Video Lecture # 01
Introduction to System Programming

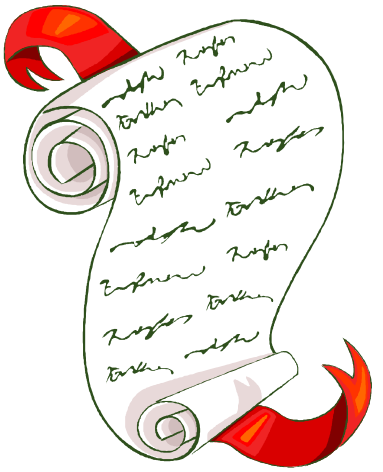
Course: System Programming

Instructor: Arif Butt

Punjab University College of Information Technology (PUCIT)
University of the Punjab

Today's Agenda

- Course Information
- Application vs System Programming
- Discussion on course Matrix



Course Info

- **Required Textbook:** Advanced Programming in UNIX Environment 3rd Edition Ritchard Stevens

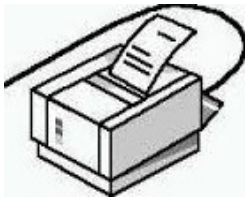
- **Prerequisites :** Video Lectures on C-Refresher
  Video Lectures on Operating System with Linux

- **Lectures Slides:** http://arifbutt.me
- **Resources Website:** http://arifbutt.me
- **Codes Available at:** https://bitbucket.org/arifpucit/spvl-repo

- **24 hour turnaround for email:** arif@pucit.edu.pk

Application Programmer VS System Programmer Perspective

Application Programmer Perspective

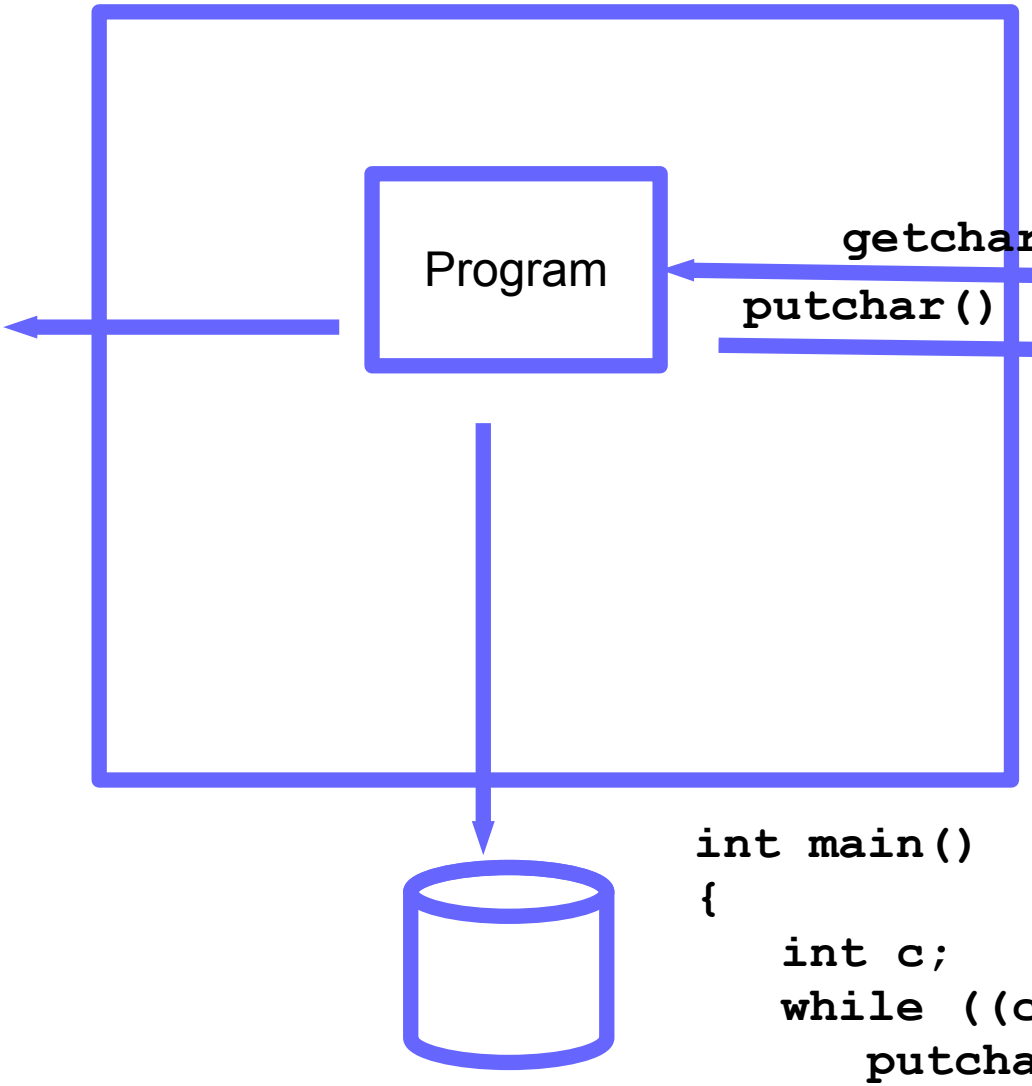


Scanner

Speaker

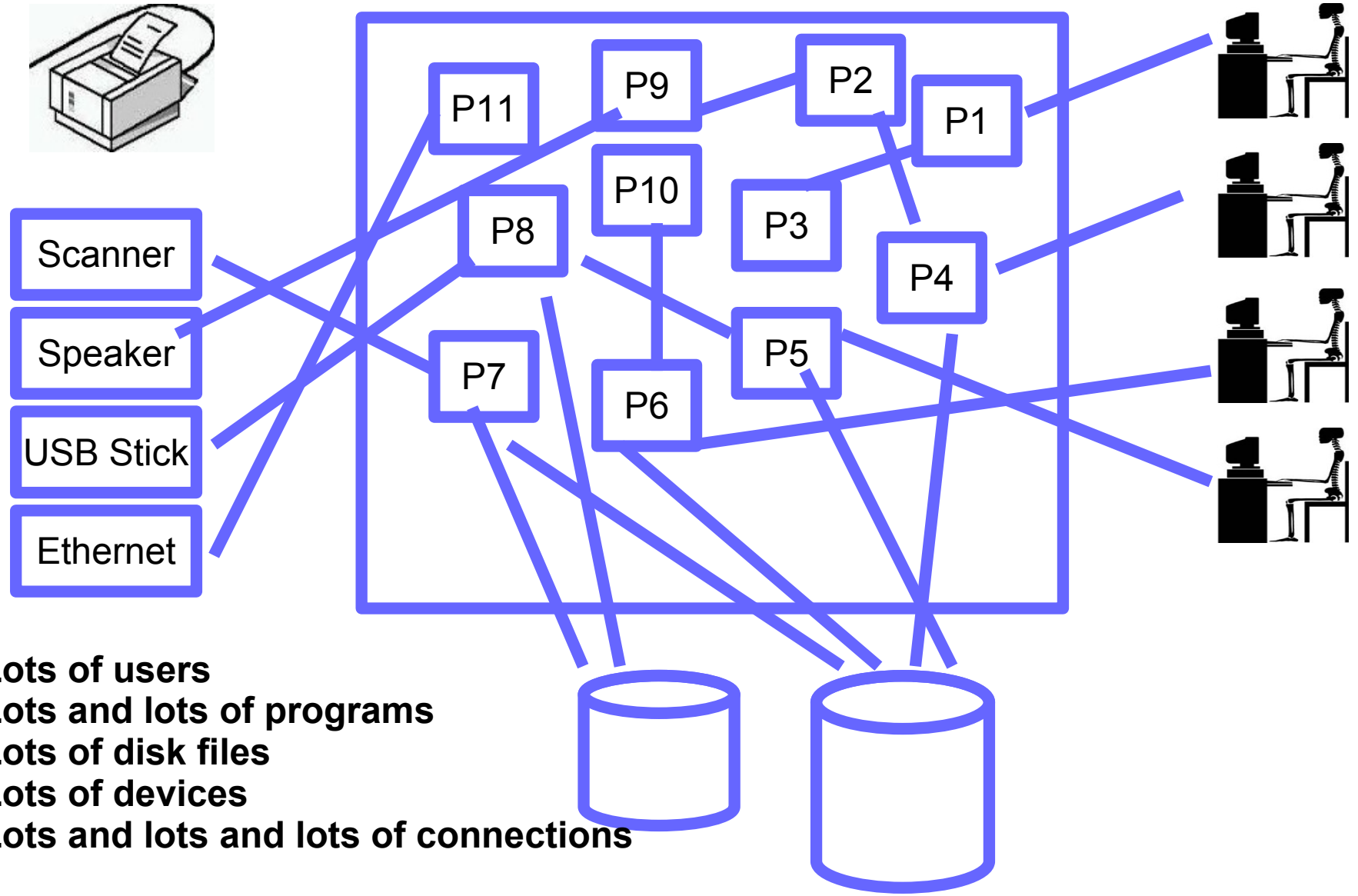
USB Stick

Ethernet



```
int main()
{
    int c;
    while ((c = getchar()) != EOF)
        putchar(c);
    return 0;
}
```

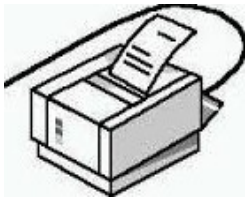
System Programmer Perspective



Lots of users
Lots and lots of programs
Lots of disk files
Lots of devices
Lots and lots and lots of connections

Who is managing all these resources and connecting various devices to correct programs

System Programmer Perspective

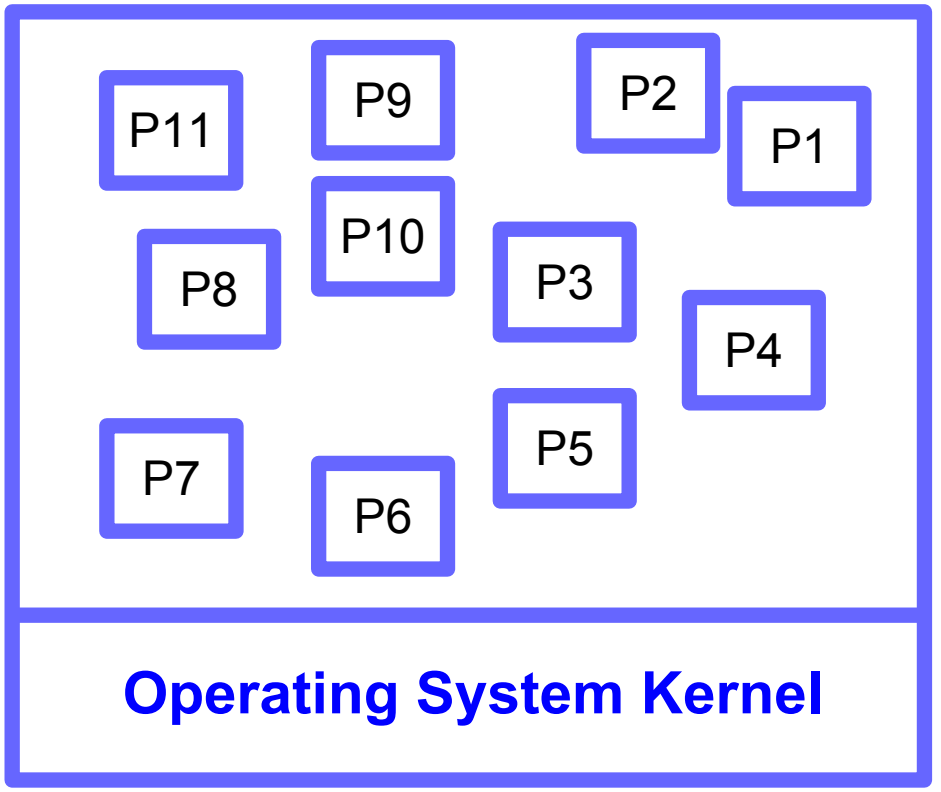


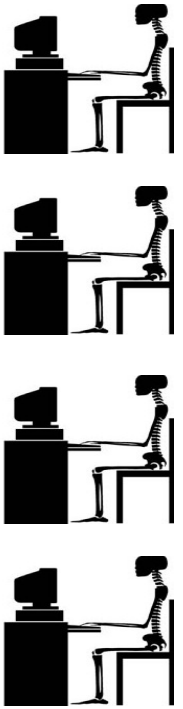
Scanner

Speaker

USB Stick

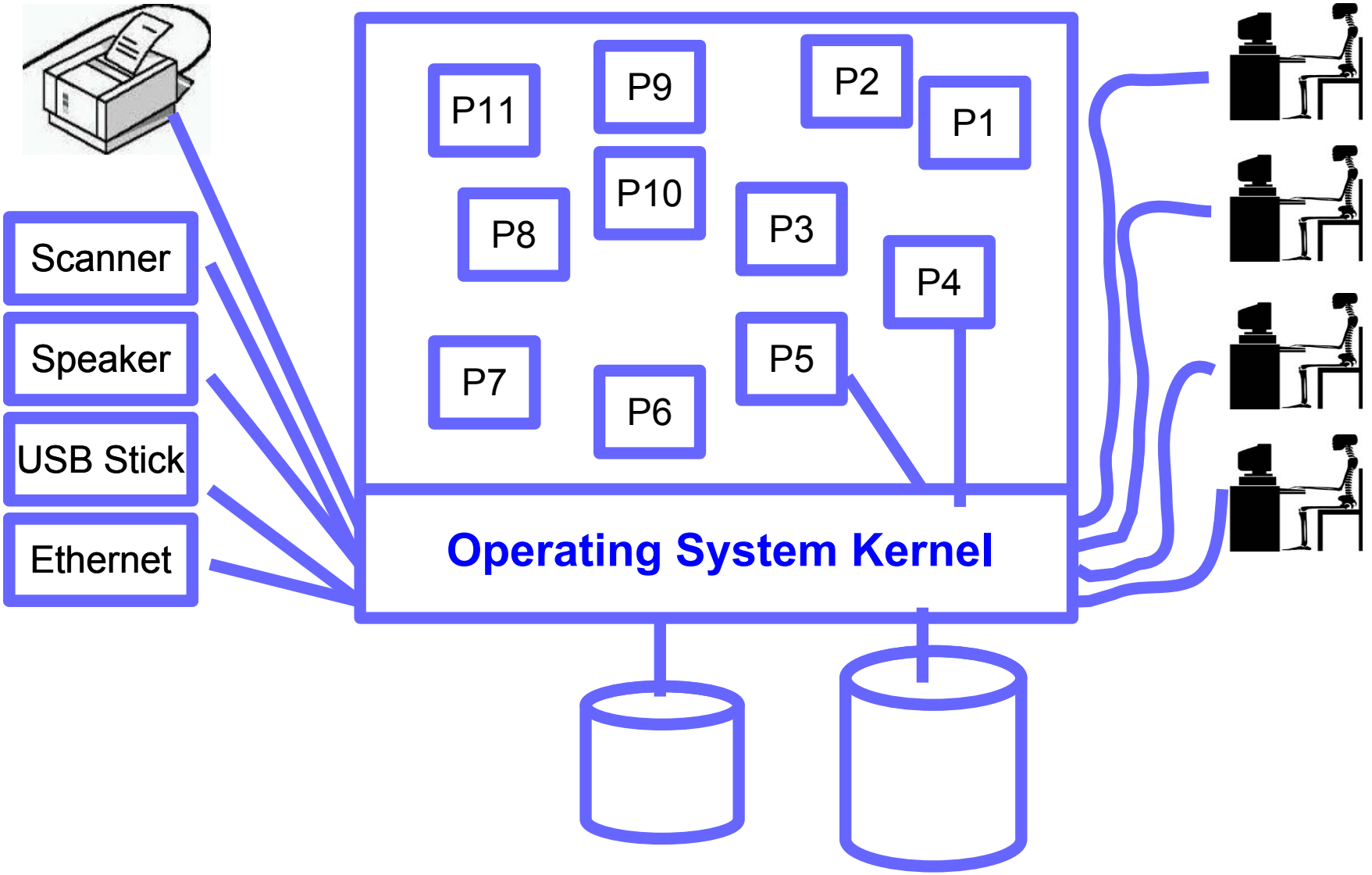
Ethernet





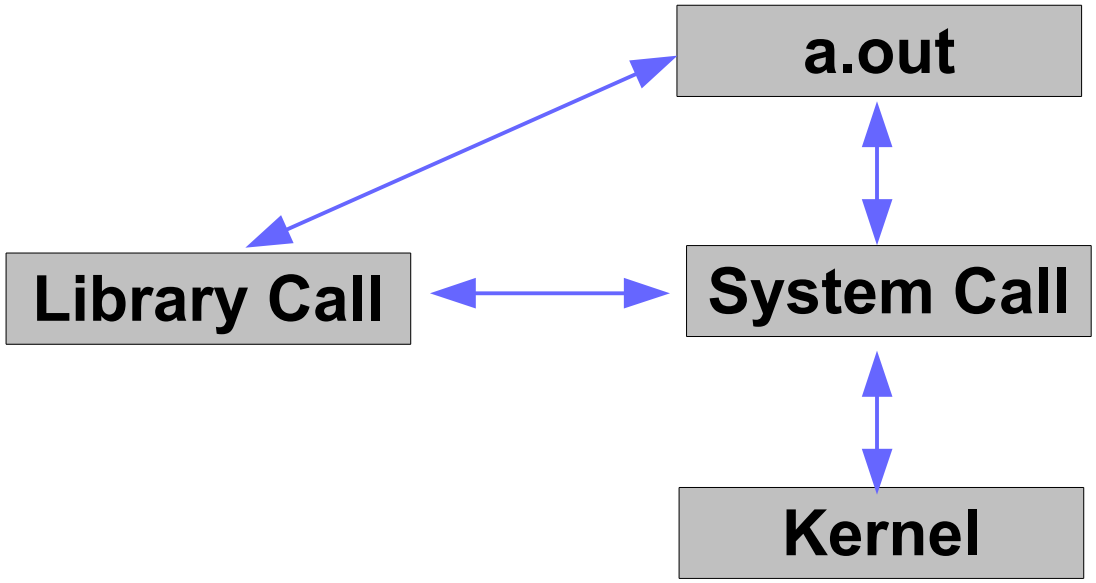
Role of Operating System is to manage all these resources and to connect various devices to the correct programs

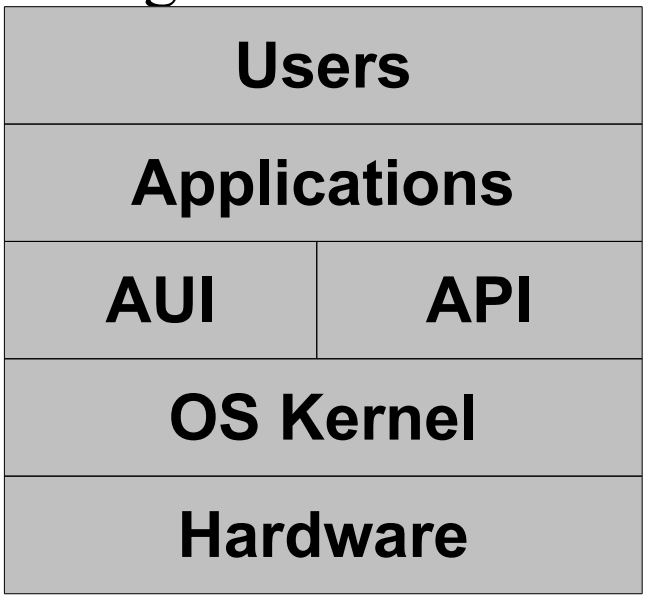
System Programmer Perspective



System Programming

- ***Application programming*** aims to produce software which provides services to the user, e.g., word processors, browsers, multi-media applications, etc.
- ***System Programming*** aims to produce software which provides services to computer hardware, e.g., disk defragmenter. The next step is writing kernel code to manage main memory, disk space management, cpu scheduling, and management of I/O devices through device drivers.





System Programmer Perspective

- A system programmer write programs that may have to acquire data
  - from a file(s) (residing on a local or remote disc) that may have been opened by some other user(s),
  - from other programs running on the same or different machine,
  - from the operating system itself.
- After processing, the programs may have to write results to a shared resource, which other processes are also writing, or results may need to be delivered to another process asynchronously, i.e., not when the process asked for it, but at some later unpredictable time. **The challenge is to master these concepts.**

Operating System Services

Some important tasks a kernel performs are:

- File Management
- Process management
- Memory management
- Information management
- Signal handling
- Synchronization
- Communication (IPC)
- Device management

Two methods by which a program can make requests for services from the Kernel:

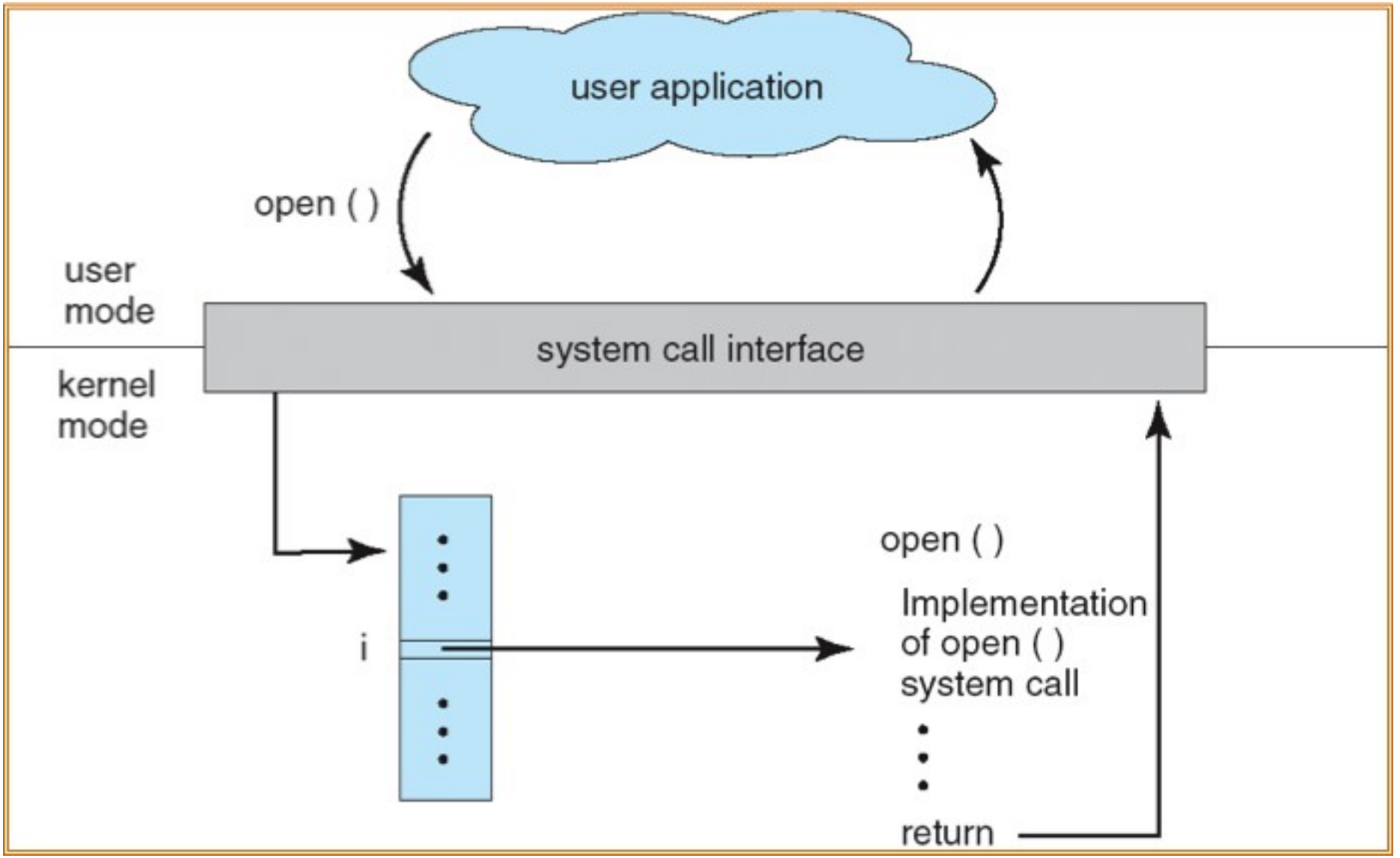
- By making a system call (entry point built directly into the kernel)
- By calling a library routine that makes use of this system call

System Calls

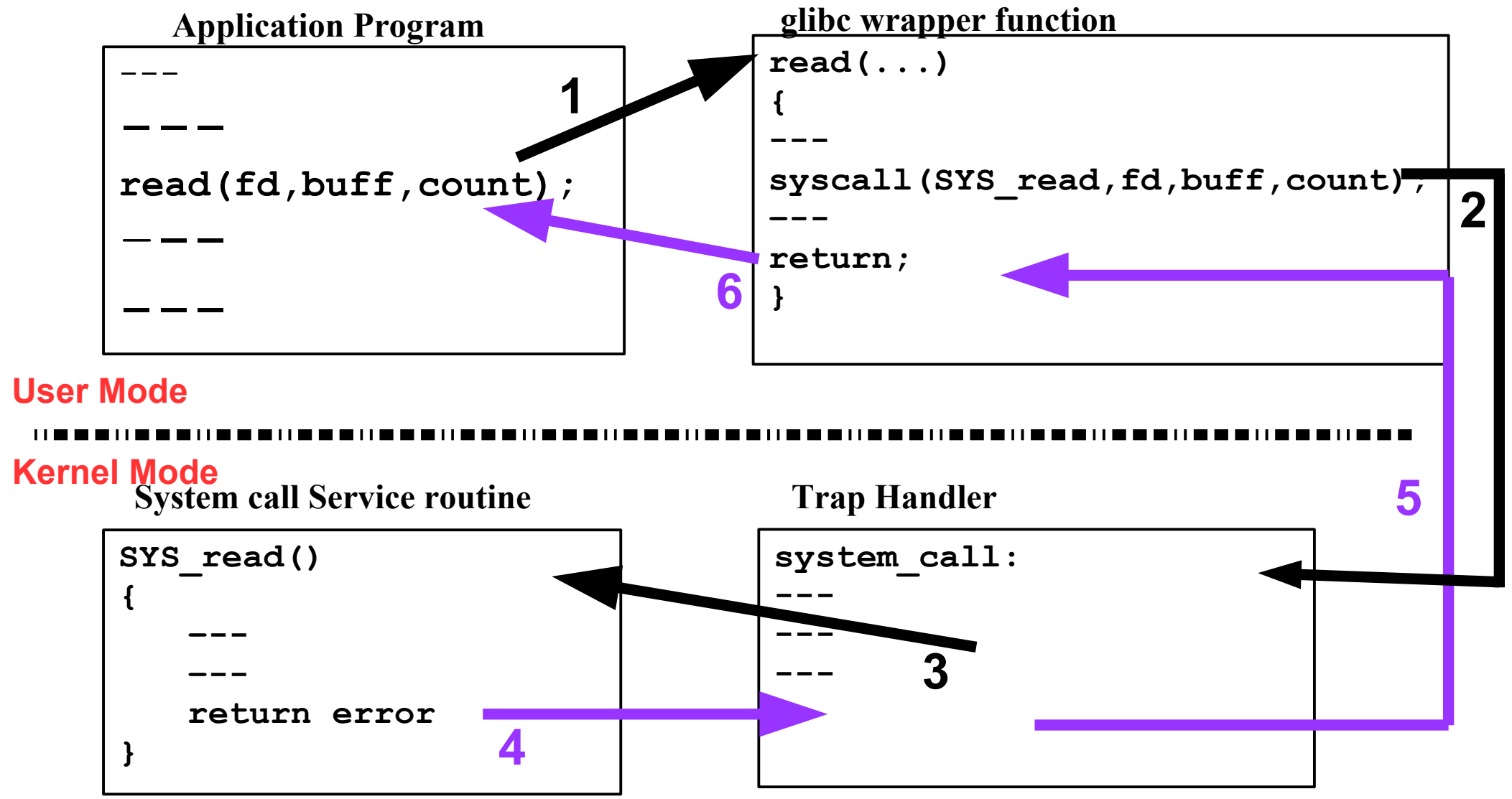
A system call is the *controlled* entry point into the kernel code, allowing a process to request the kernel to perform a privileged operation. Before going into the details of how a system call works, following points need to be understood:

- A system call changes the processor state from user mode to kernel mode, so that the CPU can access protected kernel memory
- The set of system calls is fixed. Each system call is identified by a unique number
- Each system call may have a set of arguments that specify information to be transferred from user space to kernel space and vice versa

System Call Invocation



System Call Invocation



What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 1**: Preparing your tool box
  - GNU `gcc` and `gdb`
  - NASM
  - UNIX `readelf, objdump, od, nm, ldd` and `ldconfig`
  - Creating your own Static and Dynamic libraries
  - UNIX `make, autoconf` and `cmake` utilities
  - Versioning Systems (`git`)
  - How a C program starts and terminates
  - Understanding Process Stack (Behind the curtain)
  - Understanding Process Heap (Behind the curtain)

What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 2:** File, Information and Time Management
  - UNIX universal I/O model and File System Architecture
  - Writing your own `ls`, `pwd`, `find`, `grep` and `more` utilities
  - Digging out information from system files
  - Writing `uname` and `who` utilities
  - Changing Terminal attributes
  - Modifying real time using Linux OS API

What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 3:** Process Management and Scheduling
  - Process identifications
  - Process trees, chains and fans
  - Zombies and orphan processes
  - Process groups, sessions and controlling terminals
  - UNIX daemons
  - System-VR3 and System-VR4 Scheduler
  - Linux O(1) and CFS Scheduler
  - Microsoft and Solaris Scheduler
  - Changing process scheduling parameters using Linux OS API

What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 4:** UNIX IPC Tools
  - Signals
  - PIPES/FIFOS
  - Message Queues
  - Shared Memory
  - Memory Mappings

What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 5:** Thread Management and Synchronization
  - POSIX Thread API
  - Synchronization among threads using mutex and condition variables
  - Synchronization among related and unrelated processes using semaphores

What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 6:** Network Programming
  - UNIX socket API
  - Writing concurrent servers using
    i. Multi-process Model
    ii. Multi-threaded Model
    iii. Non-Blocking I/O using `select()` system call
  - Managing your servers using `xinet` and `system` daemons

What we will do in this course?

In this course we shall learn to use the services provided by the kernel using **Linux OS API**, and this is what the course is all about

- **Module 7:** Network Security
  - Buffer overflow vulnerability
  - Exploiting it using Metasploit framework

Things to do!

If you have problems visit me in counseling hours