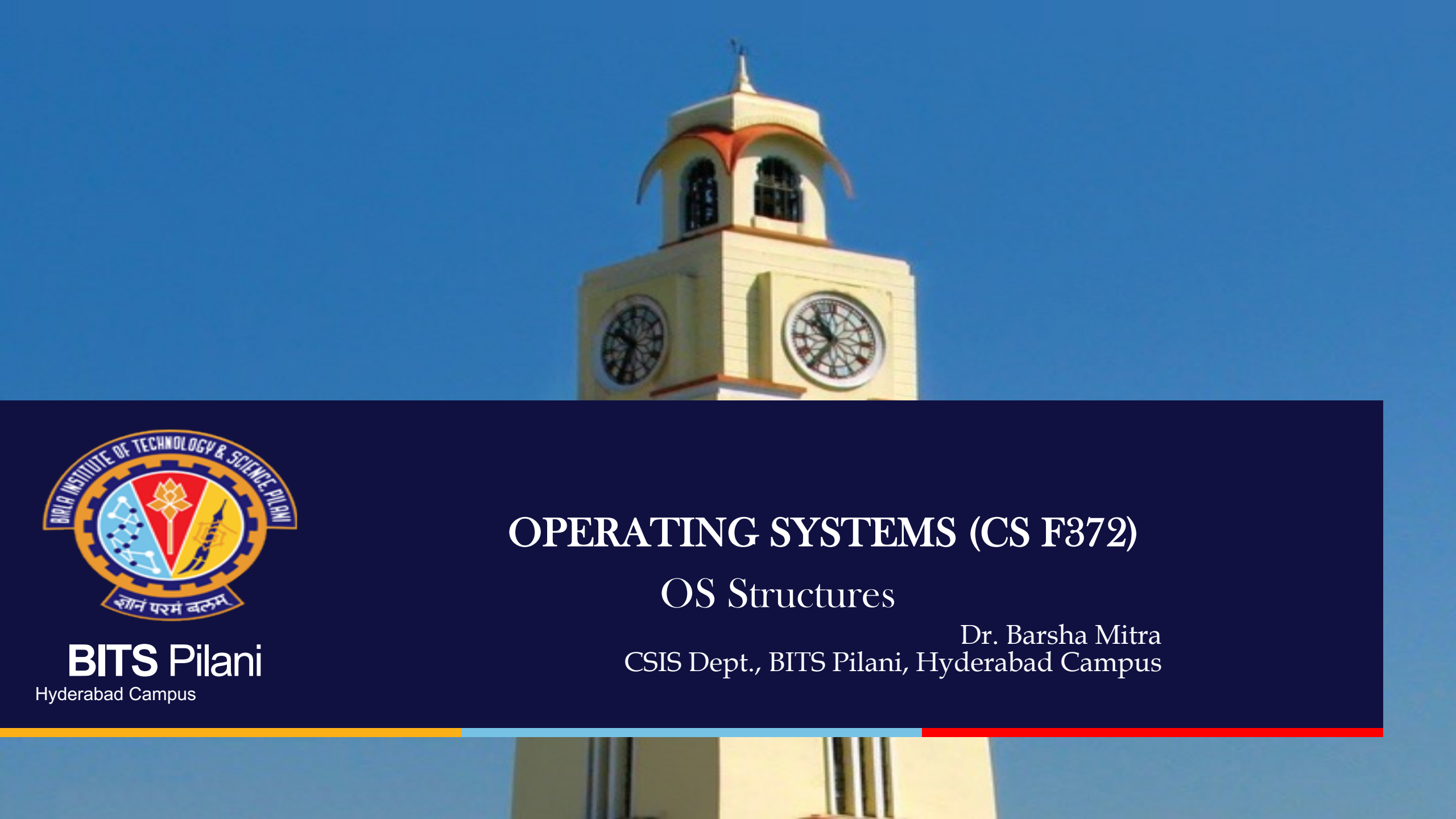


BITS Pilani
Hyderabad Campus

OPERATING SYSTEMS (CS F372)

OS Structures

Dr. Barsha Mitra
CSIS Dept., BITS Pilani, Hyderabad Campus

Operating System Services

- **User interface** - almost all operating systems have a user interface (**UI**)
  - **Command-Line (CLI), Graphics User Interface (GUI)**
- **Program execution** - system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error)
- **I/O operations** - running program may require I/O, which may involve a file or an I/O device
- **File-system manipulation** - read and write files and directories, create and delete them, search them, list file Information, permission management

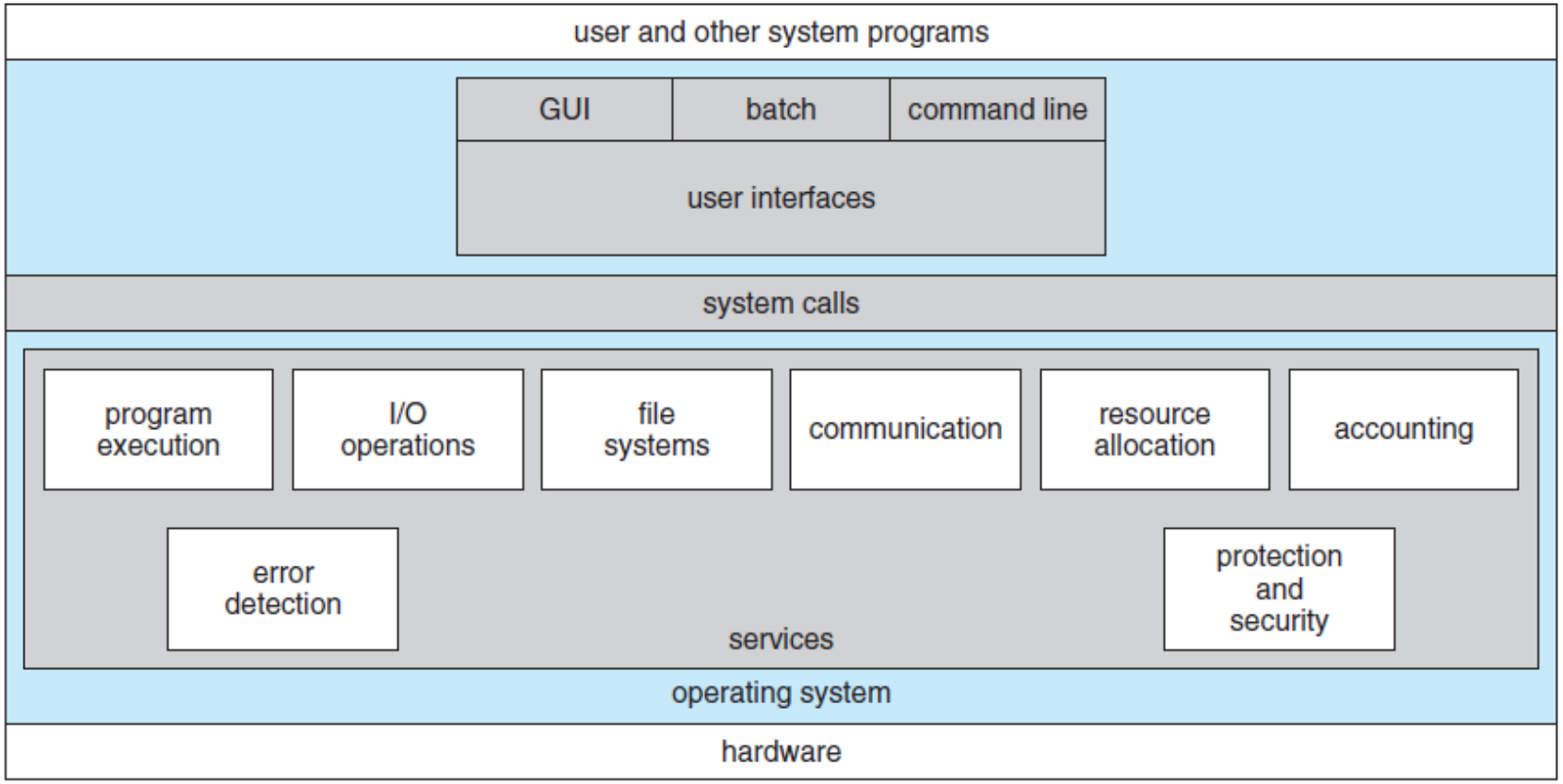
Operating System Services

- **Communications** – processes may exchange information, on the same computer or between computers over a network, **shared memory or message passing**
- **Error detection** –
  - OS needs to be constantly aware of possible errors
  - May occur in CPU and memory h/w, in I/O devices, in user program
  - OS should take the appropriate action to ensure correct and consistent computing
  - Take corrective actions

Operating System Services

- **Resource allocation** – allocating resources like CPU cycles, main memory, file storage, I/O devices for multiple concurrently executing processes
- **Accounting** – keep track of which users use how much and what kinds of computer resources
- **Protection and security** –
  - owners of information stored in a multiuser or networked computer system want to control use of that information
  - concurrent processes should not interfere with each other or with OS
  - ensuring that all accesses to system resources is controlled
  - security of the system from outsiders requires user authentication, extends to defending external I/O devices from invalid access attempts

Operating System Services



User and Operating-System Interface: CLI

- **CLI** or command interpreter
- Sometimes implemented in kernel, sometimes by separate program (Unix, Windows)
- Sometimes multiple flavors implemented – shells
- Primarily fetches a command from user and executes it

User and Operating-System Interface: GUI

- User-friendly interface
- Usually mouse, keyboard, and monitor
- Icons represent files, programs, actions, etc.
- Various mouse buttons over objects in the interface cause various actions (provide information, options, execute function, open directory (known as a folder))
- Many systems now include both CLI and GUI interfaces
  - Microsoft Windows is GUI with CLI “command” shell
  - Unix and Linux have CLI with optional GUI interfaces (CDE, KDE, GNOME)

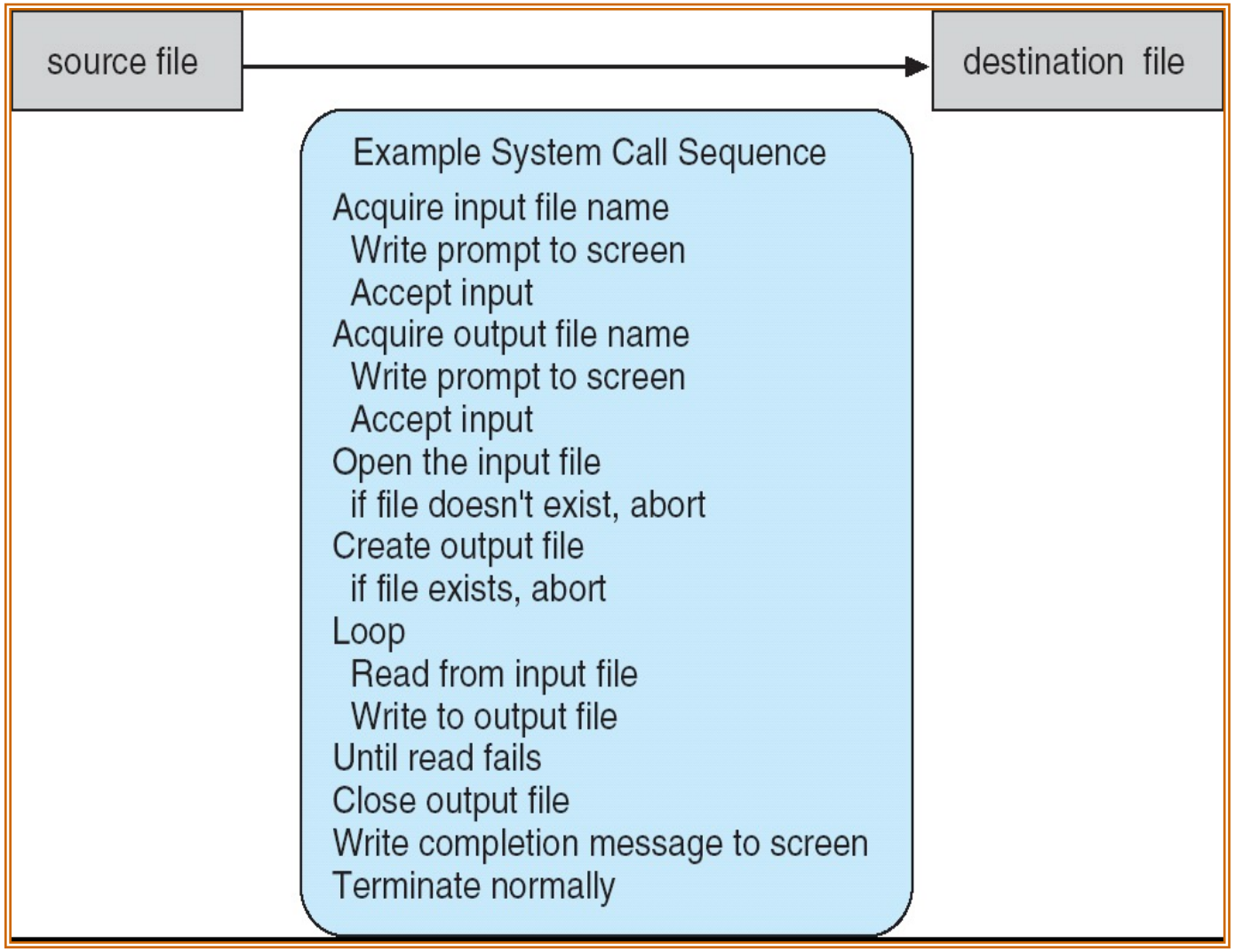
User and Operating-System Interface: Touchscreen Interface

- Touchscreen devices require new interfaces
- Mouse not possible or not desired
- Actions and selection based on gestures
- Virtual keyboard for text entry
- Voice commands

User and Operating-System Interface:

Choice of Interface

System Calls



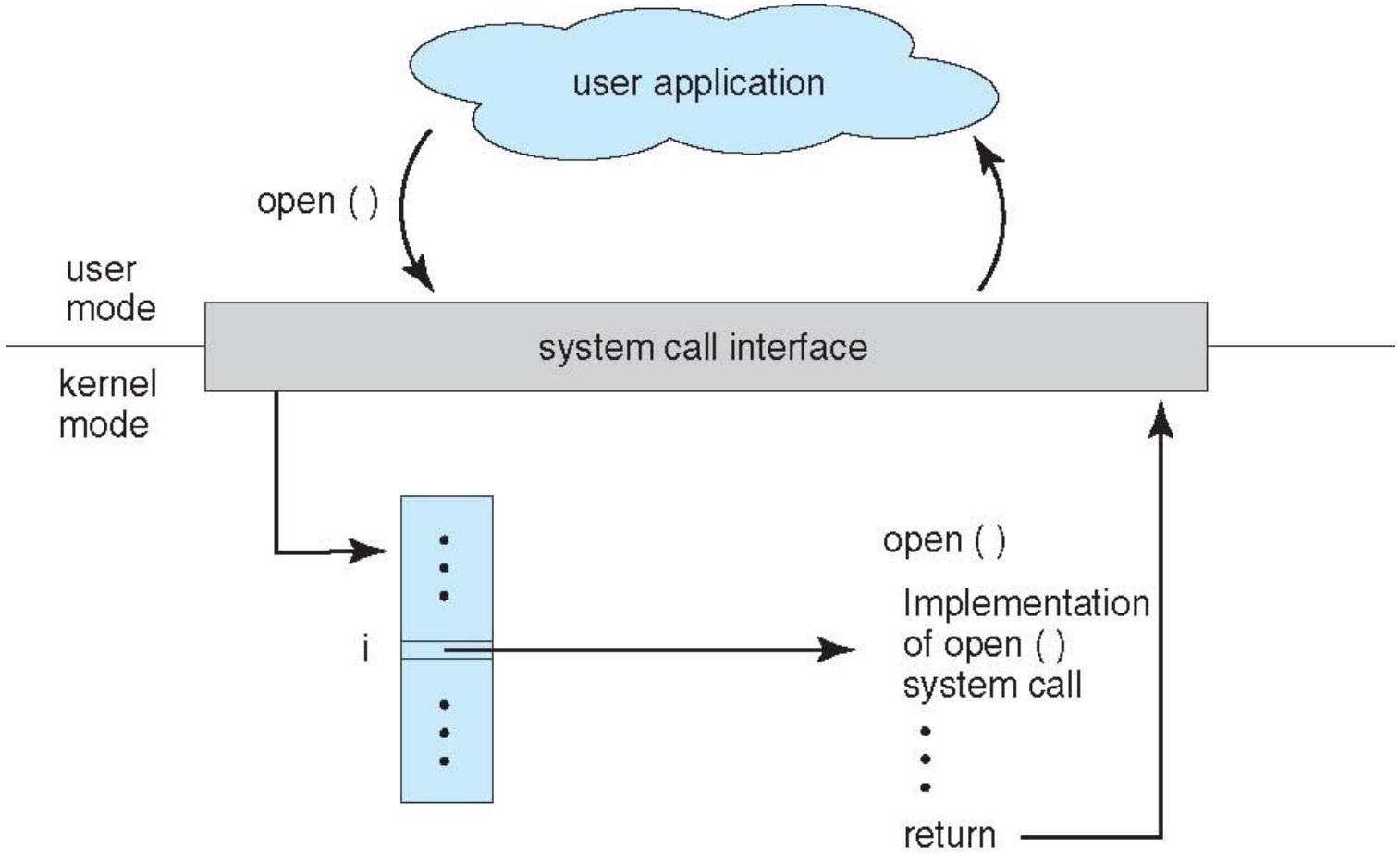
System Calls

- Interface to the services provided by the OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level Application Programming Interface (API) rather than direct system call use
- API specifies a set of functions available to application programmers
- Programmers access API via code library provided by the OS
- Three most common APIs are
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (including all versions of UNIX, Linux, and Mac OS X)
  - Java API for the Java virtual machine (JVM)

System Calls

- A number is associated with each system call
- System-call interface maintains a table indexed according to these numbers
- The system call interface invokes the intended system call in OS kernel and returns status of the system call and any return values
- The caller need know nothing about how the system call is implemented
- Just needs to obey API and understand what OS will do as a result of call execution
- Most details of OS interface hidden from programmer by API
- Managed by run-time support library (set of functions built into libraries included with compiler)

System Calls



Types of System Calls

- **Process control**
  - create process, terminate process
  - end, abort
  - load, execute
  - get process attributes, set process attributes
- **File management**
  - create file, delete file
  - open, close file
  - read, write file
  - get and set file attributes
- **Device management**
  - request device, release device
  - read, write
  - get device attributes, set device attributes
  - logically attach or detach devices
- **Information Maintenance**
- **Communication**
- **Protection**

Examples of System Calls

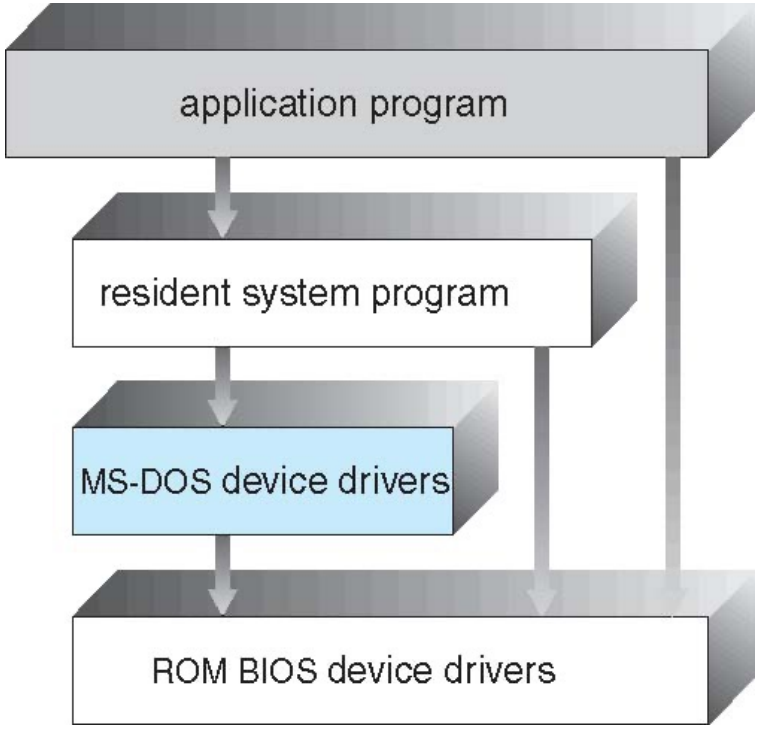
	Windows	Unix
Process Control	CreateProcess()	fork()
	ExitProcess()	exit()
	WaitForSingleObject()	wait()
File Manipulation	CreateFile()	open()
	ReadFile()	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetConsoleMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()
Information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

OS Structure

- Simple Structure/ Monolithic Kernel
- Layered Approach
- Microkernels
- Modules
- Hybrid System

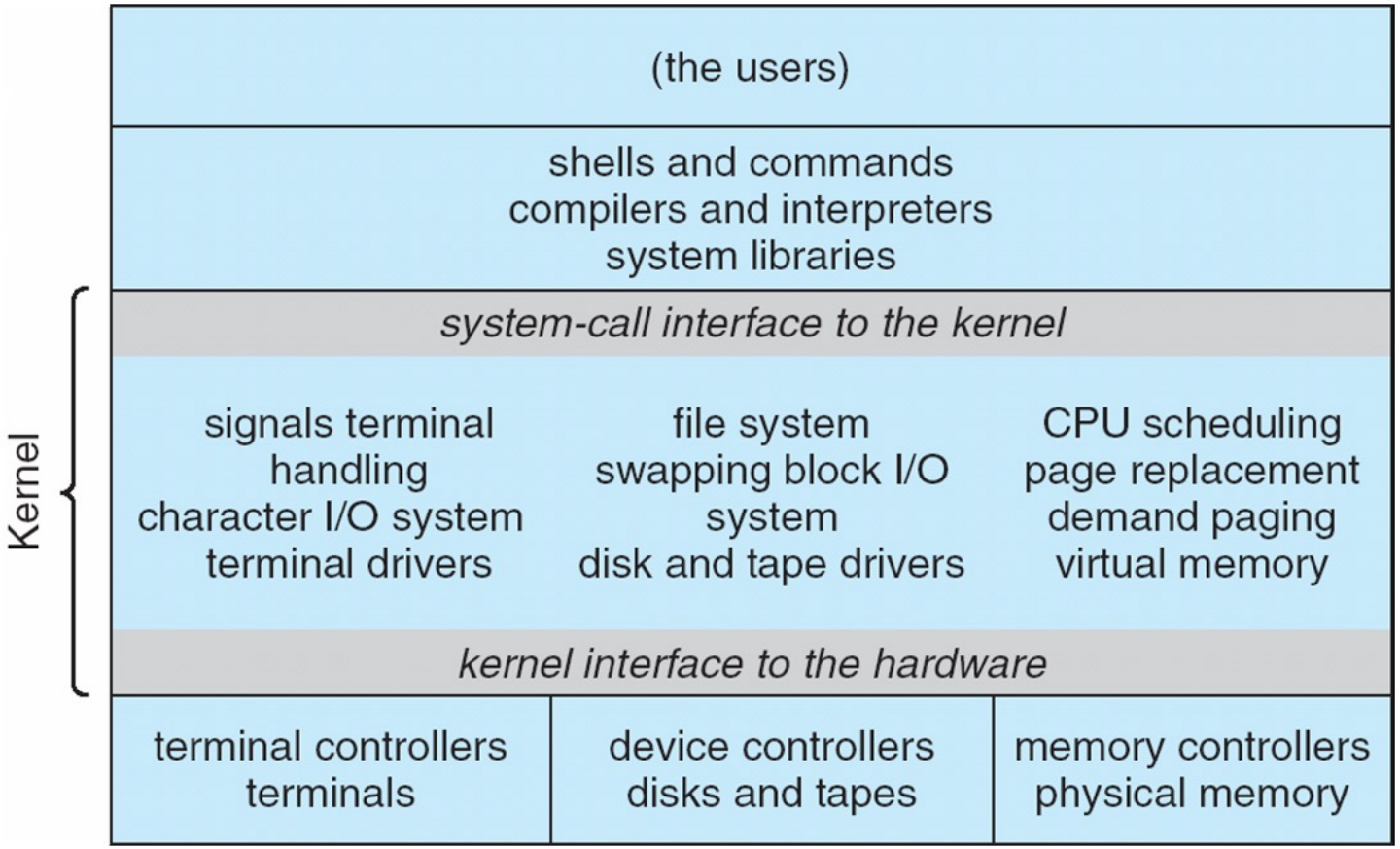
Simple Structure

- not divided into modules
- interfaces and levels of functionality are not well separated
- application programs are able to access the basic I/O routines to write directly to the display and disk drives
- vulnerable to malicious programs, causing entire system crashes when user programs fail



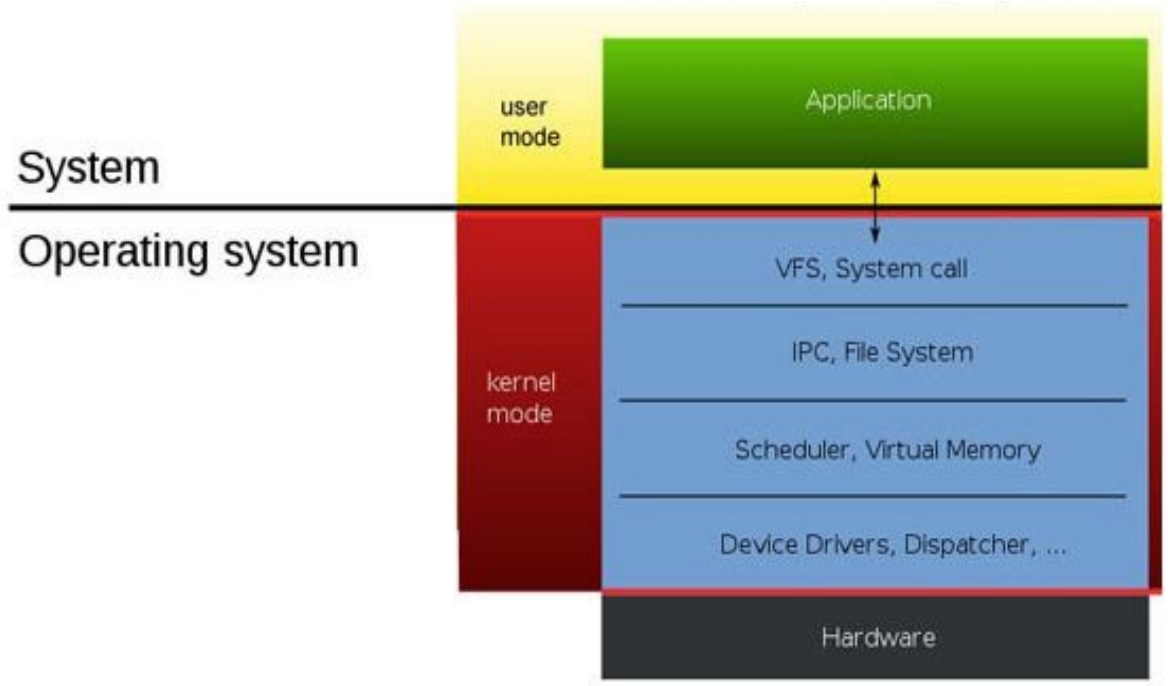
UNIX Architecture

- the original UNIX operating system had limited structuring
- consists of two separable parts
  - Systems programs
  - kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level



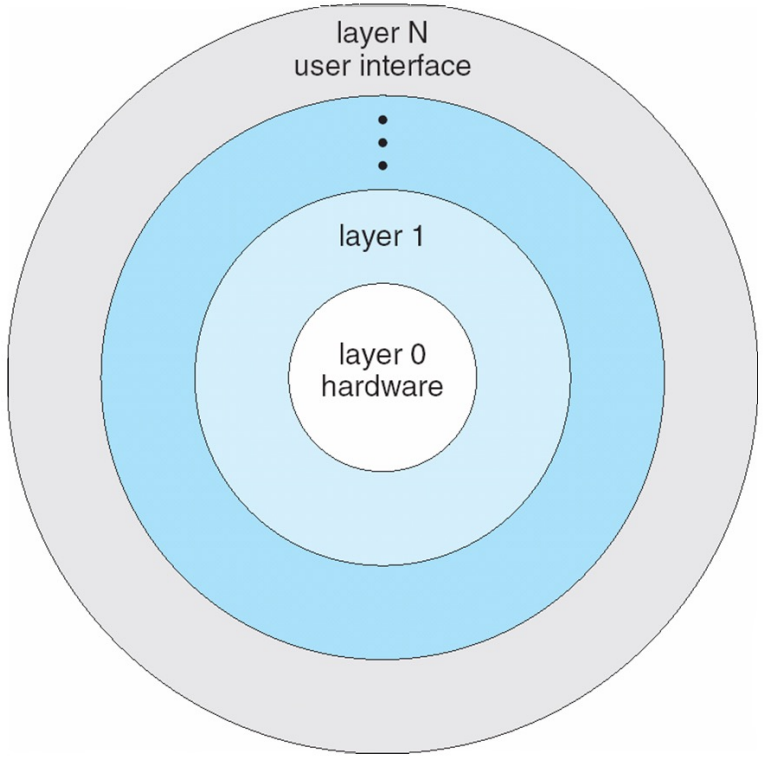
Monolithic Kernel

- entire operating system is working in kernel space
- larger in size
- little overhead in system call interface or in communication within kernel
- faster
- hard to extend
- if a service crashes, whole system is affected
- Eg., - Linux, Solaris, MS-DOS



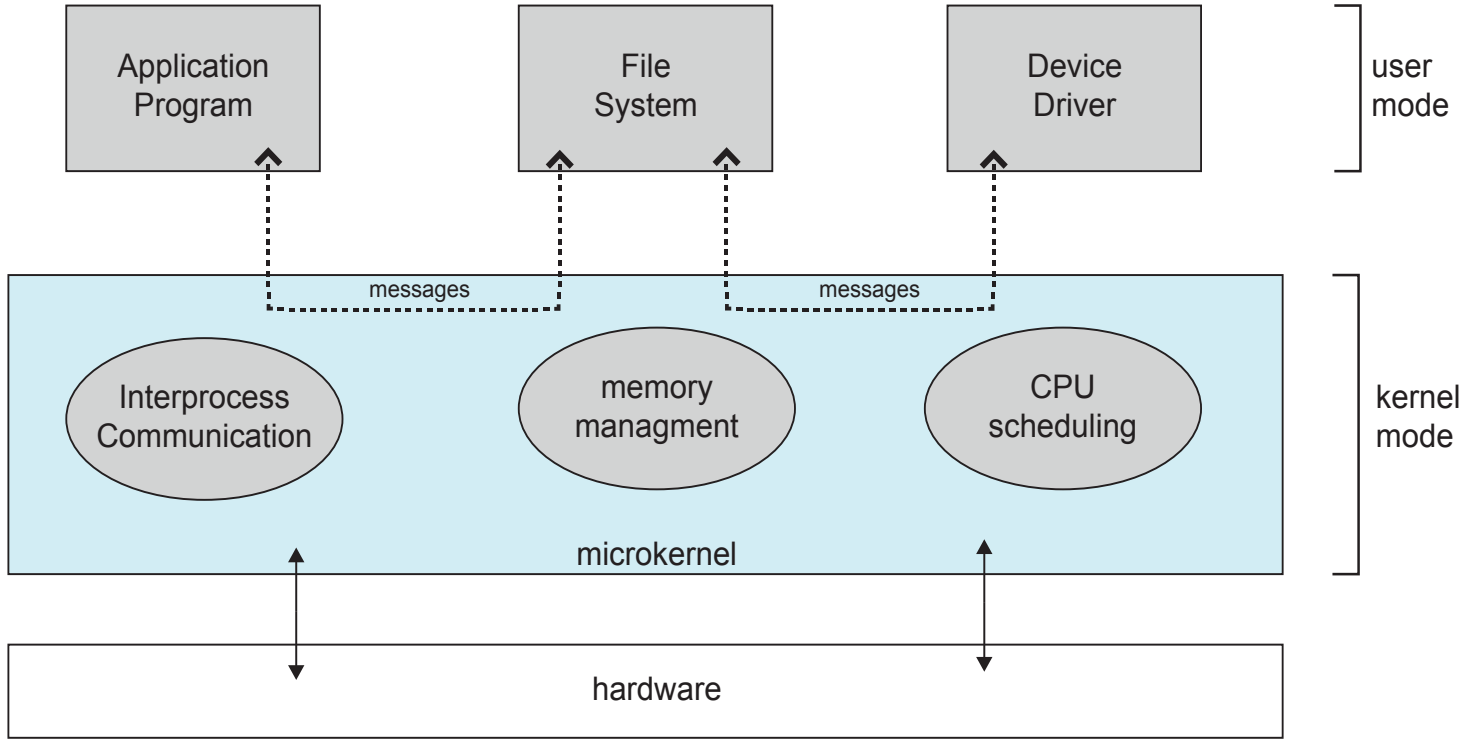
Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers
- The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers

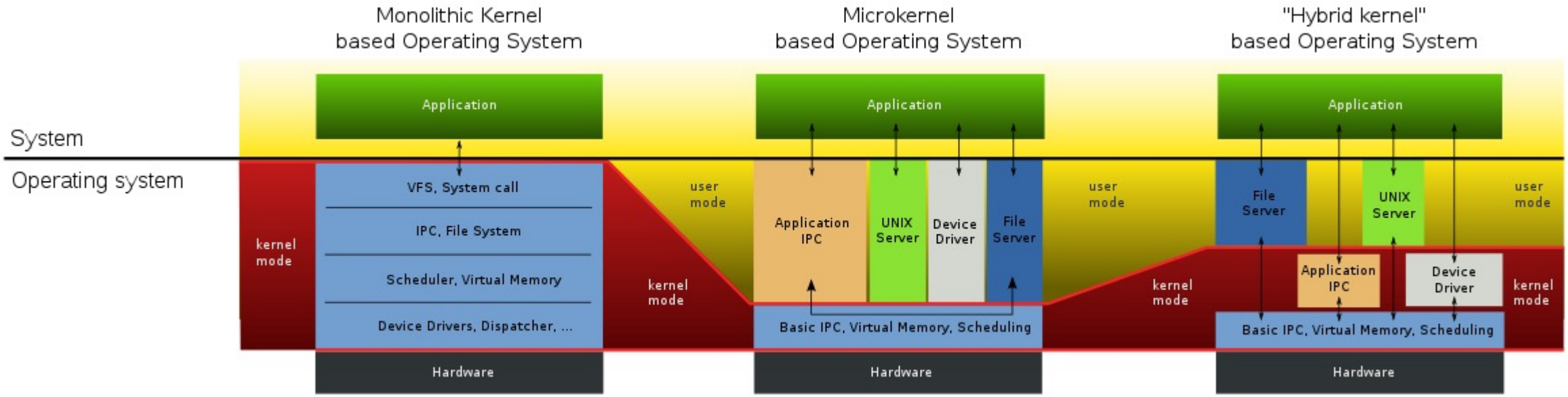


Microkernel

- user services and kernel services are in separate address spaces
- smaller in size
- slower
- extendible, all new services are added to user space
- if a service crashes, working of microkernel is not affected
- more secure and reliable
- eg., Mach, QNX, Windows NT (initial release)
- Drawback ??? *Performance overhead of user space to kernel space communication*



A Comparison

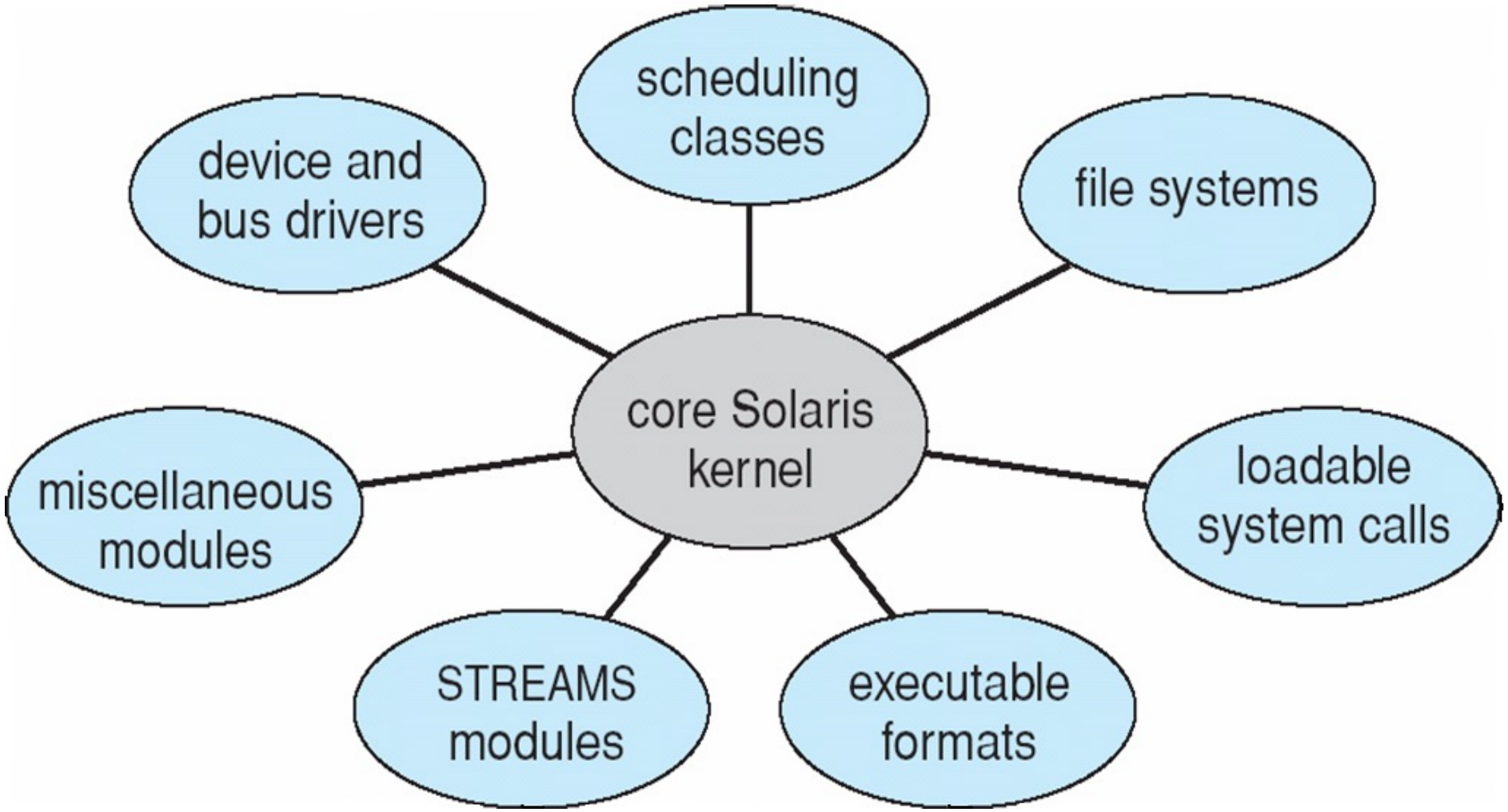


macOS X, Windows, iOS

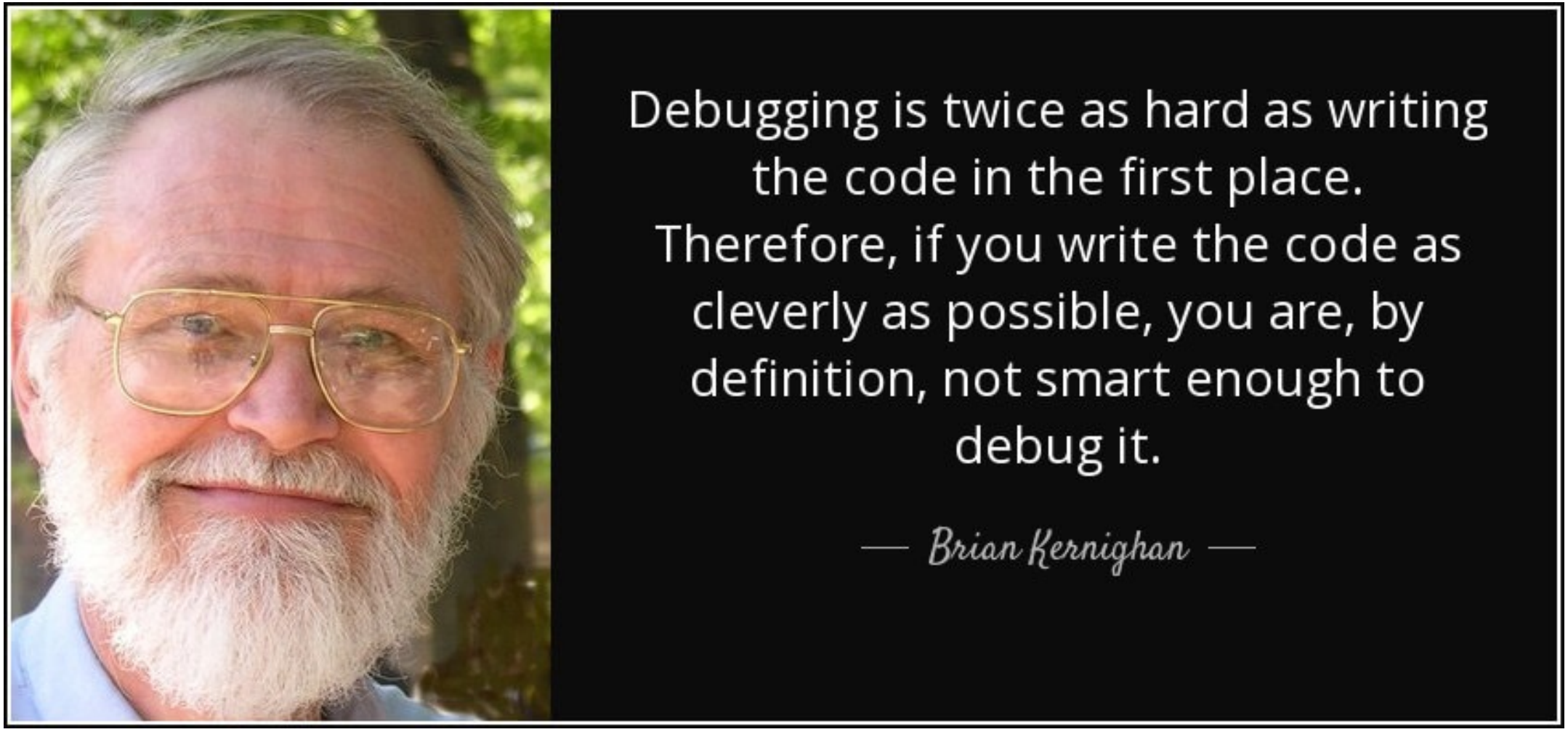
Modules

- loadable kernel modules
- kernel has a core set of components
- links in additional services via modules, either at boot time or during run time
- each module has a well defined interface
- dynamically linking services is preferable to adding new features directly to the kernel → does not require recompiling the kernel for every change
- better than a layered approach → any module can call any module
- better than microkernel → no message passing required to invoke modules

Modules


scheduling classes
device and bus drivers
file systems
core Solaris kernel
miscellaneous modules
loadable system calls
STREAMS modules
executable formats

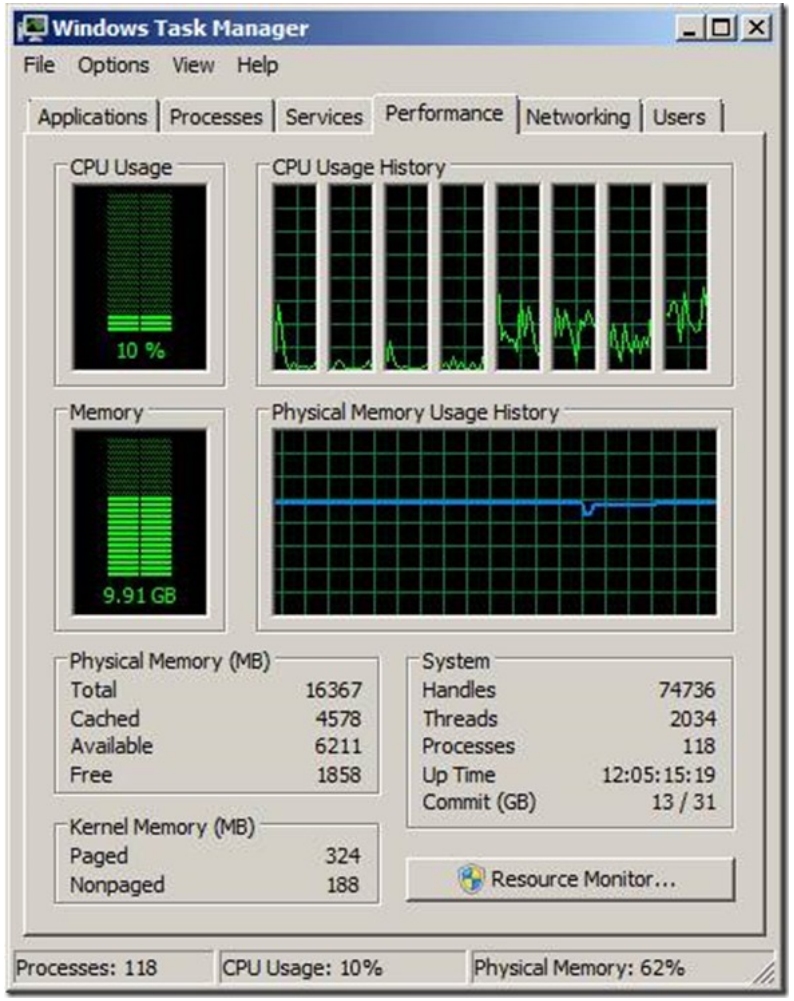
Operating-System Debugging



Performance Tuning


```
top - 04:33:30 up 17:16,  1 user,  load average: 0.05, 0.01, 0.03
Tasks:  75 total,   2 running,  73 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 88.6%id,  0.0%wa,  0.0%hi, 11.1%si,  0.0%st
Mem:    515348k total,   319956k used,   195392k free,    43432k buffers
Swap:  1048568k total,        0k used,  1048568k free,   201748k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    4 root      RT  -5     0    0    0 S  0.3  0.0   0:12.90 watchdog/0
32348 root      15   0  2200  996  796 R  0.3  0.2   0:00.04 top
    1 root      15   0  2068  612  528 S  0.0  0.1   0:34.24 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.04 ksoftirqd/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:04.77 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0   0:01.84 khelper
    7 root      11  -5     0    0    0 S  0.0  0.0   0:00.06 kthread
   10 root      10  -5     0    0    0 S  0.0  0.0   0:00.72 kblockd/0
   11 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
   47 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
   50 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
   52 root      10  -5     0    0    0 S  0.0  0.0   0:00.02 kseriod
```



System Boot

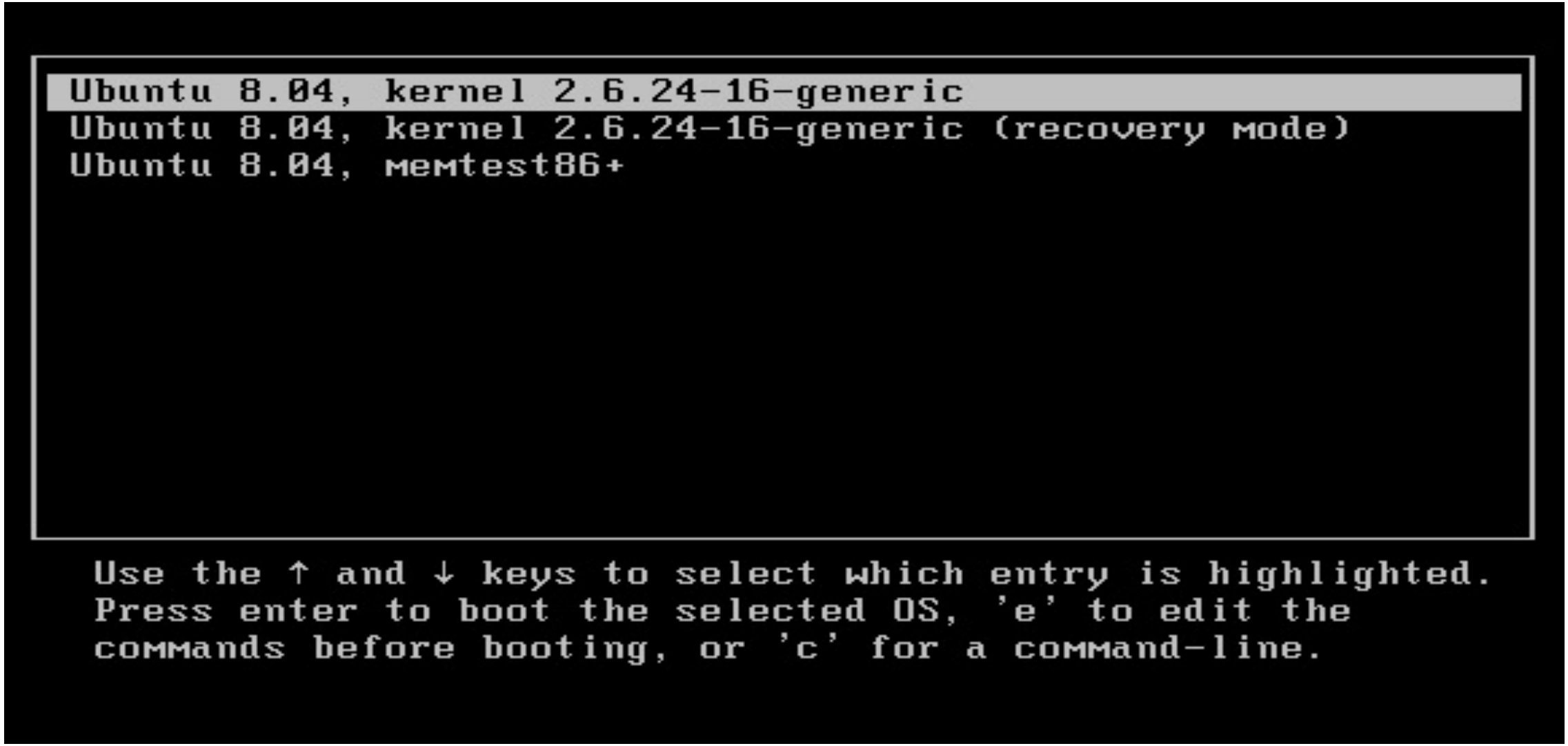
- bootstrap program / bootstrap
- simple bootstrap loader fetche
  boot program from disk, which
- program counter is loaded with
  memory location where the ini
  program is located
- diagnostics to determine the st
- *POST (Power-On Self-Test) is th*
  sequence that a computer's ba
  system (or "starting program")
  the computer keyboard, rando
  disk drives, and other hardwar
  correctly

```
Diskette Drive B  : None                Serial Port(s)    : 3F0 2F0
Pri. Master Disk  : LBA,ATA 100,  250GB Parallel Port(s)  : 370
Pri. Slave  Disk  : LBA,ATA 100,  250GB DDR at Bank(s)    : 0 1 2
Sec. Master Disk  : None
Sec. Slave  Disk  : None

Pri. Master Disk  HDD S.M.A.R.T. capability ... Disabled
Pri. Slave  Disk  HDD S.M.A.R.T. capability ... Disabled

PCI Devices Listing ...
Bus  Dev  Fun  Vendor Device  SVID  SSID Class  Device Class                IRQ
---------------------------------------------------------------------------
  0   27    0  8086   2668    1458  A005  0403  Multimedia Device           5
  0   29    0  8086   2658    1458  2658  0C03  USB 1.1 Host Cntrlr         9
  0   29    1  8086   2659    1458  2659  0C03  USB 1.1 Host Cntrlr        11
  0   29    2  8086   265A    1458  265A  0C03  USB 1.1 Host Cntrlr        11
  0   29    3  8086   265B    1458  265A  0C03  USB 1.1 Host Cntrlr         5
  0   29    7  8086   265C    1458  5006  0C03  USB 1.1 Host Cntrlr         9
  0   31    2  8086   2651    1458  2651  0101  IDE Cntrlr                 14
  0   31    3  8086   266A    1458  266A  0C05  SMBus Cntrlr               11
  1    0    0  10DE   0421    10DE  0479  0300  Display Cntrlr              5
  2    0    0  1283   8212    0000  0000  0180  Mass Storage Cntrlr        10
  2    5    0  11AB   4320    1458  E000  0200  Network Cntrlr             12
                                                ACPI Controller             9
```

System Boot


Ubuntu 8.04, kernel 2.6.24-16-generic
Ubuntu 8.04, kernel 2.6.24-16-generic (recovery mode)
Ubuntu 8.04, memtest86+
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

Thank You