MODULE 1 :
Introduction to Unix

What is an Operating System?

- Operating System(OS) is a system software
- OS can be defined as an organized collection of software consisting of procedures for operating a computer
- OS provides an environment for execution of programs
- OS acts as an interface between the user and the hardware of the computer system

What is an Operating System?(contd.)

- Operating system interacts with user in two ways
- Operating system commands
  - ✓ Enables user to interact directly with the operating system
- Operating system calls
  - ✓ Provides an interface to a running program and the operating system
  - ✓ System calls in UNIX are written in C

What is UNIX?

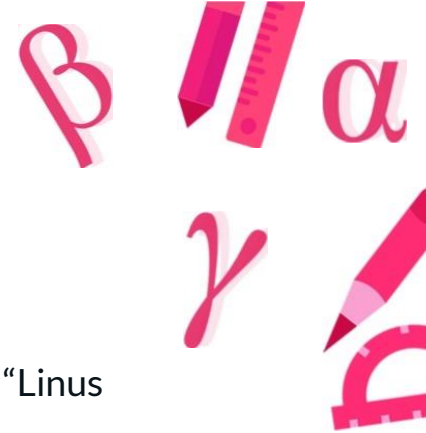
- Unix is a multi-user, multi-tasking operating system
- You can have many users logged into a system simultaneously, each running many programs
- It's the kernel's job to keep each process and user separate and to regulate access to system hardware, including cpu, memory, disk and other I/O devices

History of UNIX

- First Version was created in Bell Labs in 1969
- Some of the Bell Labs programmers who had worked on this project, Ken Thompson, Dennis Ritchie, Rudd Canaday, and Doug McIlroy designed and implemented the first version of the Unix File System along with a few utilities. It was given the name UNIX by Brian Kernighan.
- 1973 Unix is re-written mostly in C, a new language developed by Dennis Ritchie
- Being written in this high-level language greatly decreased the effort needed to port it to new machines

History of UNIX (contd.)

- 1977 There were about 500 Unix sites world-wide.
- 1980 BSD 4.1 (Berkeley Software Distribution or Berkeley Unix because it was based on the source code of the original Unix.)
- 1983 SunOS, BSD 4.2, System V
- 1991 Linux was originated.



What is Linux ?

- Linux is a free Unix-type operating system originally created by “Linus Torvalds” with the assistance of developers around the world.
- It originated in 1991 as a personal project of Linus Torvalds, a Finnish graduate student.
- The Kernel version 1.0 was released in 1994 and today the most recent stable version is 4.18.5
- Developed under the GNU General Public License , the source code for Linux is freely available to everyone.

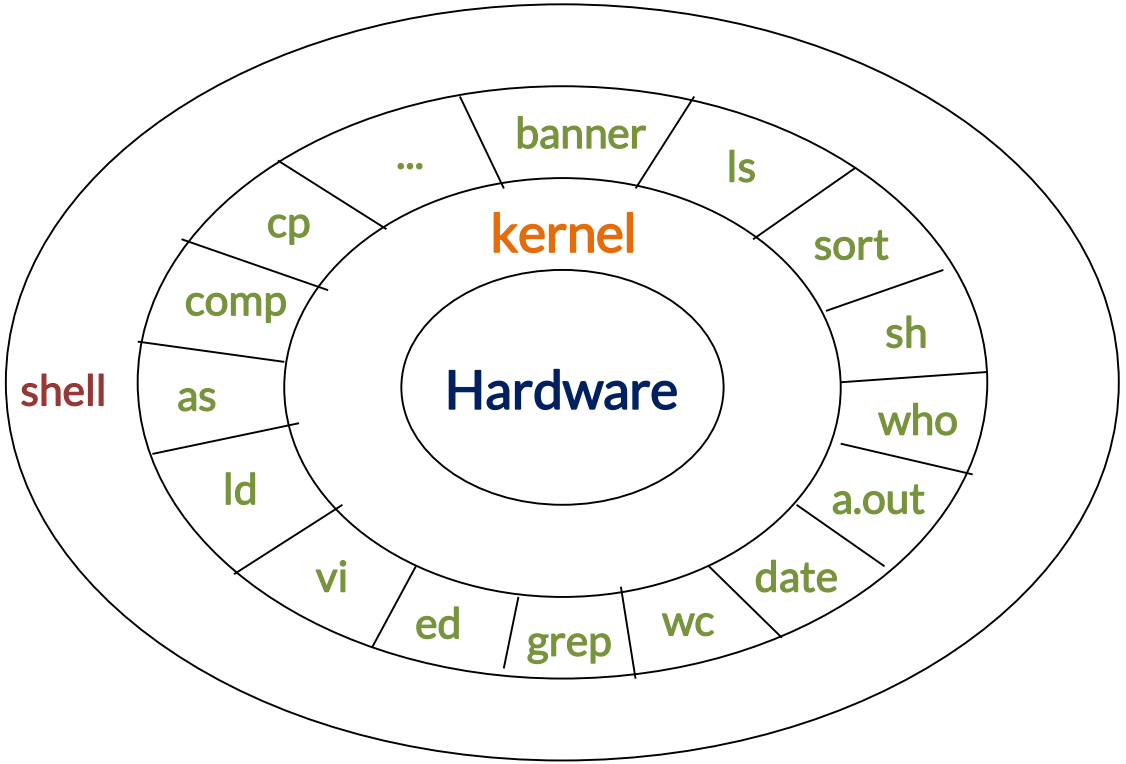
Features of Linux

- UNIX-like operating system.

Features:

- Preemptive multitasking, multiuser system.
- Virtual memory (protected memory, paging)
- Shared libraries
- Demand loading, dynamic kernel modules
- Shared copy-on-write executables
- TCP/IP networking
- SMP support
- Open source

Layered Architecture



Unix System Architecture

Unix system follows a layered approach. It has four layers

- The innermost layer is the hardware layer
- In the second layer, the kernel is placed "*Core of OS*"
- The utilities and other application programs form the third layer
- Fourth layer is the one with which the user actually interacts.

What is a Kernel ?

- AKA: executive, system monitor.
- Controls and mediates access to hardware.
- Implements and supports fundamental abstractions:
  - ✓ *Processes, files, devices etc.*

- Schedules / allocates system resources:
  - ✓ *Memory, CPU, disk, descriptors, etc.*

- Enforces security and protection.
- Responds to user requests for service (system calls).

What is a Shell ?

Shell is a utility program that comes with the **UNIX** system.

Features of Shell are:

- Interactive Processing
- Background Processing
- I/O Redirection
- Pipes
- Shell Scripts
- Shell Variables
- Programming Constructs

Where is Linux used ?

Linux is simply a computer operating system, so its uses are as diverse as any other Operating System. It is popular in certain areas, though:

- Web Serving
- Networking
- Databases
- Desktops
- Scientific Computing
- Home Computing
- Smartphones

Distributions of Linux

Here are some of the most popular linux distributions:

- RedHat Enterprise Linux (RHEL)
- Fedora
- Ubuntu
- Debian
- SuSE Linux Enterprise Server (SLES)
- OpenSuSE
- Linux Mint

Logging in to Linux System

There are many ways that we can access a UNIX system. The main mode of access to UNIX machine is through a terminal, which usually includes a keyboard , and a video monitor.

For each terminal connected to the UNIX system, the Kernel runs a process called a `tty` that accepts input from the terminal, and sends output to the terminal.

Logging in

Logging in to a UNIX system requires two pieces of information:

A **username**, and a **password**. When we establish a connection to UNIX system, we are given a login prompt that looks like –

```
login:
```

Logging out

When we are ready to disconnect the session, type the command -

```
exit
```

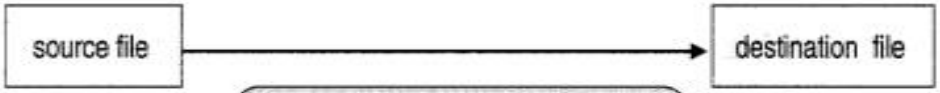
Some shells will recognize other commands to log you out, like "`logout`" or even "`bye`".

System Calls

- All operating systems provides some mechanism to request services from the kernel.
- The system call is the fundamental interface between an application and the Linux kernel.
- In Linux system, there is an interface layer between the user space and kernel space. This layer consists of library made up of functions that communicate between normal user applications and the Kernel.
- This implementation of C library is called libc/glibc. This provides wrapper functions for the system calls.

Example on how System Calls are used

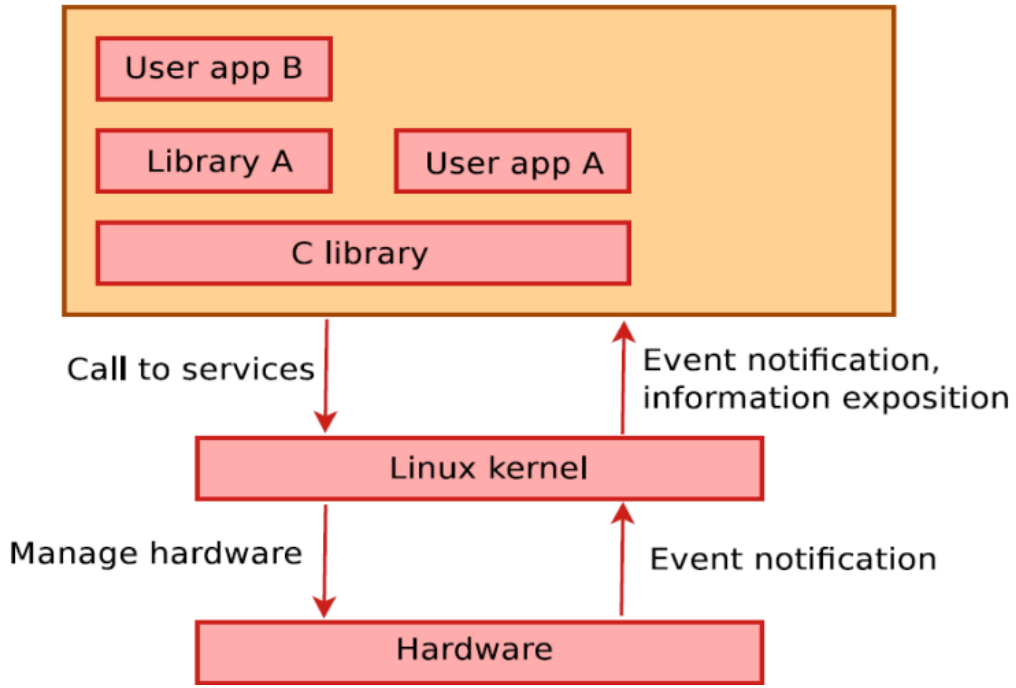
- Writing a simple program to read data from one file and copy them to another file
- Program inputs
  - ✓ Name of two files
- Program opens the input file and create output file and open it
- Each of these operations require another system calls
- Now that both files are setup, enter into the loop that read from input file (system call) and writes into the output file (another system call



Example System Call Sequence

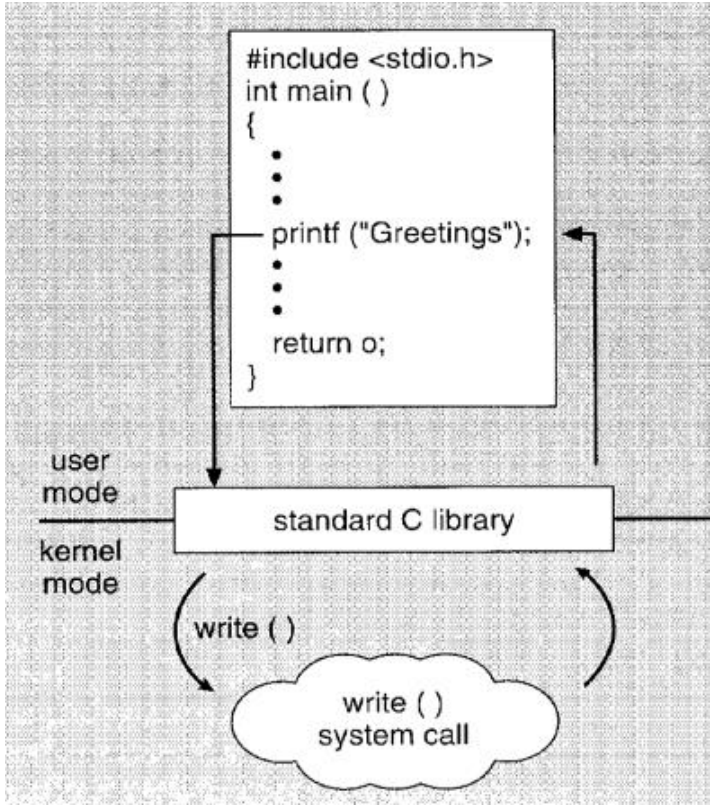
Acquire input file name
  Write prompt to screen
  Accept input
Acquire output file name
  Write prompt to screen
  Accept input
Open the input file
  if file doesn't exist, abort
Create output file
  if file exists, abort
Loop
  Read from input file
  Write to output file
Until read fails
Close output file
Write completion message to screen
Terminate normally

Interfacing between User and Kernel Space



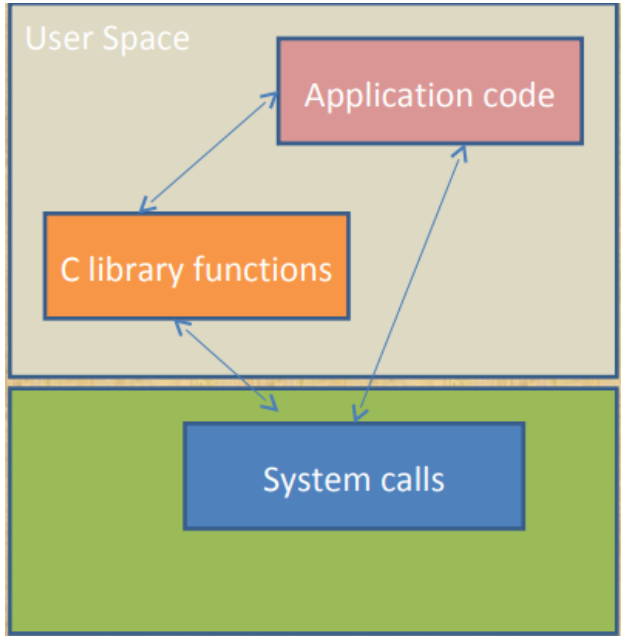
C library Interface

- The standard C library provides a portion of the system call interface for many versions of UNIX and Linux
- For example, the ‘printf’ statement. The C library interprets this call and invokes the necessary system call(s) in the operating system in this instance, the write() system call
- The C library takes the value returned by write() and passes it back to the user program.


#include <stdio.h>
int main ( )
{
printf ("Greetings");
return o;
}
user mode
kernel mode
standard C library
write ( )
write ( )
system call

System Calls and Library Functions

- **Examples of systems calls**
  - ✓ read() / write ()
  - ✓ fork() / exec ()
- **Example of library functions**
  - ✓ date / time
  - ✓ strcpy / atoi / prinf



System Call Categories

System calls can be roughly grouped into five major categories:

- Process management
- File management
- Device management
- Information/Maintenance
- Communication

To get the details of the available system calls in your kernel use the command: `man syscalls`

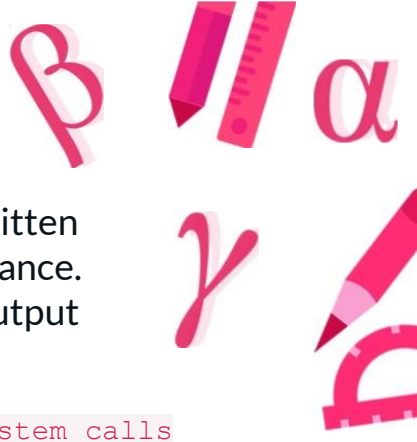
Analyzing System Calls with strace

```
❯ whatis strace
strace (1)           - trace system calls and signals
```

Depending on what you want to analyze, you can (just like with debuggers such as gdb) either newly start a command either with strace, or you can attach strace to an already running process and analyze it:

```
$ strace cat file1.txt
```

Usually strace outputs one line per system call. This line contains the name of the kernel routine and their parameters, as well as the return value of the system call.



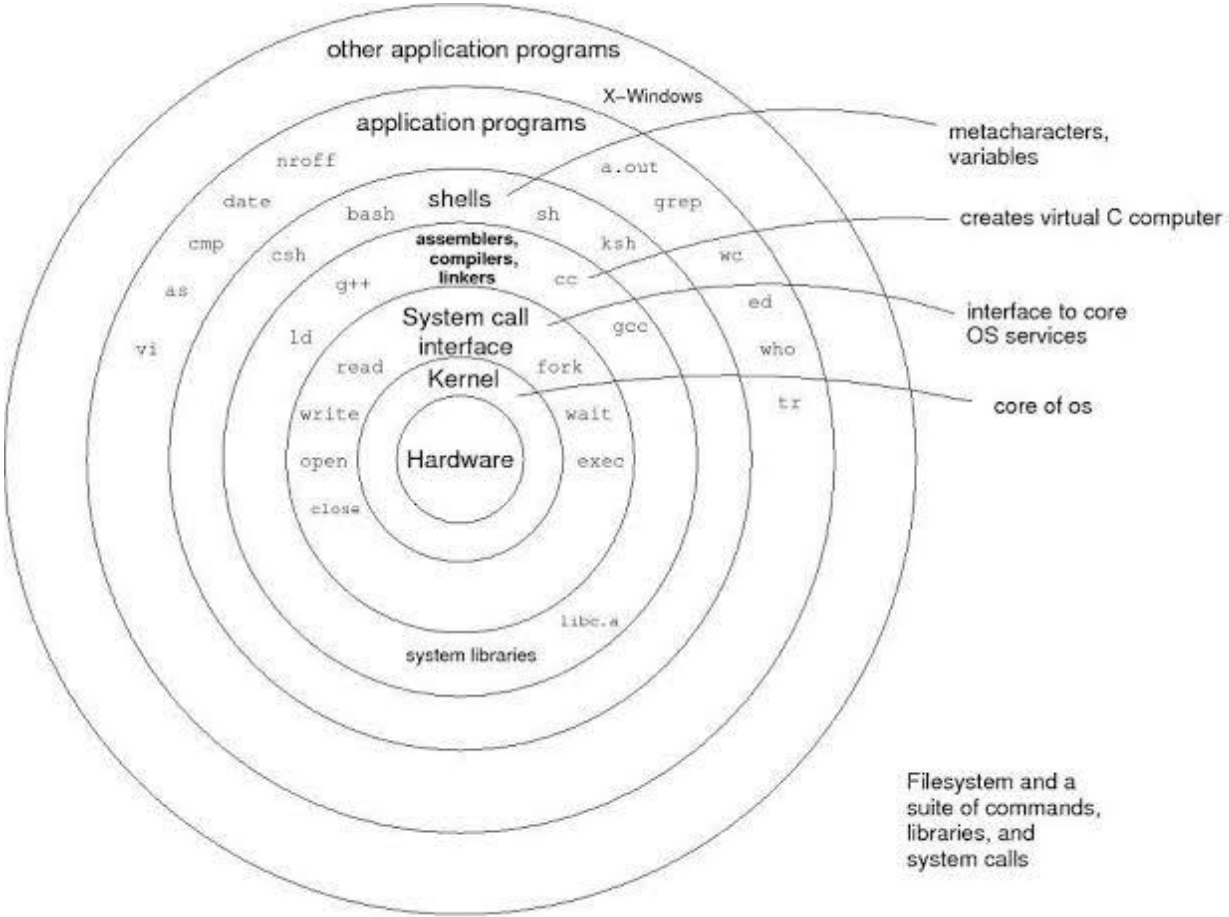
The output of strace can quickly get very large, and the synchronously written output to stderr or even to a log file can considerably slow down performance. If you know exactly which system calls are of interest, you can limit the output to one or few kernel calls -

```
$ strace -e open cat file1.txt        -   Will only capture open system calls

$ strace -e  open,read,write cat file1.txt   -  Captures open,read & write syscalls
```

strace can also simultaneously trace several processes (`option -p PID`) or follow all child processes (`with option -f`). In these cases, each line of strace output will start with the PID of corresponding process.

```
$ strace -e open,read,write -f -p "PID"      -  Captures syscalls in follow mode
```



Conceptual Architecture of UNIX SYSTEMS

Thank You!