ECE437/CS481

INTRODUCTION TO OS
OS STRUCTURE

Chapter 2.1-2.7

Xiang Sun

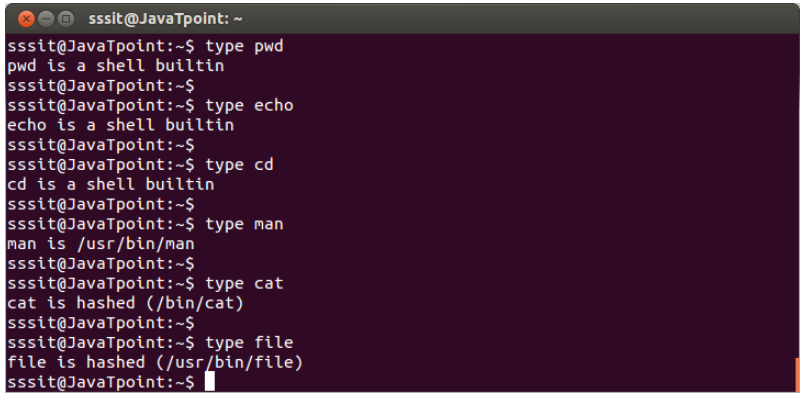
The University of New Mexico

User & OS

❑ User Interface

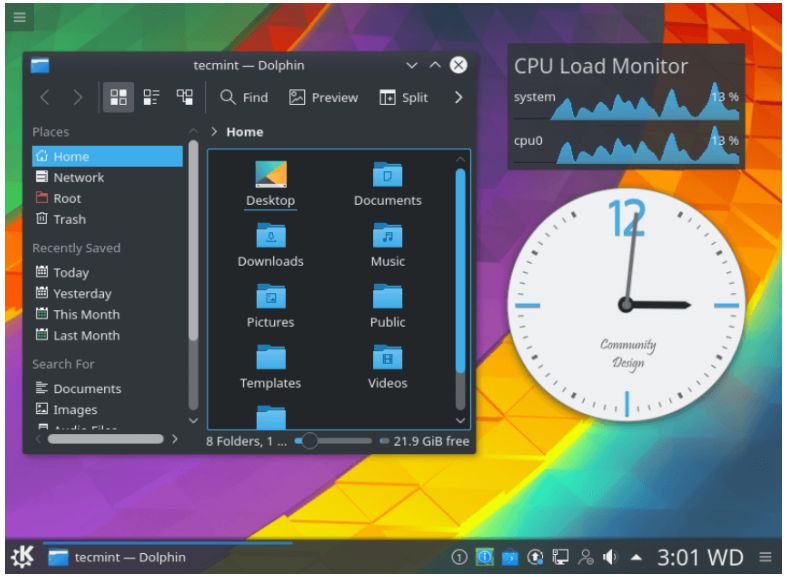
➢ Command Line Interface (CLI)

- ✓ typical examples: Linux shells, Window command line
- ✓ Efficient, flexible control, such as SHELL programming



➢ 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

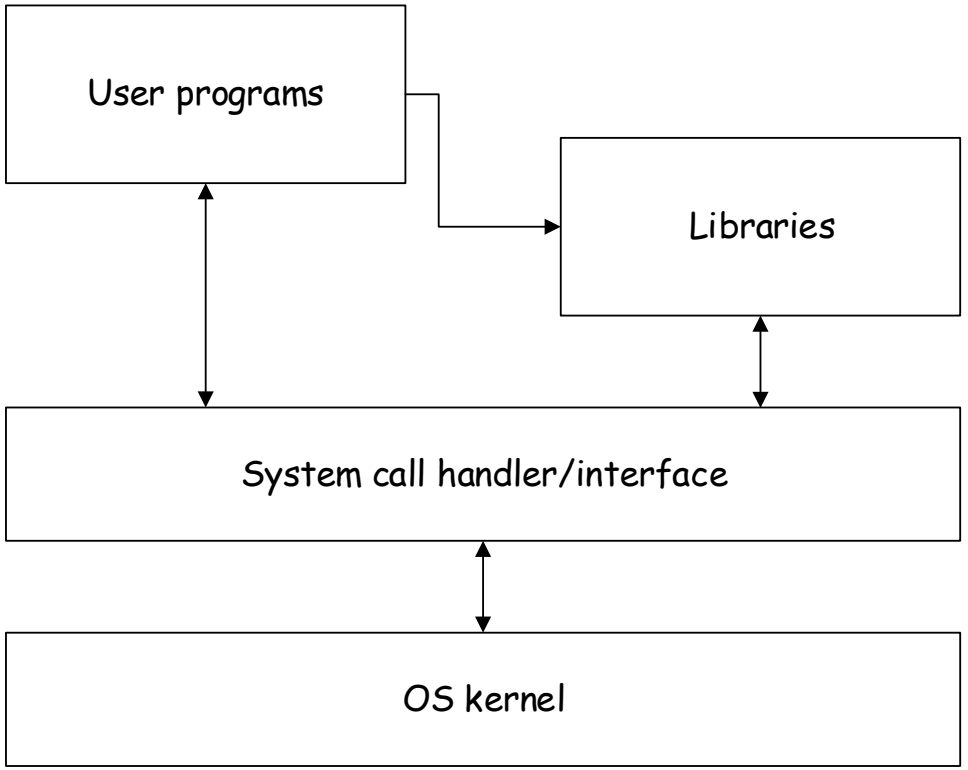


© by Dr. X. Sun

User & OS

❑ Application Programmer's Interface (API)

- ➢ Language libraries: C, C++, Java, Fortran
- ➢ System call handler/interface: entry points to the kernel



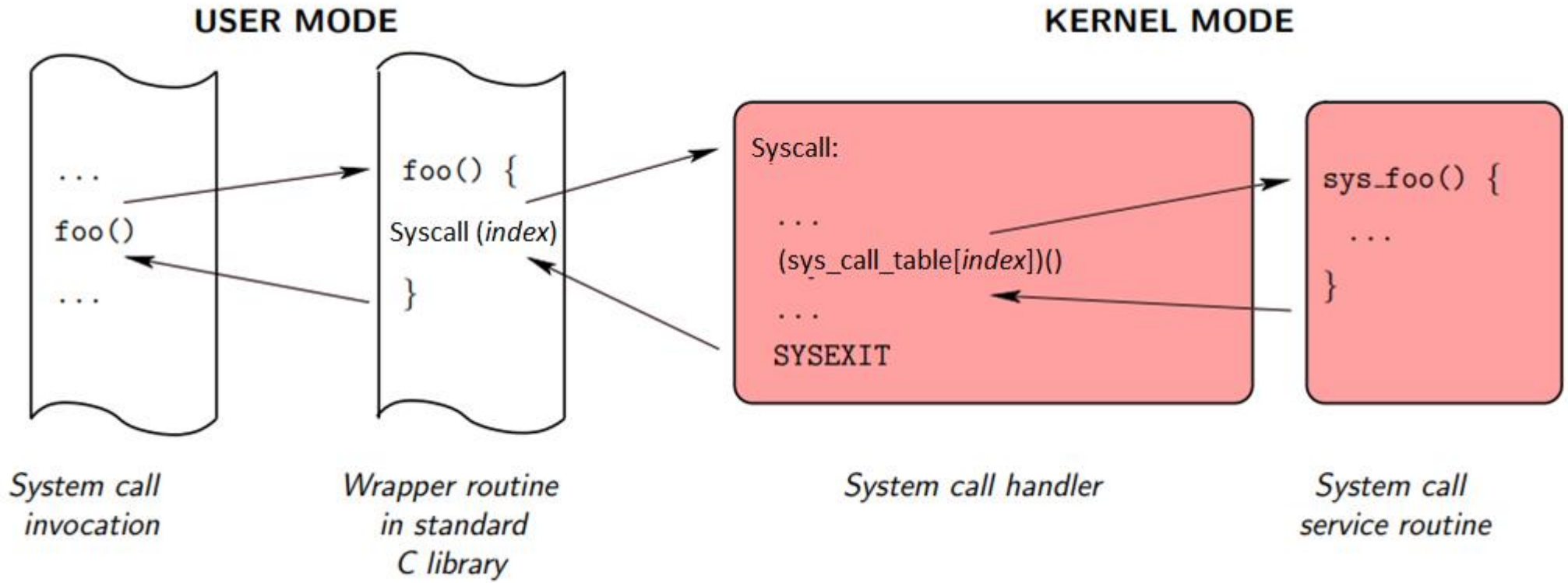

```
fopen(const char *file, const char *mode)
{
        FILE *fp;
        int f;
        int flags, oflags;

        if ((flags = __sflags(mode, &oflags)) == 0)
                return (NULL);
        if ((fp = __sfp()) == NULL)
                return (NULL);
        if ((f = open(file, oflags, DEFFILEMODE)) < 0) {
                fp->_flags = 0;                 /* release */
                return (NULL);
  }
        fp->_file = f;
        fp->_flags = flags;
        fp->_cookie = fp;
        fp->_read = __sread;
        fp->_write = __swrite;
        fp->_seek = __sseek;
        fp->_close = __sclose;

        /*
         * When opening in append mode, even though we use O_APPEND,
         * we need to seek to the end so that ftell() gets the right
         * answer.  If the user then alters the seek pointer, or
         * the file extends, this will fail, but there is not much
         * we can do about this.  (We could set __SAPP and check in
         * fseek and ftell.)
         */
        if (oflags & O_APPEND)
                (void) __sseek((void *)fp, (fpos_t)0, SEEK_END);
        return (fp);
}
```

© by Dr. X. Sun

❑ Application Programmer's Interface (API)



© by Dr. X. Sun

System call

- The kernel keeps a list of all registered system calls in the system call table, stored in sys_call_table

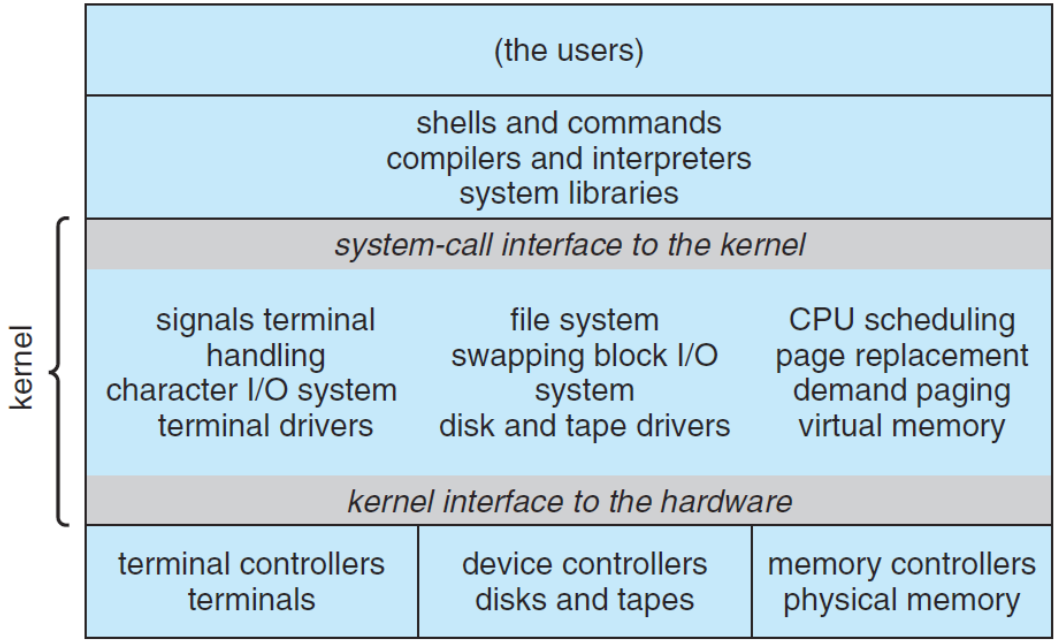
```
# The format is:
# <number> <abi> <name> <entry point>
#
# The abi is "common", "64" or "x32" for this file.
#
0      common  read                    sys_read
1      common  write                   sys_write
2      common  open                    sys_open
3      common  close                   sys_close
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
```

© by Dr. X. Sun

OS Structure

❑ Organization of Operating Systems—Monolithic Approach

- No structure—placing all the functions of the kernel into a single, static binary file that runs in a single address space.
- For example, in the Linux structure, user programs use system calls to invoke functions of the kernel.
- Quick communications between user processes/programs and the kernel, but difficult to extend/modify the kernel.

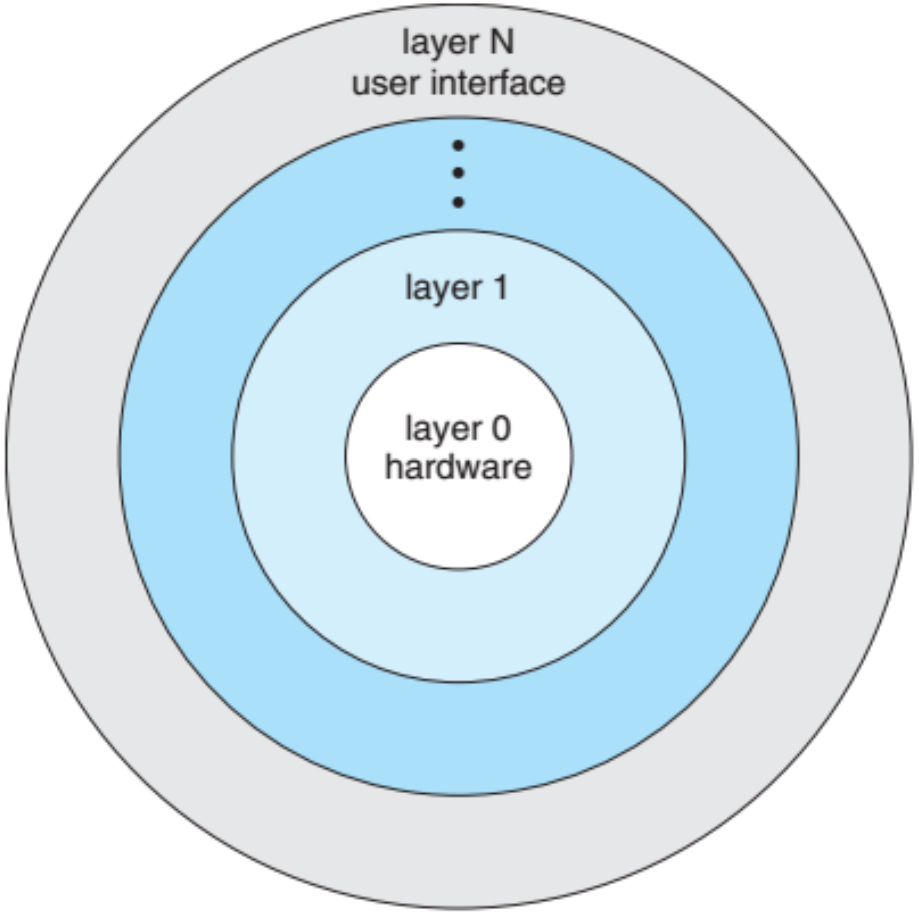


© by Dr. X. Sun

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 layer/approach

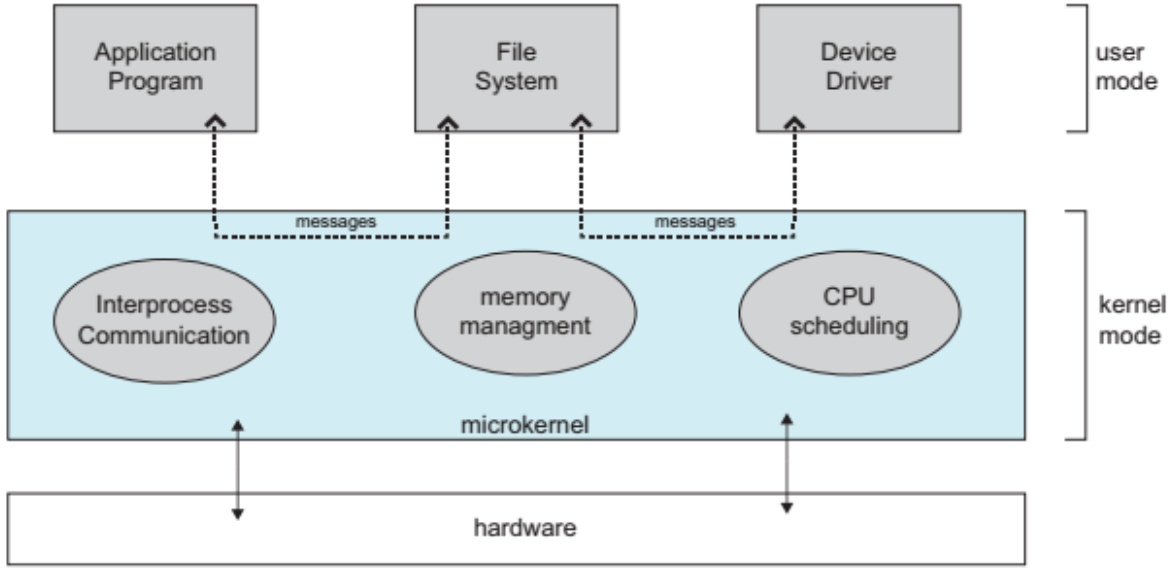


© by Dr. X. Sun

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 the microkernel:
  - ✓ provide essential services
  - ✓ provide communications among services and user programs based on message passing

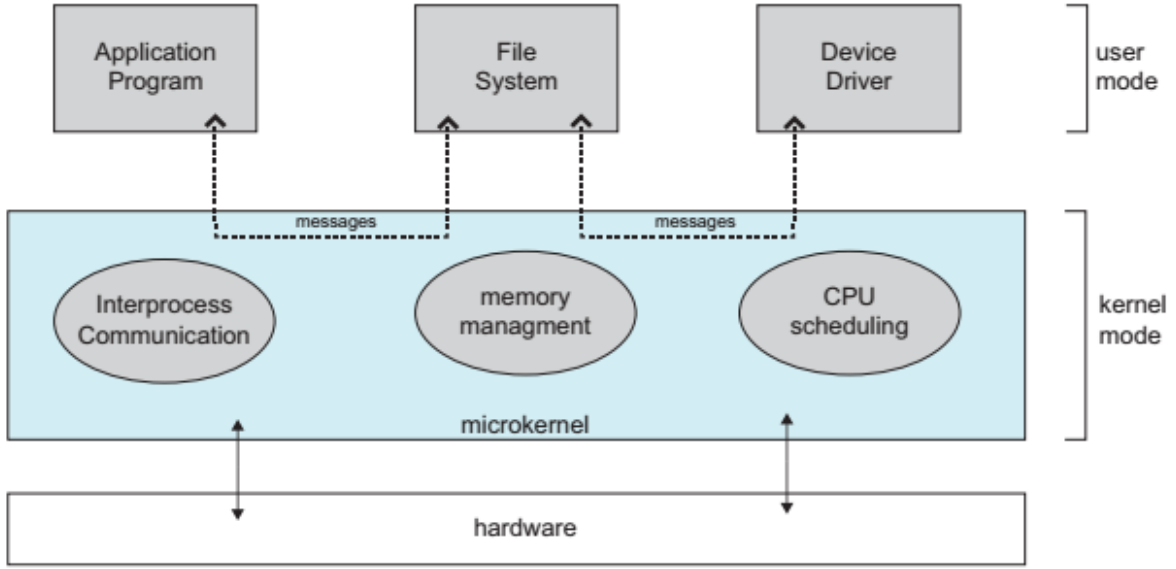


© by Dr. X. Sun

OS Structure

❑ Organization of Operating Systems—Microkernel approach

- ➢ Pros:
  - ✓ More reliable since less code is running in kernel mode.
  - ✓ More flexible for dynamical service/function configuration.
  - ✓ Easier to extend services.
- ➢ Cons
  - ✓ Overhead of user space to kernel space communications.

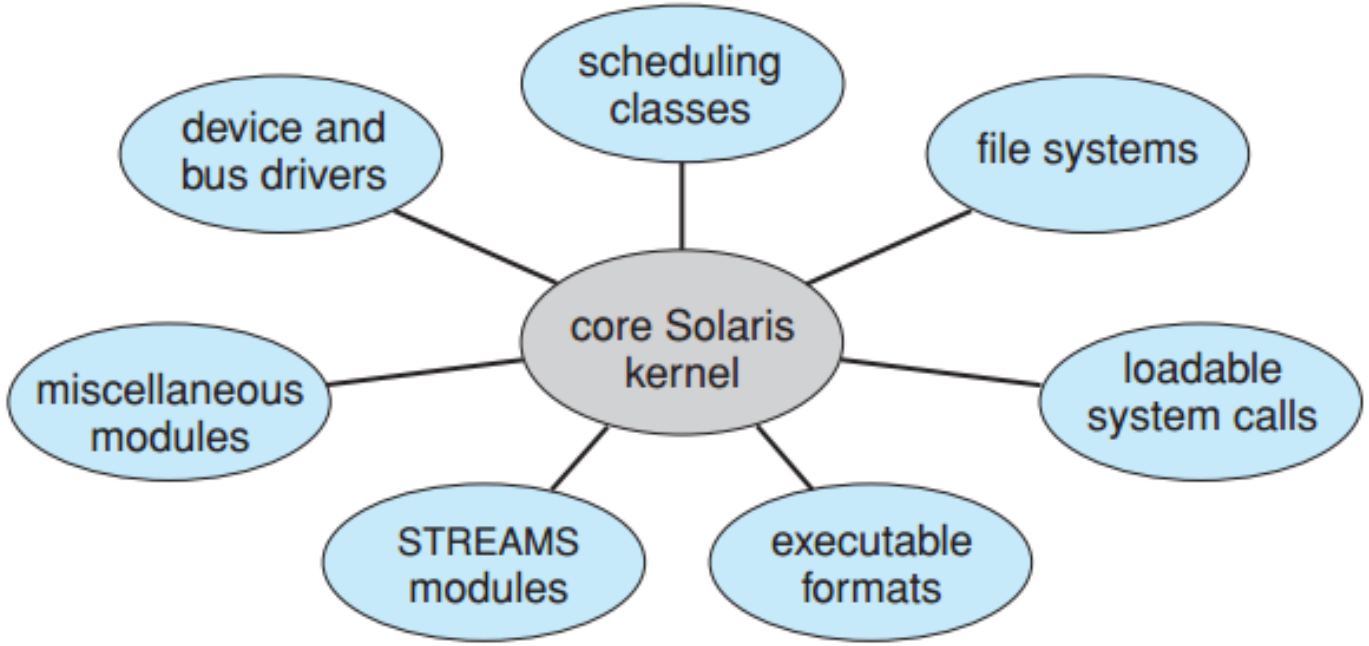


© by Dr. X. Sun

OS Structure

❑ Organization of Operating Systems—Module Approach

- Each service/function of kernel is implemented as a loadable module.
- It is more flexible than the Monolithic approach and more efficient than the Microkernel approach.
- Most of the current OSs are applies Hybrid structure.



© by Dr. X. Sun

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

© by Dr. X. Sun

OS Overall

- Usage share of operating systems
  - the percentage market share of the operating systems used in various devices, 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%

© by Dr. X. Sun