CS343: Operating System

OS Top-down Approach

Lect06 : 8th Aug 2023

Dr. A. Sahu

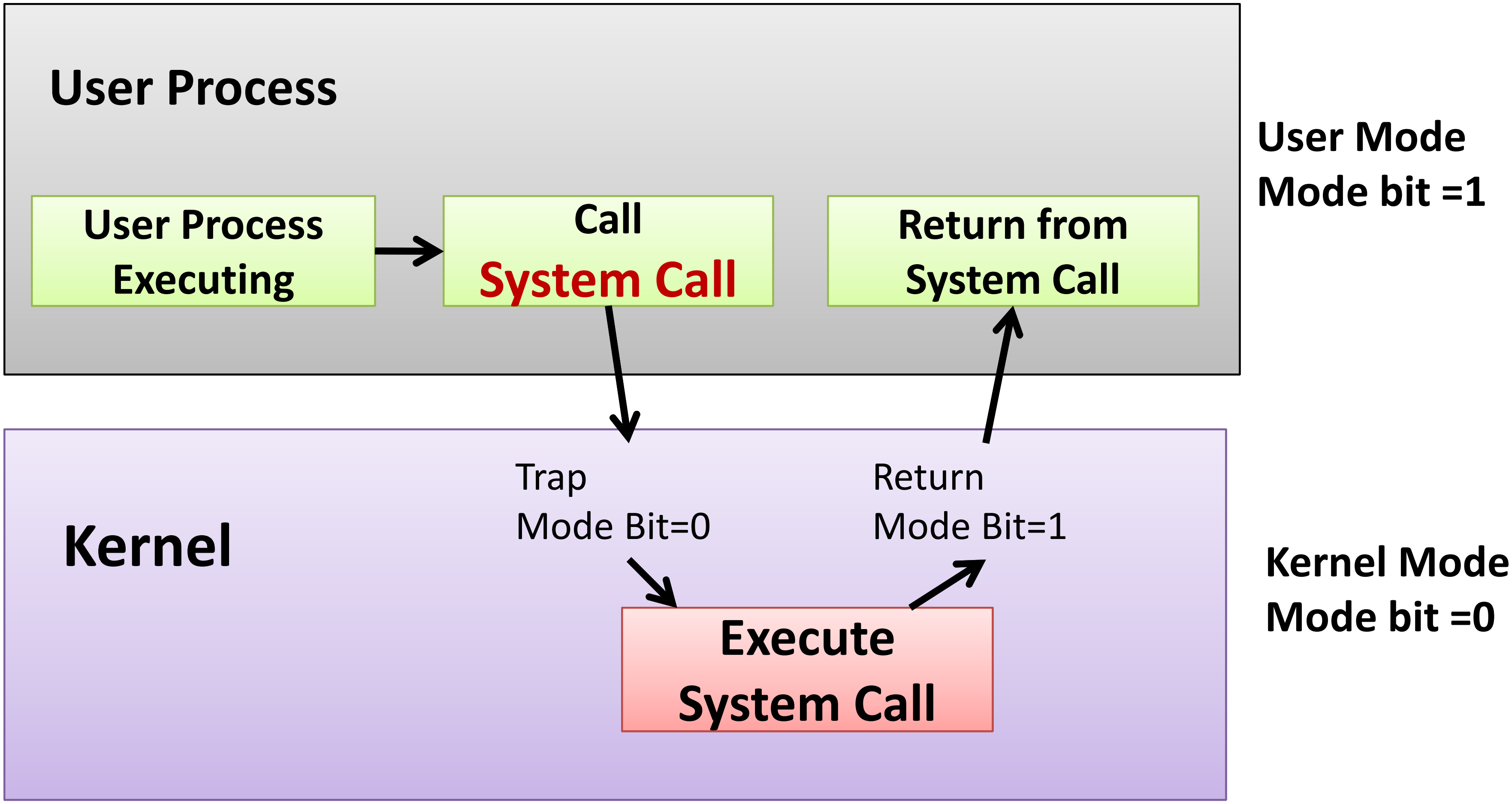
Dept of Comp. Sc. & Engg.

Indian Institute of Technology Guwahati

Outline

- OS Services: System call overview
  - Implementation
- OS : top down approaches

Transition from User to Kernel Mode



User Process

User Process Executing

Call System Call

Return from System Call

User Mode
Mode bit =1

Kernel

Trap
Mode Bit=0

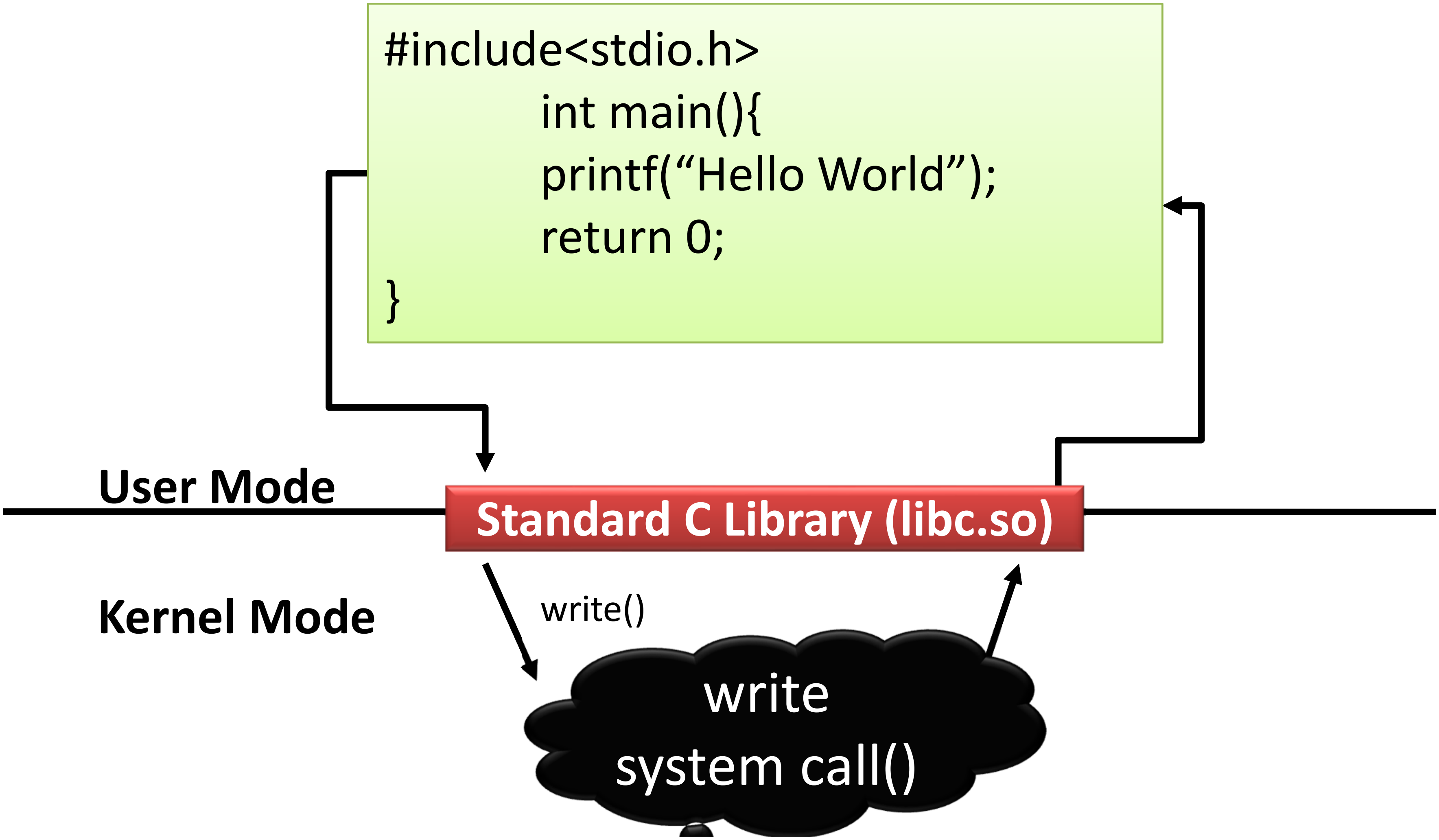
Return
Mode Bit=1

Execute System Call

Kernel Mode
Mode bit =0

Standard C Library Example

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



```
#include<stdio.h>
          int main(){
          printf("Hello World");
          return 0;
}
```

User Mode

Kernel Mode

Standard C Library (libc.so)

write()

write
system call()

Example of Standard API

- Standard API to read data from file or I/O

```
#include <unistd.h>
ssize_t   read(int  fd, void *buff, size_t count);
```

- I/O device abstracted as File

```
$ man 2  read                  // 2nd manual
```

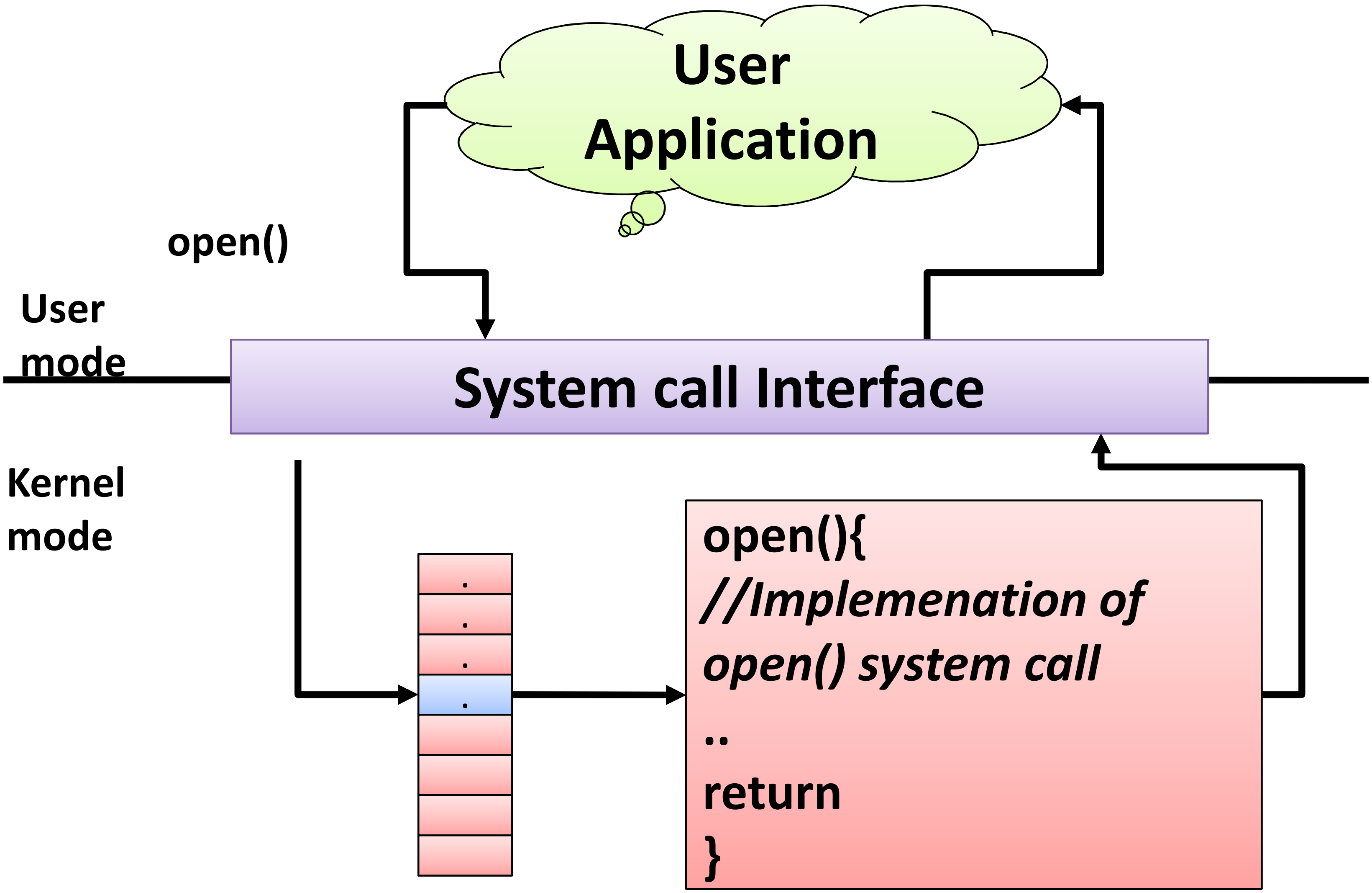
System Call Implementation

- Typically, a number associated with each system call
  - **System-call interface** maintains a table indexed according to these numbers
- The system call interface
  - invokes the intended system call in OS kernel
  - And returns status of the system call and any return values

System Call Implementation

- 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 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)

API–System Call–OS Relationship



User Application

open()

User mode

System call Interface

Kernel mode

```
open(){
//Implemenation of
open() system call
..
return
}
```

OS Management: Top Down Approach

- Process
- Memory
- Storage
- I/Os Subsystem
- Protection and Security

So user need **system call service of OS** for all above items

Process Management
&
Related System Call

Process Management

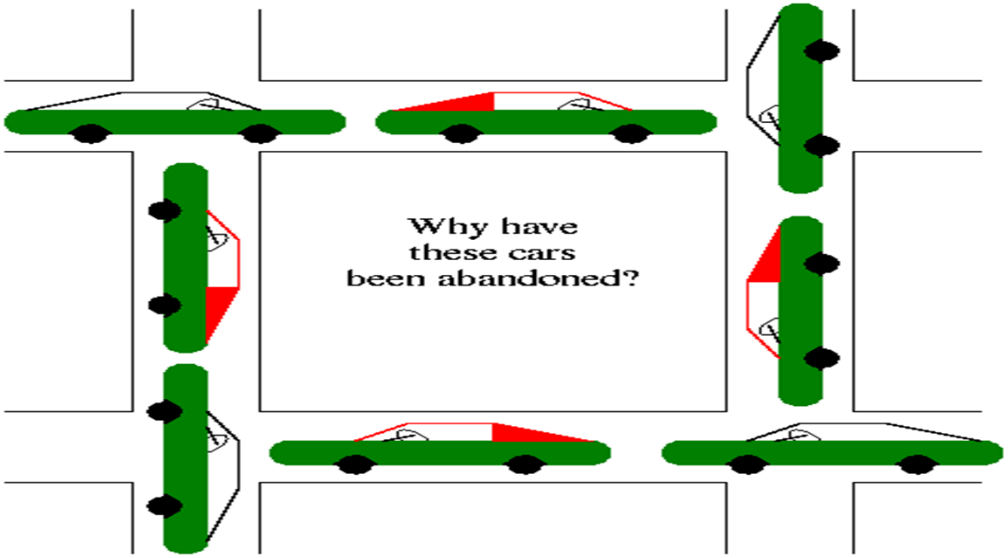
- A process is a program in execution
- **Process** is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources

Process Management Activities

- OS 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 **synchronization, communication, Deadlock** handling
  - Next Slide definition of **synch., comm & deadlock**

Synch., Comm. & DeadLock

- Synchronization
  - Two process trying access a shared object/variable .
  - Need to be access one at a time: Mutual Exclusion
  - Mutex, Lock, Monitor ===> Pthread API
- Communication : When many process collaborate and do a big work, Sending message through pipe/socket
- Deadlock
  - Circular Wait
  - No pmtn
  - Hold and wait
  - Mutual exclusive



Process Management

- Single-threaded process
  - has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion

Process Management

- 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 control: System Calls

- create process, terminate process
- end, abort, load, execute
- get process attributes, set process attributes
- wait for time, wait event, signal event
- allocate and free memory
- Dump memory if error
- **Debugger** for determining **bugs, single step** execution
- **Locks** for managing access to shared data between processes

Memory and I/O Management
&
Related System Call

Memory Management

- To execute a program
  - All (or part) of the instructions must be in memory
  - All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users

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 de-allocating memory space as needed

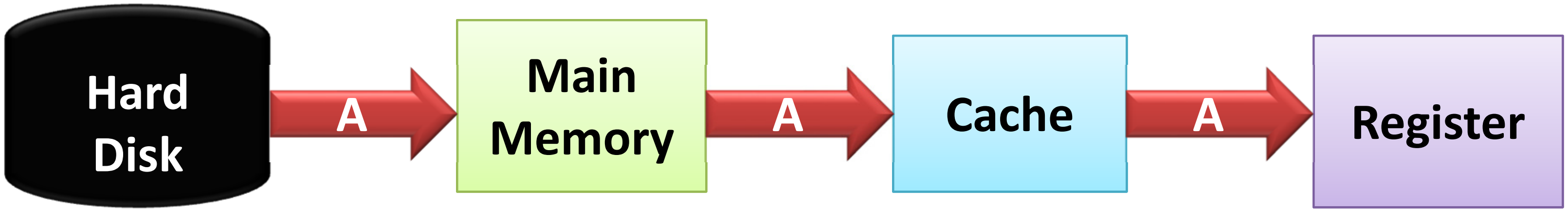
Mass-Storage Management

- 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 really
  - Hinges on disk subsystem and its algorithms

Mass-Storage Management

- OS activities
  - Free-space management, Storage allocation
  - Disk scheduling
- 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)

Migration of data "A" from Disk to Register



- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy
- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist

Storage Management

- 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)
- File System Management

File System Management

- Files usually organized into 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