

Operating Systems

CS220

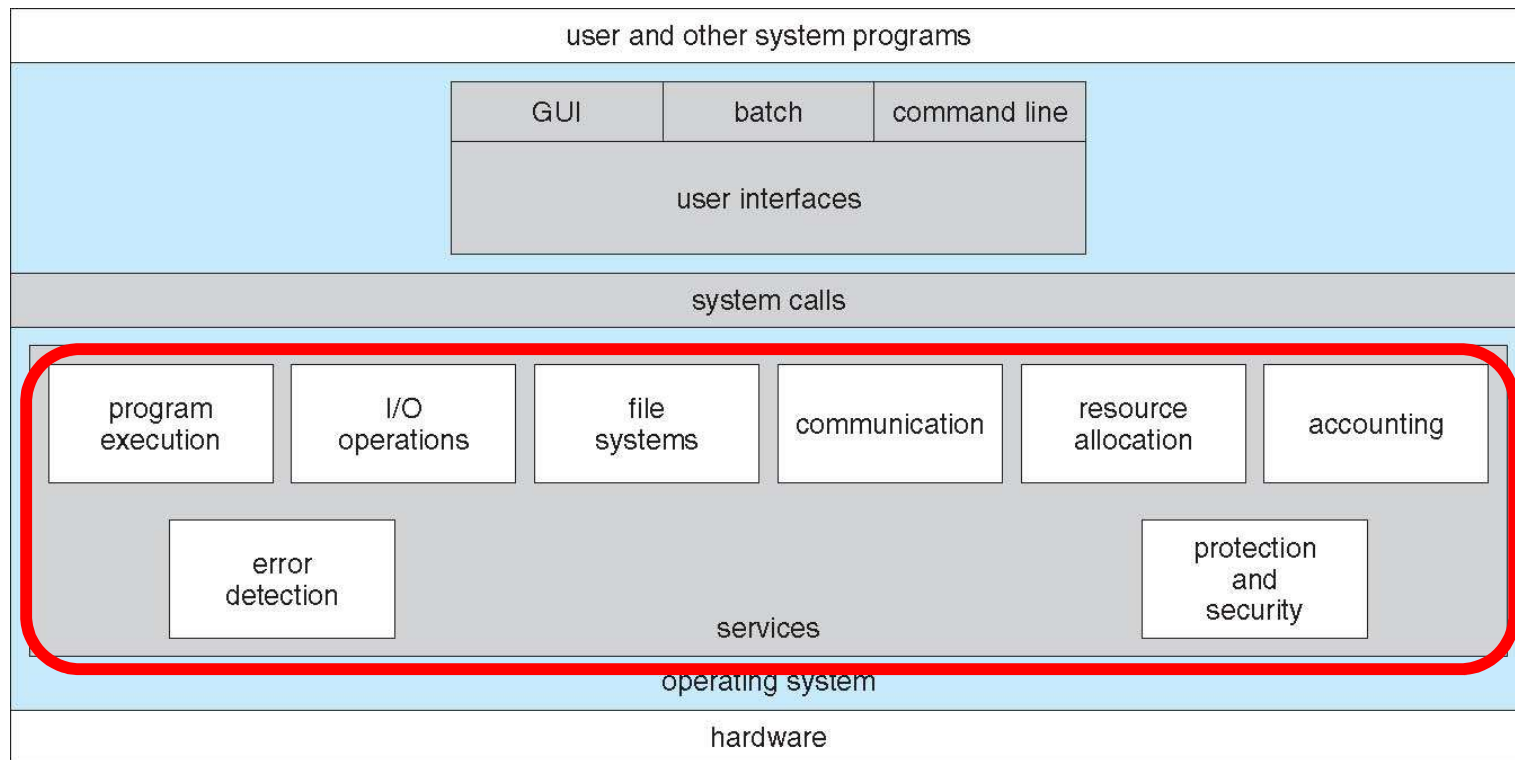
Lecture 3

OS Services

13th April 2021

By: **Dr. Rana Asif Rehman**

A View of Operating System Services



What Categories of Services Might be Provided by OS?

- Program execution
- OS Operations
- Process management
- Memory management
- Storage management
- I/O Subsystem
- Protection & Security

1. Program Execution

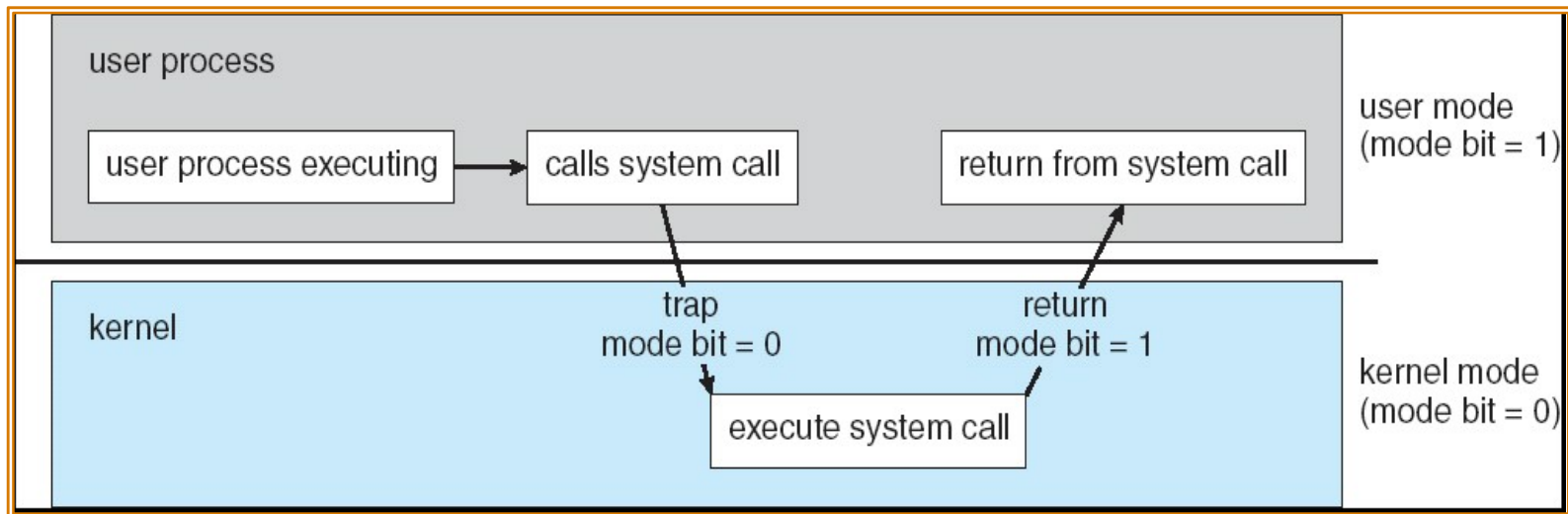
- **Multiprogramming** needed for efficiency of utilization
 - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
 - A subset of total jobs in system is kept in memory
 - One job selected and run via **job scheduling**
 - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing (multitasking)** is logical extension in which the CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
 - **Response time** should be $\ll 1$ second
 - Each user has at least one program executing in memory \rightarrow **process**
 - If several jobs ready to run at the same time \rightarrow **CPU scheduling**
 - If processes don't fit in memory, **swapping** moves them in and out to run
 - **Virtual memory** allows execution of processes not completely in memory

2. OS Operations

- **Interrupt** driven by hardware
- Software error or request creates **exception** or **trap**
 - Division by zero, request for operating system service
 - Other process problems include infinite loop, processes modifying each other or the operating system
 - **Dual-mode** operation allows OS to protect itself and other system components
- **User mode** and **kernel mode**
 - **Mode bit** provided by hardware
 - Provides ability to distinguish when system is running user code or kernel code
 - Some instructions designated as **privileged**, only executable in kernel mode
 - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
 - Set interrupt after specific period
 - Operating system decrements counter
 - When counter reaches zero, generate an interrupt
 - Setup before scheduling process to regain control or terminate program that exceeds allotted time



3. Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a **passive entity**, process is an **active entity**.
- Process needs resources to accomplish its task
 - CPU, memory, I/O, files
 - Initialization data
- Process termination requires reclaiming of any reusable resources
- Single-threaded process has one **program counter** specifying the location of the next instruction to execute
 - Process executes instructions sequentially, one at a time, until completion
- **Multi-threaded process has one program counter per thread**
- Typically system has many processes, some user, some operating system, running concurrently on one or more CPUs
 - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process communication
- Providing mechanisms for process synchronization
- Providing mechanisms for deadlock handling

4. Memory Management

- Memory management activities
 - Keeping track of which parts of memory are currently being used and by whom
 - Deciding which processes (or parts thereof) and data to move into and out of memory
 - Allocating and deallocating memory space as needed
- All data in memory before and after processing
- All instructions in memory to allow execution
- Memory management determines what is in memory
 - Optimizing CPU utilization & computer response to users

5. Storage Management

5.1. File-System management

- Files usually organized in directories
- Access control on most systems to determine who can access what
- OS activities include
 - Creating and deleting files and directories
 - Primitives to manipulate files and directories
 - Mapping files onto secondary storage
 - Backup files onto stable (non-volatile) storage media
- OS provides uniform, logical view of information storage
 - Abstracts physical properties to logical storage unit – file
 - Each medium is controlled by device (i.e., disk drive, tape drive)
 - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

5. Storage Management

5.2. Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
 - Free-space management
 - Storage allocation
 - Disk scheduling
 - Disk fragmentation / defragmentation
- Some storage need not be fast
 - Tertiary storage includes optical storage, magnetic tape
 - Still must be managed – by OS or applications
 - Varies between WORM (write-once, read-many-times) and RW (read-write)

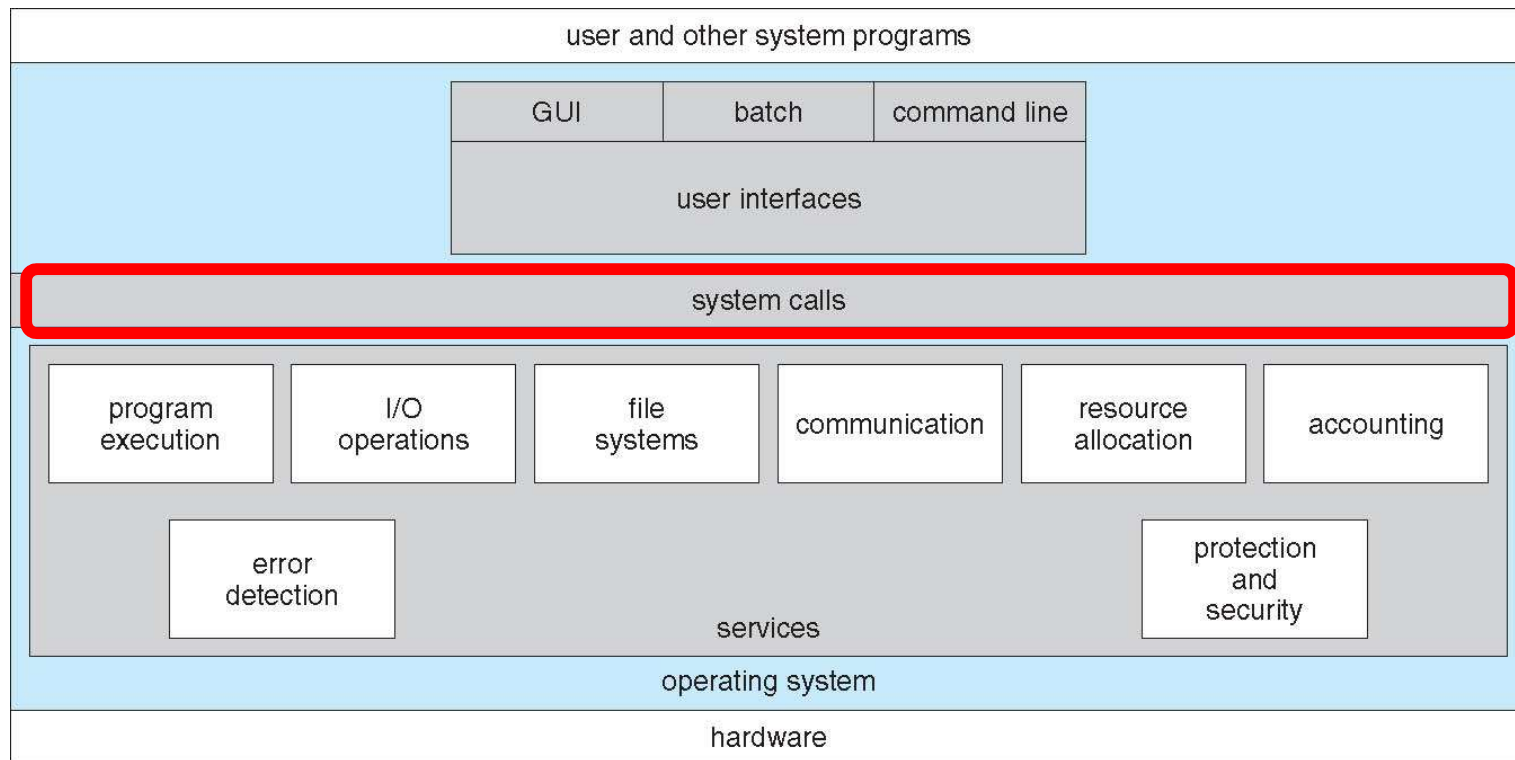
6. I/O Subsystem

- I/O Subsystem responsible for
 - Memory management of I/O including
 - **Buffering**: storing data temporarily while it is being transferred
 - **Caching**: storing parts of data in faster storage for performance
 - **Spooling**: the overlapping of output of one job with input of other jobs
 - General device-driver interface
 - Drivers for specific hardware devices
- One purpose of OS is to hide peculiarities of hardware devices from the user

7. Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
 - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
 - User identities (**user IDs**, security IDs) include name and associated number, one per user
 - User ID then associated with all files, processes of that user to determine access control
 - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
 - **Privilege escalation** allows user to change to an effective ID with more rights

System Calls

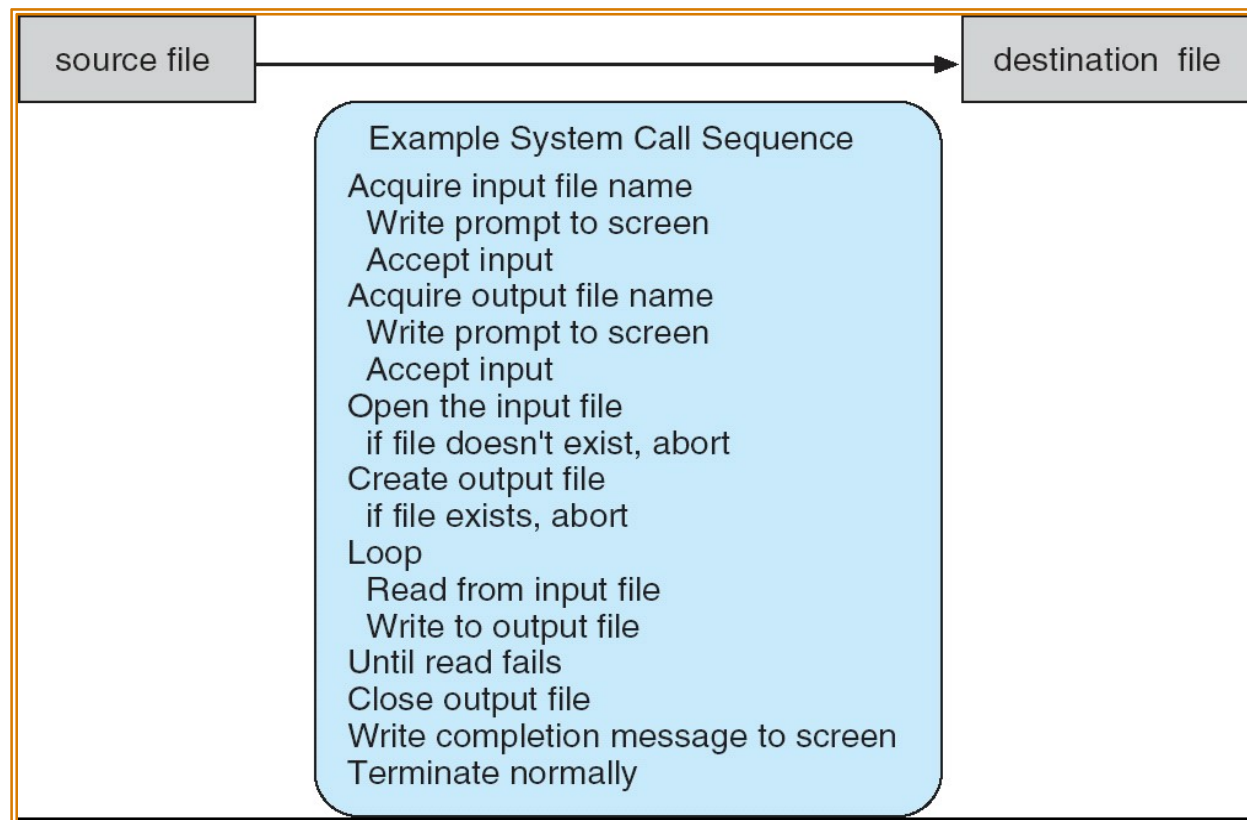


System Calls

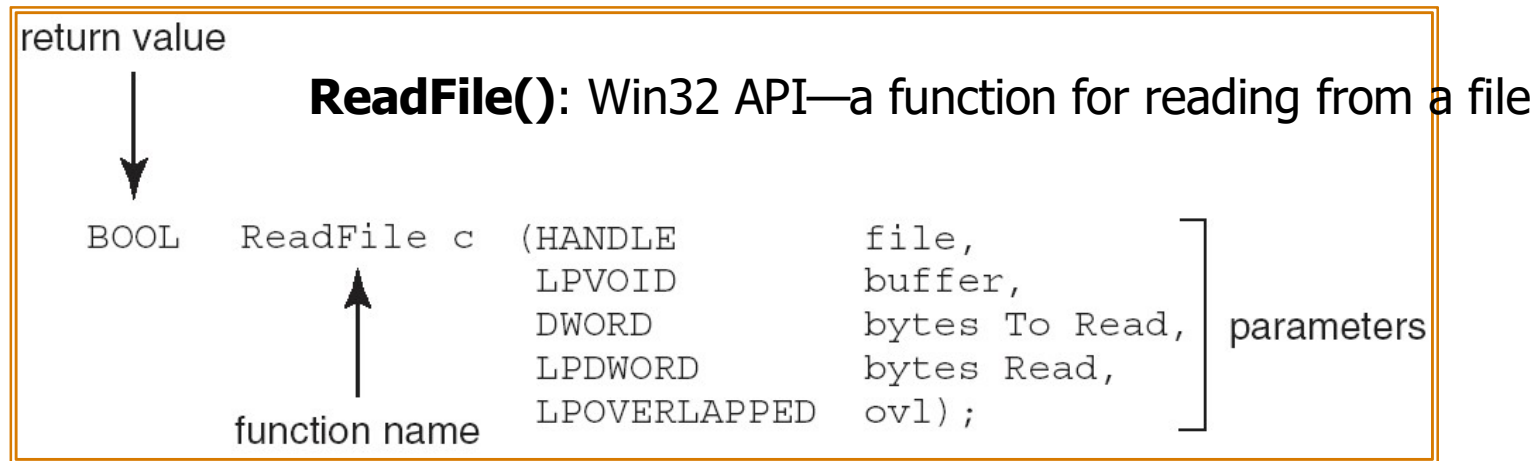
- Programming interface to the services provided by OS
- Typically written in a high-level language (C or C++)
- Mostly accessed by programs via a high-level **Application Program Interface (API)** rather than direct system call use
- Three most common APIs are **Win32 API** for Windows, **POSIX API** for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X), and **Java API** for the Java virtual machine (JVM)
- **Why use APIs rather than system calls?**

Example of System Calls

- System call sequence to copy the contents of one file to another file

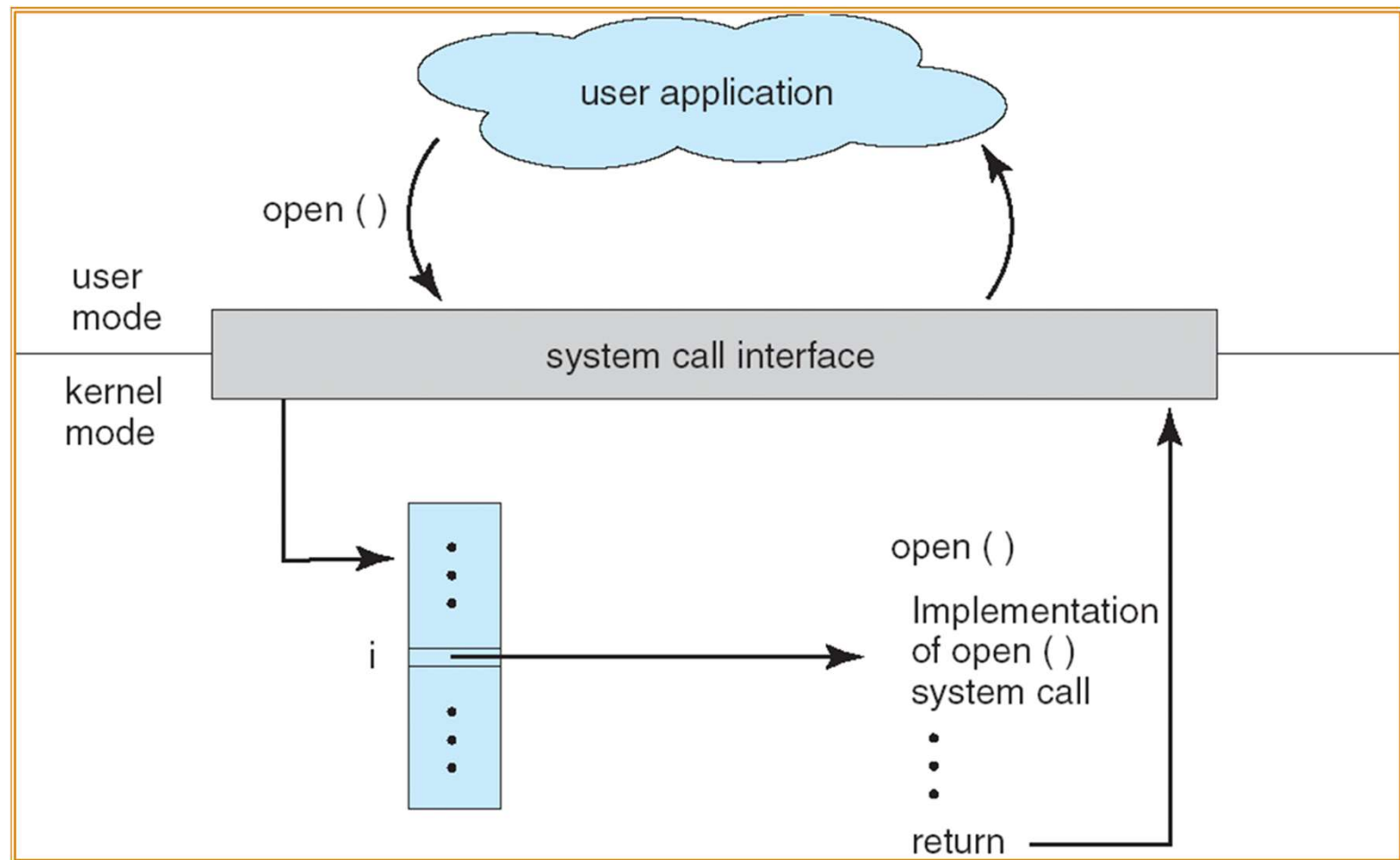


Example of Standard API



- A description of the parameters passed to ReadFile()
 - HANDLE file—the internal handle of the file to be read
 - LPVOID buffer---A buffer where the data will be read into and written from
 - DWORD bytesToRead—the number of bytes to be read into the buffer
 - LPDWORD bytesRead—the number of bytes read during the last read
 - LPOVERLAPPED ovl—indicates if overlapped I/O is being used

API – System Call – OS Relationship

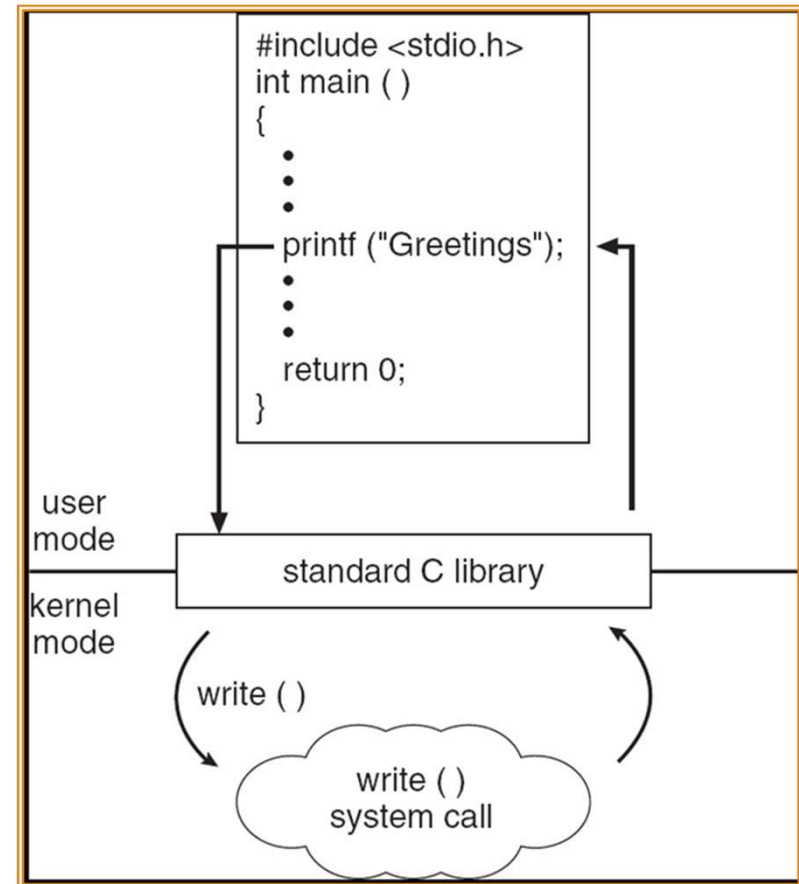


System Call Implementation

- Typically, a number associated with each system call
 - System-call interface maintains a table indexed according to these numbers
- The system call interface invokes 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 the call
 - 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)

Standard C Library Example

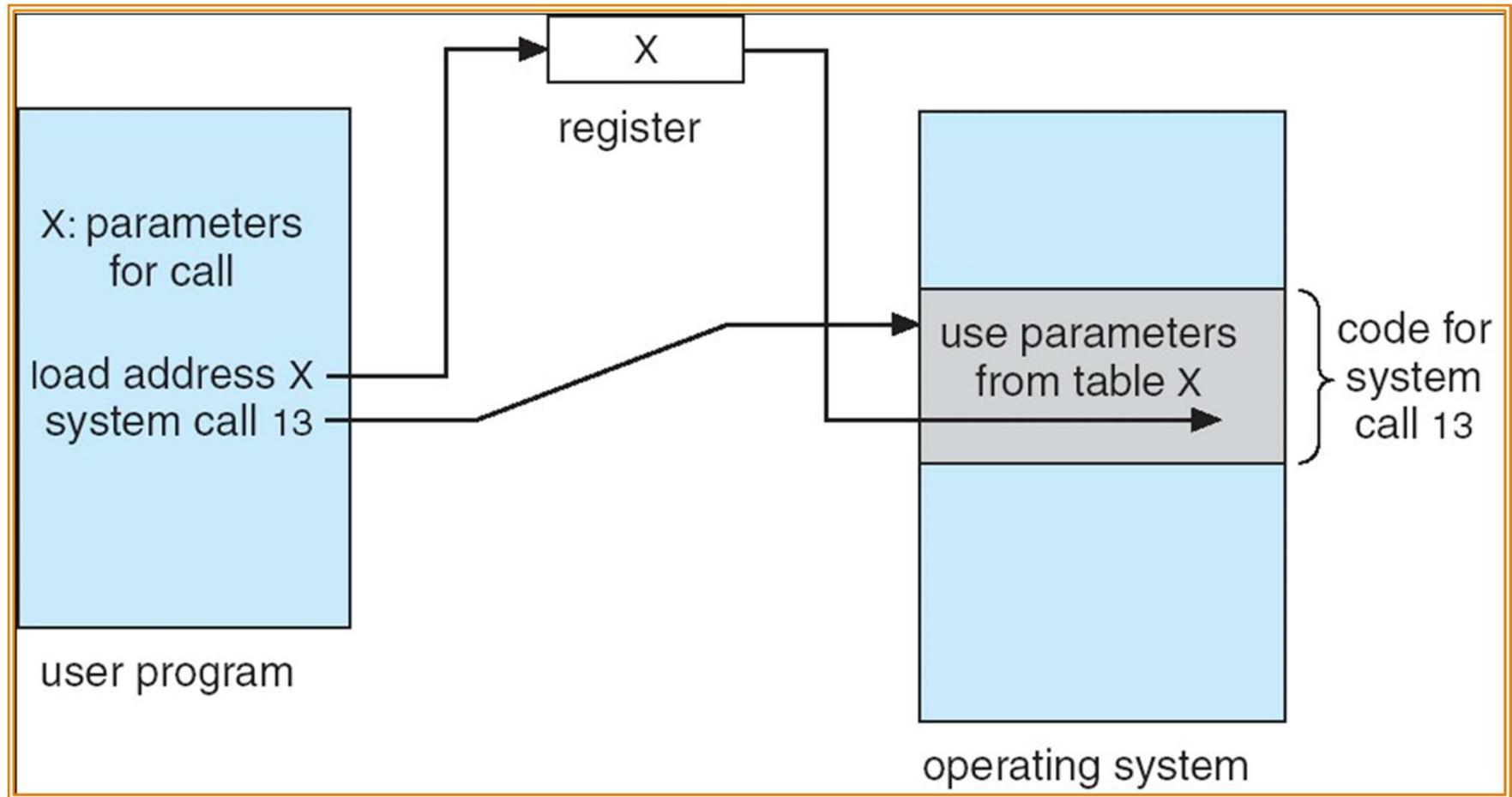
- C program invoking printf() library call, which calls write() system call



System Call Parameter Passing

- Often more information is required than simply identity of desired system call
 - Exact type and amount of information vary according to OS and call
- Methods used to pass parameters to OS
 - Simplest: *pass the parameters in registers*
 - In some cases, may be more parameters than registers
 - *Parameters stored in a block, or table, in memory*, and address of block passed as a parameter in a register
 - This approach taken by Linux and Solaris
 - *Parameters placed, or pushed, onto the stack* by the program and popped off the stack by the operating system
 - Block and stack methods do not limit the number or length of parameters being passed

Parameter Passing via Table



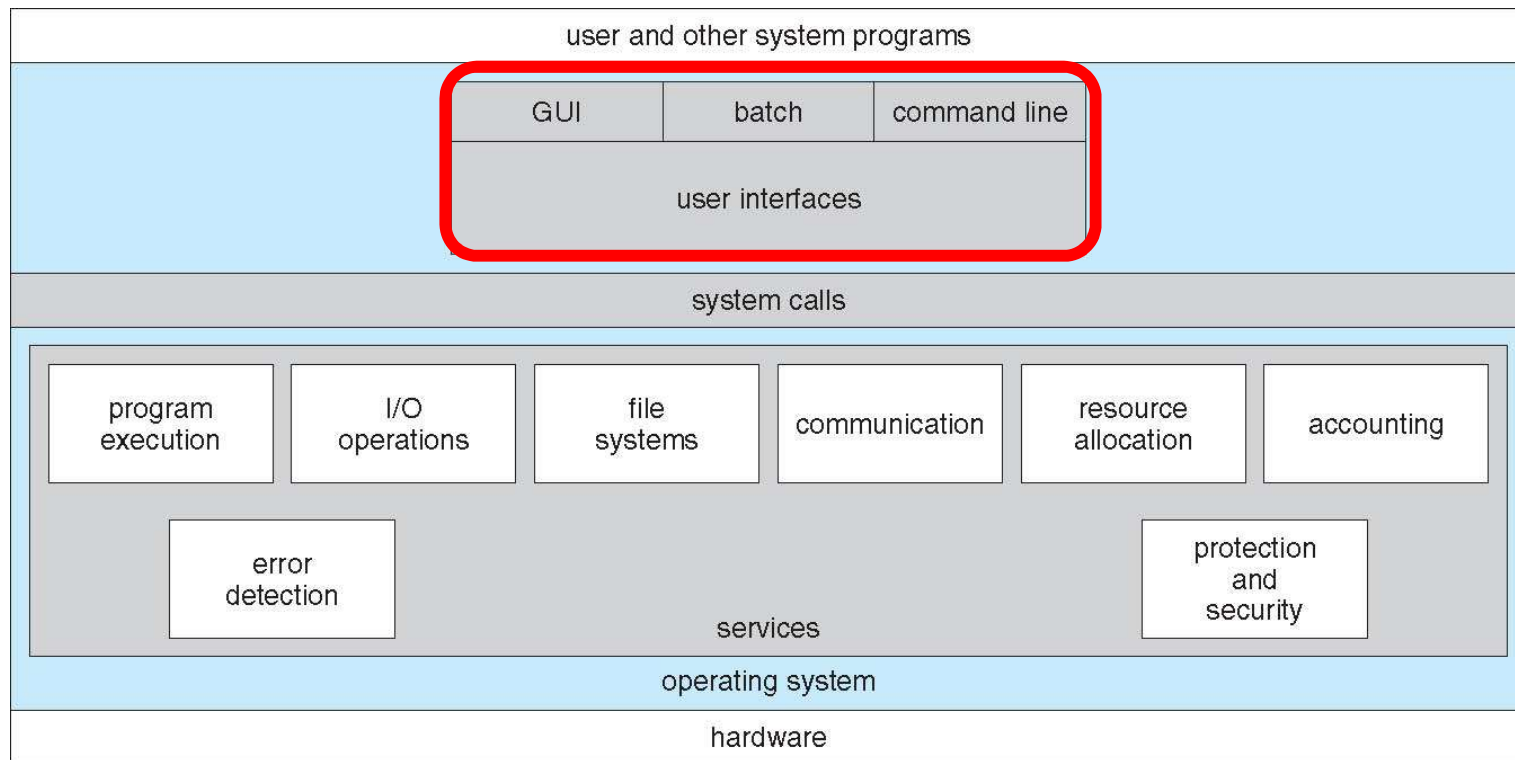
Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications

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()

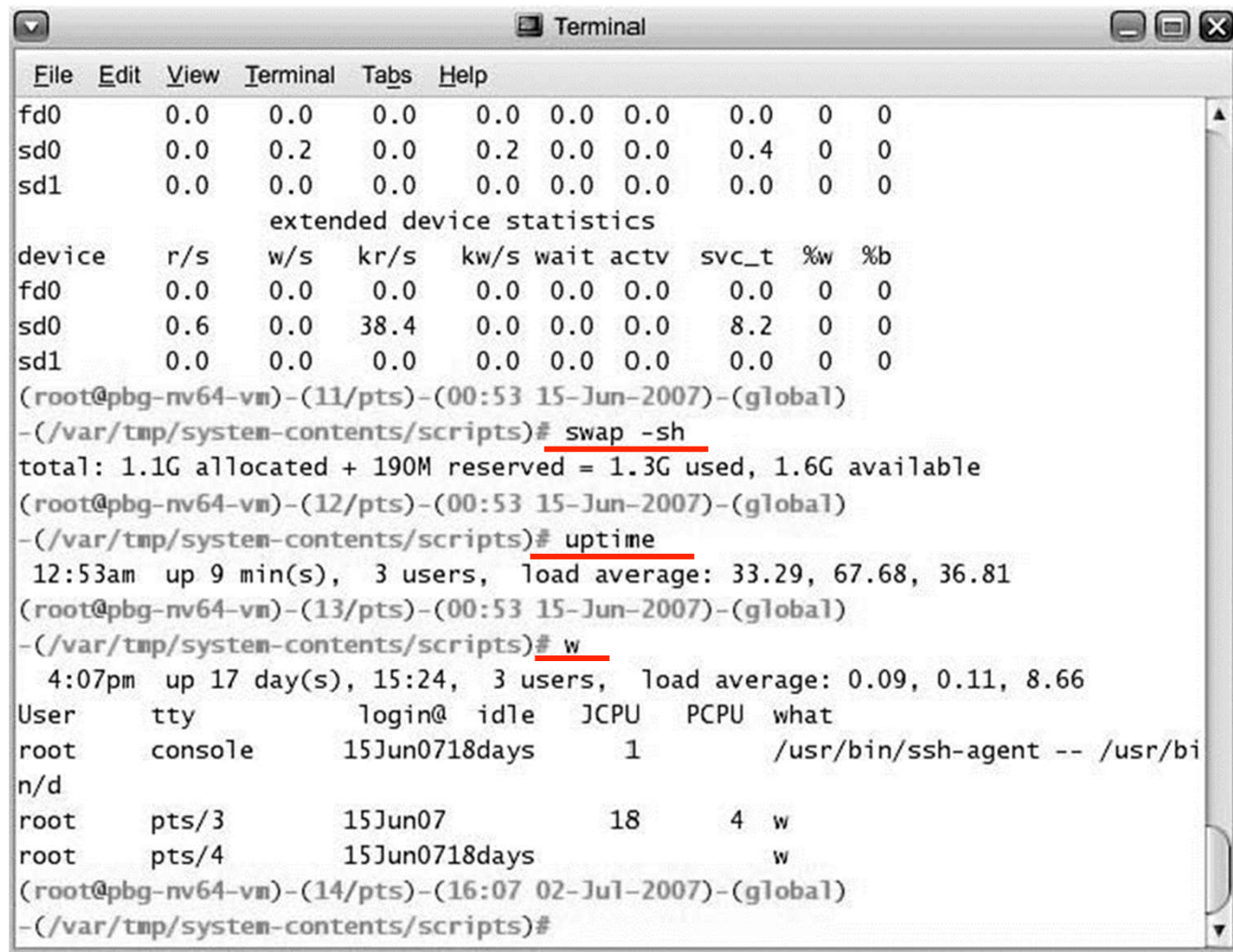
User Interface



User Operating System Interface - CLI

- Command-Line Interface (CLI) allows direct command entry
 - Sometimes implemented in kernel, sometimes by systems program
 - Sometimes multiple flavors implemented – [shells](#)
- Primarily receives command from user and executes it
 - Sometimes commands built-in, sometimes just names of programs, sometimes a combination
 - If the latter, adding new features doesn't require CLI modification

Bourne Shell Command Interpreter



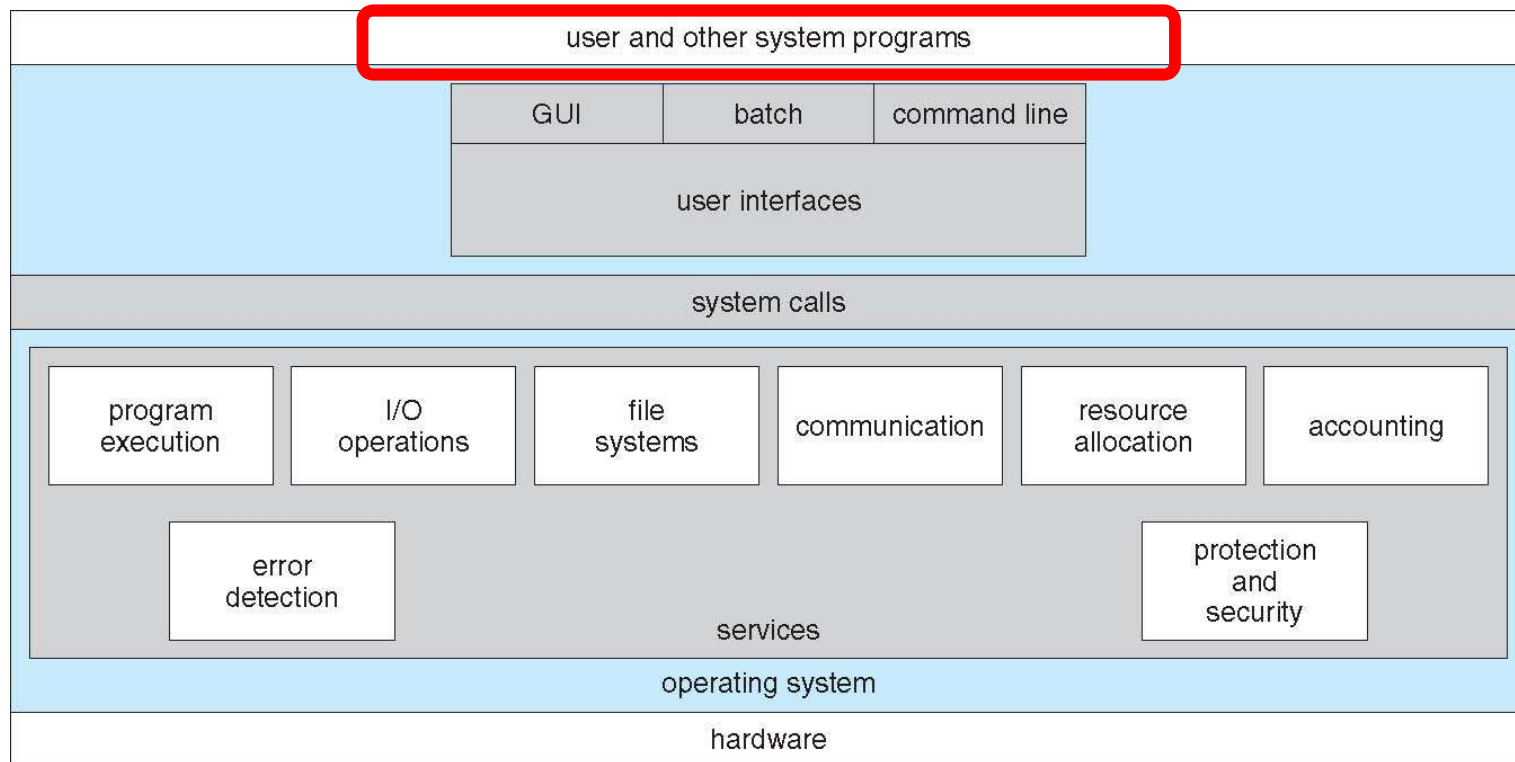
A terminal window titled "Terminal" showing the output of several system commands. The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The output includes disk statistics for fd0, sd0, and sd1, extended device statistics, the output of the `swap -sh` command, the output of the `uptime` command, and the output of the `w` command.

```
File Edit View Terminal Tabs Help
fd0      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
sd0      0.0    0.2    0.0    0.2  0.0  0.0    0.4  0  0
sd1      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
      extended device statistics
device   r/s    w/s    kr/s    kw/s wait actv  svc_t  %w  %b
fd0      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
sd0      0.6    0.0   38.4    0.0  0.0  0.0    8.2  0  0
sd1      0.0    0.0    0.0    0.0  0.0  0.0    0.0  0  0
(root@pbg-nv64-vn)-(11/pts)-(00:53 15-Jun-2007)-(global)
- (/var/tmp/system-contents/scripts)# swap -sh
total: 1.1G allocated + 190M reserved = 1.3G used, 1.6G available
(root@pbg-nv64-vn)-(12/pts)-(00:53 15-Jun-2007)-(global)
- (/var/tmp/system-contents/scripts)# uptime
12:53am up 9 min(s), 3 users, load average: 33.29, 67.68, 36.81
(root@pbg-nv64-vn)-(13/pts)-(00:53 15-Jun-2007)-(global)
- (/var/tmp/system-contents/scripts)# w
 4:07pm up 17 day(s), 15:24, 3 users, load average: 0.09, 0.11, 8.66
User      tty          login@ idle   JCPU   PCPU   what
root      console      15Jun07 18days 1      /usr/bin/ssh-agent -- /usr/bi
n/d
root      pts/3        15Jun07 18      4      w
root      pts/4        15Jun07 18days 0      w
(root@pbg-nv64-vn)-(14/pts)-(16:07 02-Jul-2007)-(global)
- (/var/tmp/system-contents/scripts)#
```

User Operating System Interface - GUI

- User-friendly **desktop** metaphor 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**))
 - Invented at Xerox PARC
- Many systems now include both CLI and GUI interfaces
 - Microsoft Windows is GUI; CLI is “command” shell
 - Apple Mac OS X has “Aqua” GUI; UNIX kernel underneath and multiple shells available
 - Solaris has multiple GUIs; CLI is multiple shells

User and System Programs



System Programs

- System programs provide a convenient environment for program development and execution. Including:
 - File manipulation
 - Status information
 - Programming language support
 - Program loading and execution
 - Communications
 - Application programs
- Most users' view of the operating system is defined by system programs, not the actual system calls

References

- Operating System Concepts (Silberschatz, 8th edition)
Chapter 1, 2.1-2.5