ECE437/CS481

INTRODUCTION TO OS
OS STRUCTURE

Chapter 2.1-2.7

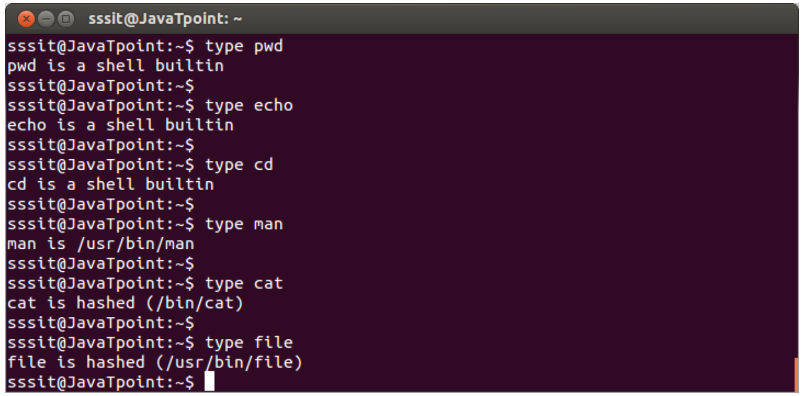
Xiang Sun

The University of New Mexico

User & OS

❑ User Interface

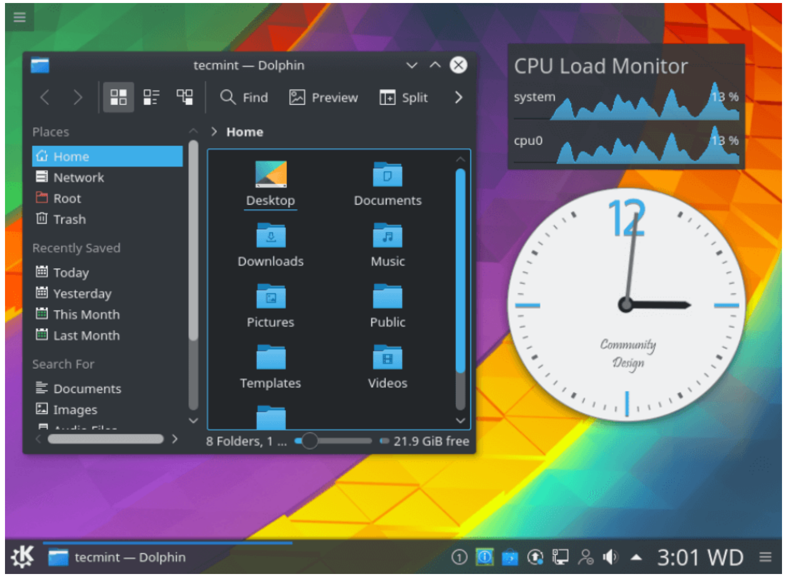
- Command Line Interface (CLI)
  - ✓ typical examples: Linux shells, Window command line
  - ✓ Efficient, flexible control, such as SHELL programming



sssit@JavaTpoint: ~

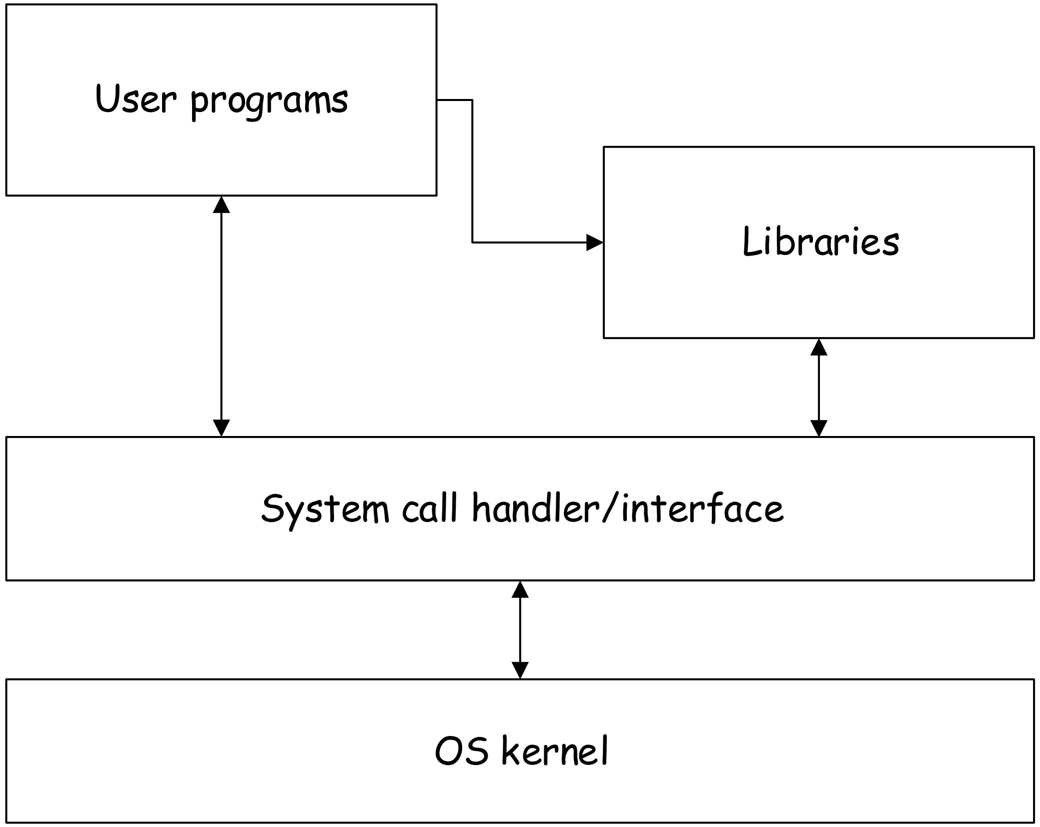
```
sssit@JavaTpoint:~$ type pwd
pwd is a shell builtin
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type echo
echo is a shell builtin
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type cd
cd is a shell builtin
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type man
man is /usr/bin/man
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type cat
cat is hashed (/bin/cat)
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ type file
file is hashed (/usr/bin/file)
sssit@JavaTpoint:~$ 
```

- GUI (Graphical user interface)
  - ✓ typical examples: Windows desktop, Linux K Desktop Environment (KDE)
  - ✓ easy to use, but introduce an extra layer of software between OS and users



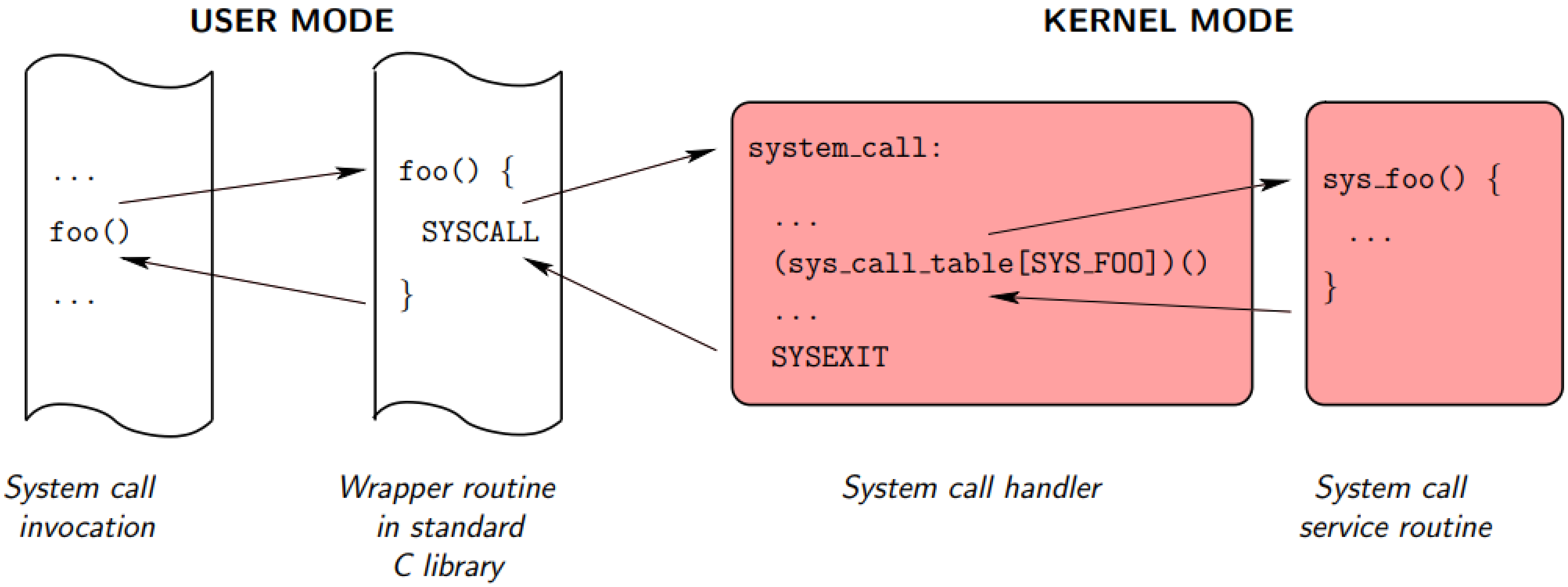
User & OS

- Application Programmer's Interface (API)
  - Language libraries: C, C++, Java, Fortran
  - System call handler/interface: entry points to the kernel



User & OS

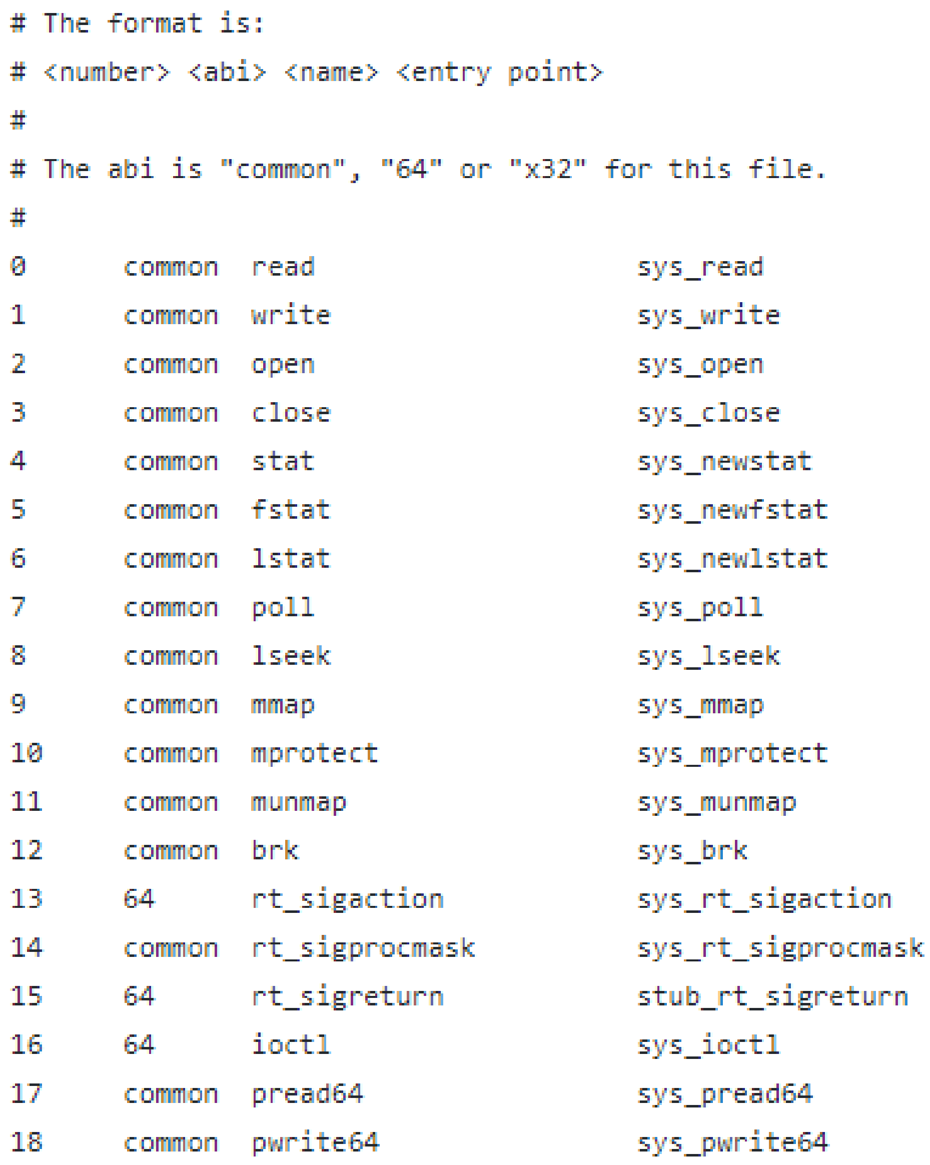
❑ Application Programmer's Interface (API)



User & OS

❑ System call

- ➢ All system calls defined in OS-specific header file Linux: /usr/include/sys/syscall.h
- ➢ The kernel keeps a list of all registered system calls in the system call table, stored in sys_call_table



```
# The format is:
# <number> <abi> <name> <entry point>
#
# The abi is "common", "64" or "x32" for this file.
#
0       common  read                    sys_read
1       common  write                   sys_write
2       common  open                    sys_open
3       common  close                   sys_close
4       common  stat                    sys_newstat
5       common  fstat                   sys_newfstat
6       common  lstat                   sys_newlstat
7       common  poll                    sys_poll
8       common  lseek                   sys_lseek
9       common  mmap                    sys_mmap
10      common  mprotect                sys_mprotect
11      common  munmap                  sys_munmap
12      common  brk                     sys_brk
13      64      rt_sigaction            sys_rt_sigaction
14      common  rt_sigprocmask          sys_rt_sigprocmask
15      64      rt_sigreturn            stub_rt_sigreturn
16      64      ioctl                   sys_ioctl
17      common  pread64                 sys_pread64
18      common  pwrite64                sys_pwrite64
```

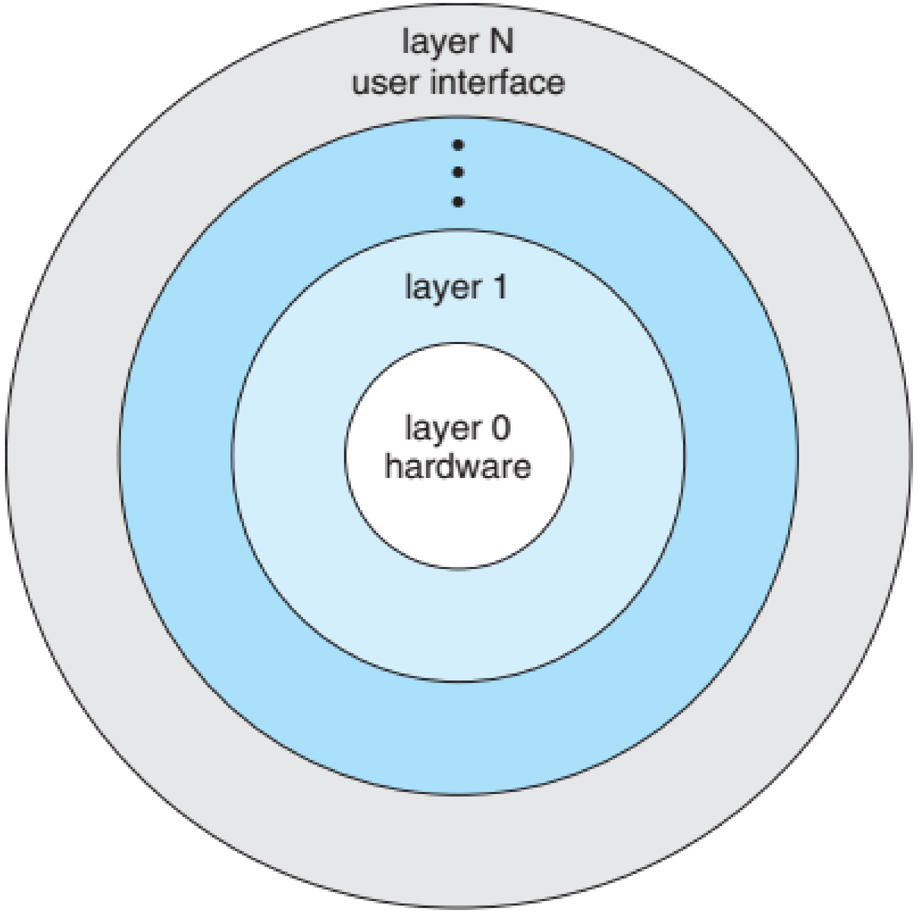
User & OS

- ❑ Types of system call
  - ➢ Process control
  - ➢ File management
  - ➢ Device management
  - ➢ Information maintenance
  - ➢ Communications
  - ➢ Protection

OS Structure

❑ Organization of Operating Systems—Layered Approach

- OS is divided into layers.
- Each built on top of lower layers.
  - ✓ The bottom layer (layer 0), is the hardware.
  - ✓ The highest (layer N) is the user interface.
- Each layer uses functions and services from only lower-level layers
- Kernel is implemented as a single approach

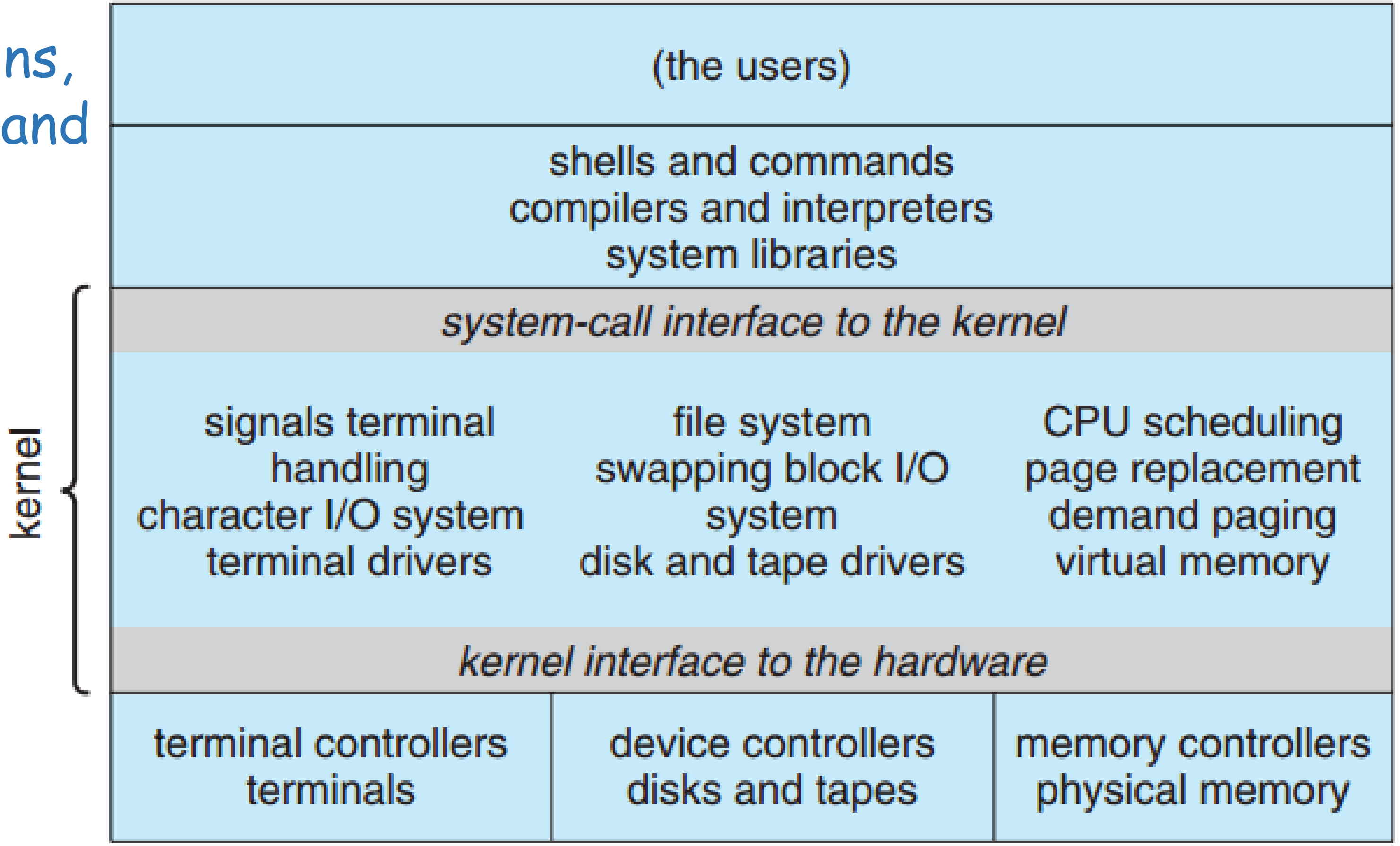


© by Dr. X. Sun

OS Structure

❑ Organization of Operating Systems—Layered Approach

- Example: Traditional UNIX system
- Kernel has too many functions, thus difficult to implement and maintain.



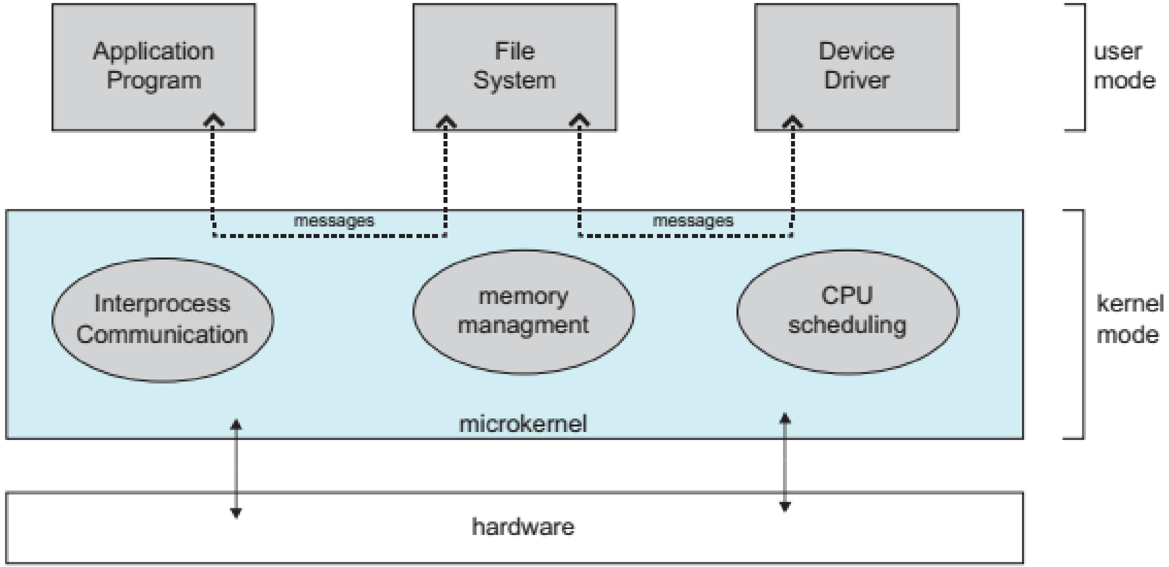
OS Structure

❑ Organization of Operating Systems—Microkernel approach

- Remove nonessential services from the kernel and implement them as system an user-level programs.----to make kernel much smaller and faster.

- How to determine a service is essential?
  ---Little consensus.

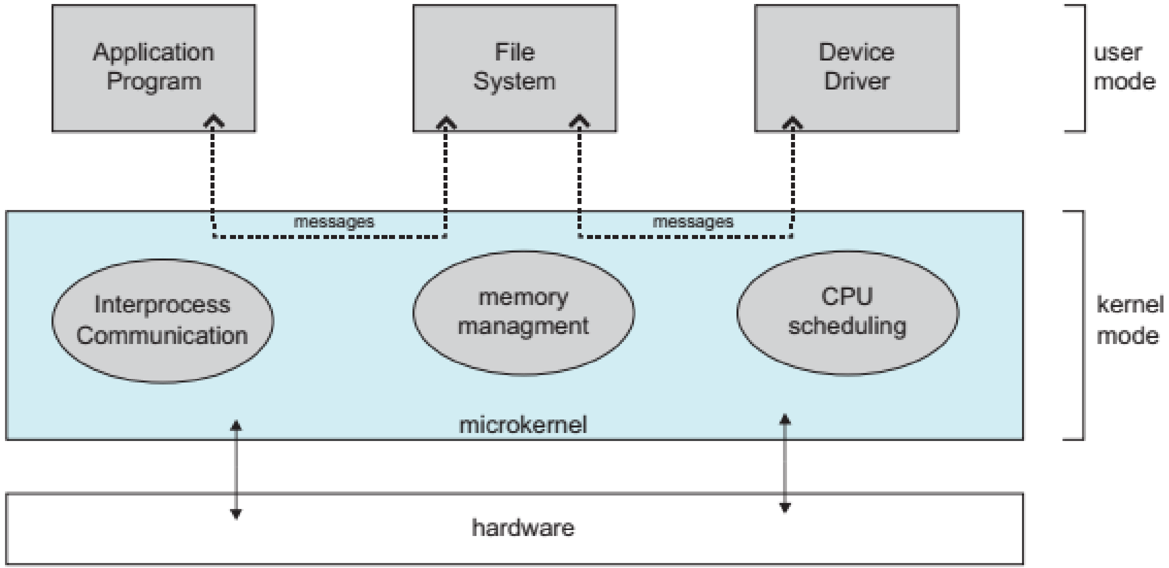
- Functions of Microkernel:
  - ✓ provide essential services
  - ✓ provide communications among services and user programs based on message passing



OS Structure

❑ Organization of Operating Systems—Microkernel approach

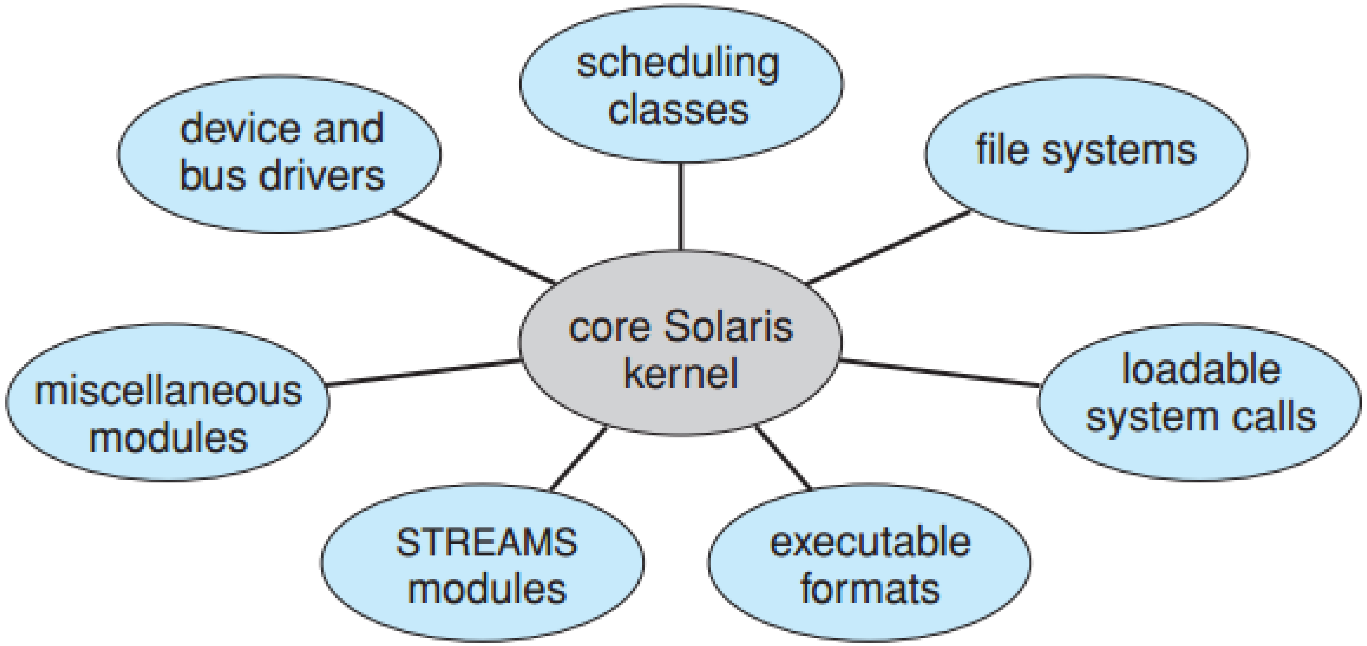
- ➢ Pros:
  - ✓ More reliable and more secure since less code is running in kernel mode.
  - ✓ Flexible for dynamical module configuration
  - ✓ Easier to extend a microkernel
  - ✓ Easier to port OS to new architectures
- ➢ Cons
  - ✓ Overhead of user space to kernel space communication



OS Structure

❑ Organization of Operating Systems—Module Approach

- Each portion of kernel is implemented as a loadable module.
- The kernel provides core services of loading and communicating with different modules.
- Example: Solaris.
- It is more flexible than the Layered approach and more efficient than the Microkernel approach.

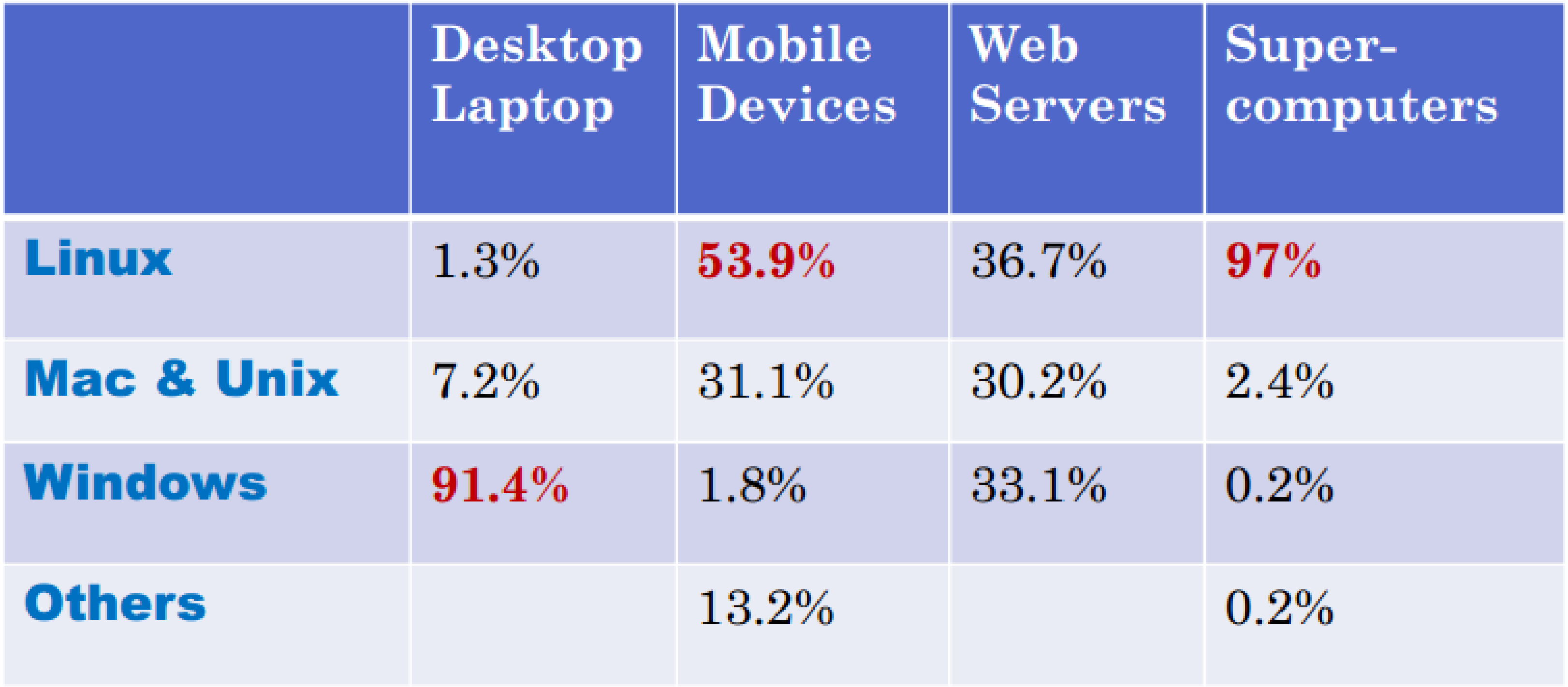


OS Overall

- Resource abstraction and sharing
  - Abstraction: hide the resource details
    - Example: disk's sector size, # of sector per track
  - Sharing: efficiently manage the resources
    - Example: time-sharing---CPU, memory
    - Example: space-sharing---memory, disk

OS Overall

- ❑ Usage share of operating systems
  - ➢ the percentage market share of the operating systems used in various computers, from Wikipedia as August 2015

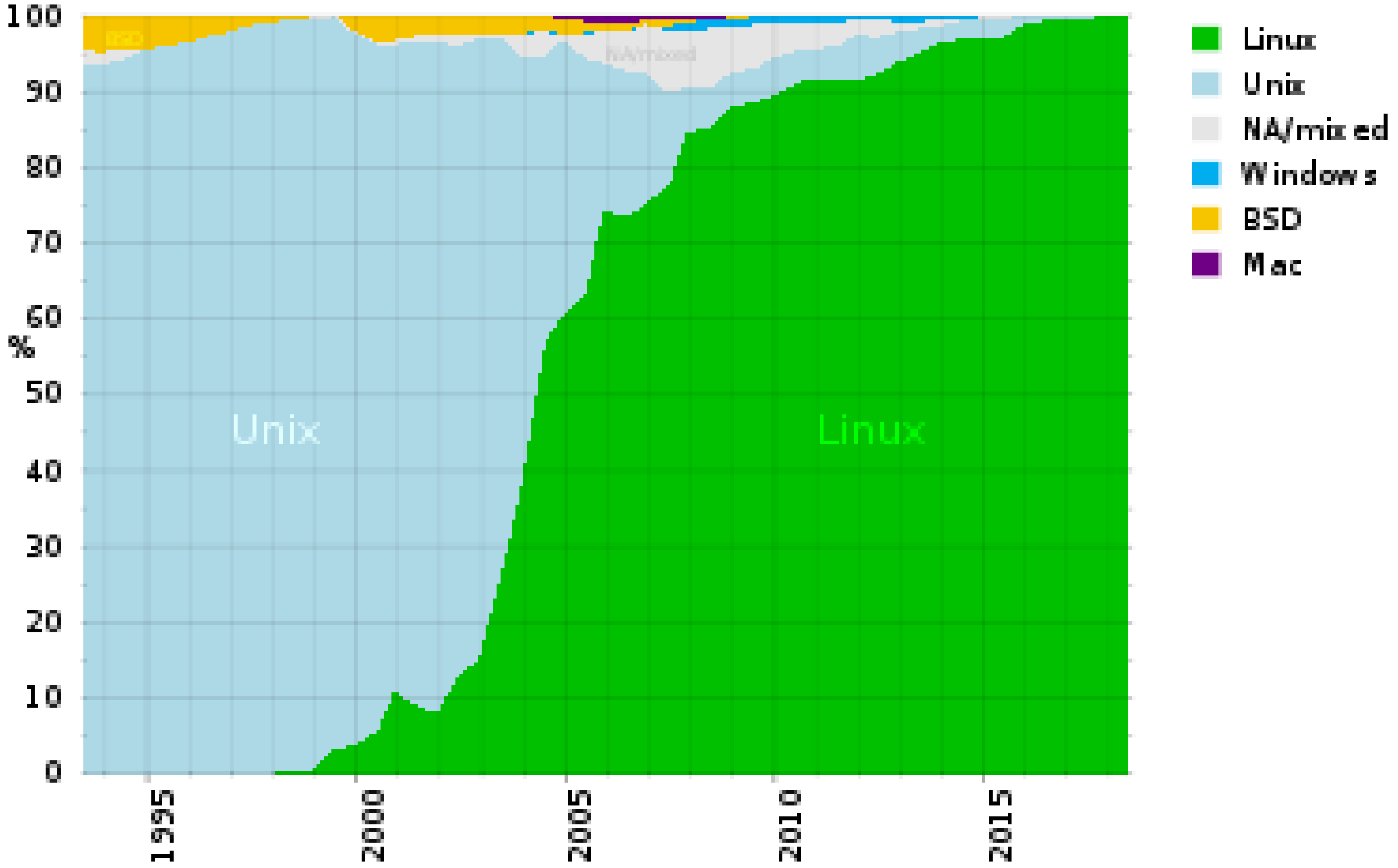


	Desktop Laptop	Mobile Devices	Web Servers	Super-computers
Linux	1.3%	53.9%	36.7%	97%
Mac & Unix	7.2%	31.1%	30.2%	2.4%
Windows	91.4%	1.8%	33.1%	0.2%
Others		13.2%		0.2%

OS Overall

- **Usage share** of operating systems

  - the percentage market share of the operating systems used in TOP500 Supercomputers, from Wikipedia as 2013


BSD
NA/mixed
Unix
Linux
Linux
Unix
NA/mixed
Windows
BSD
Mac