CSE3241: Operating System and System Programming

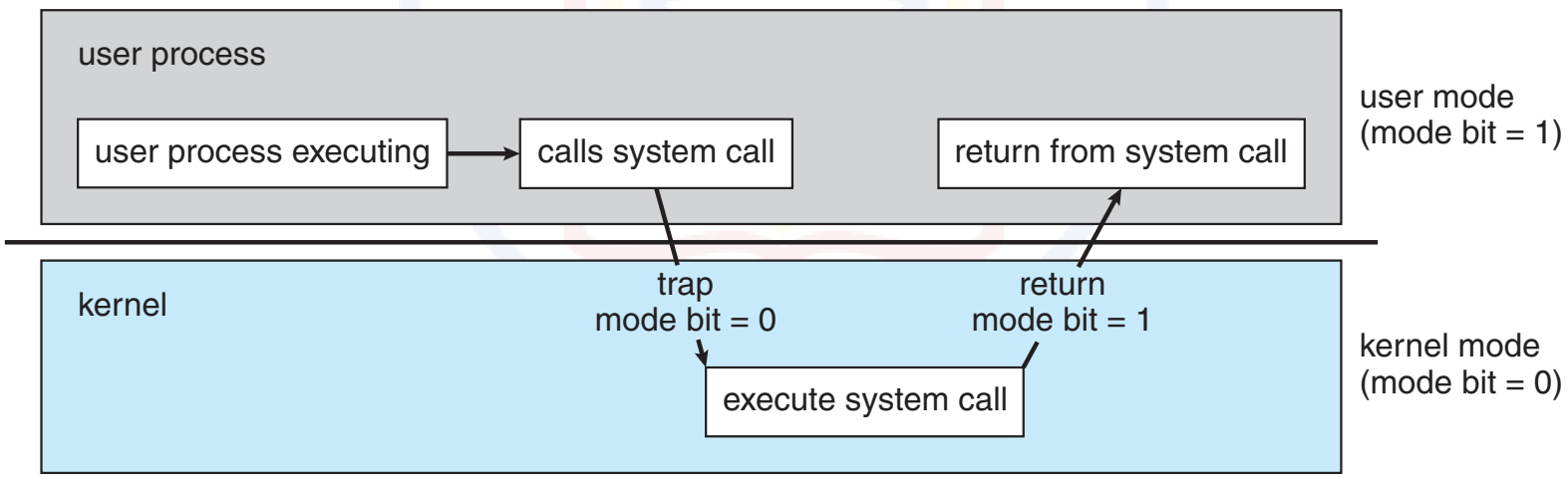
Class-9

Sangeeta Biswas, Ph.D.
Assistant Professor
Dept. of Computer Science and Engineering (CSE)
Faculty of Engineering
University of Rajshahi (RU)
Rajshahi-6205, Bangladesh
E-mail: sangeeta.cse@ru.ac.bd / sangeeta.cse.ru@gmail.com

November 7, 2017

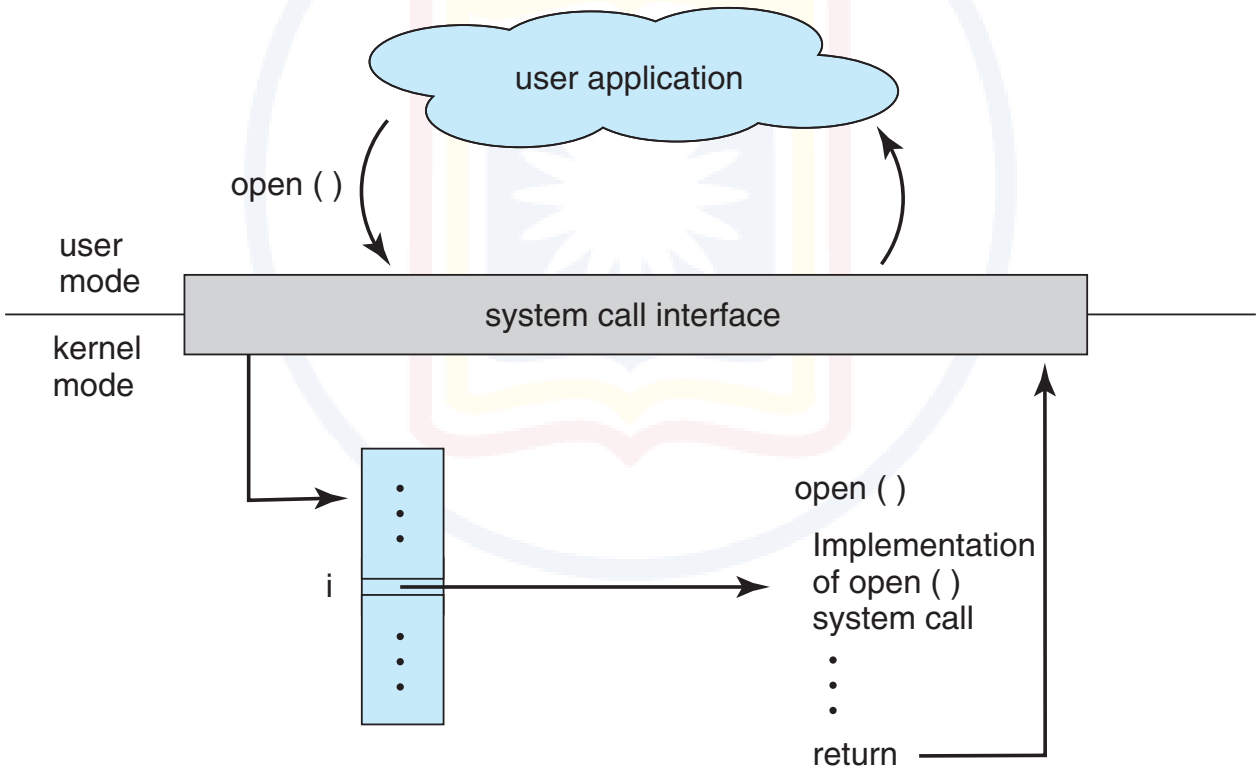
What is System Call?

- System Call is an instruction that:
  - provides an interface between restricted processes (e.g., user process) and unrestricted processes (e.g., kernel process).
  - generates a software interrupt to get services from the kernel process.
  - is also called Kernel call.

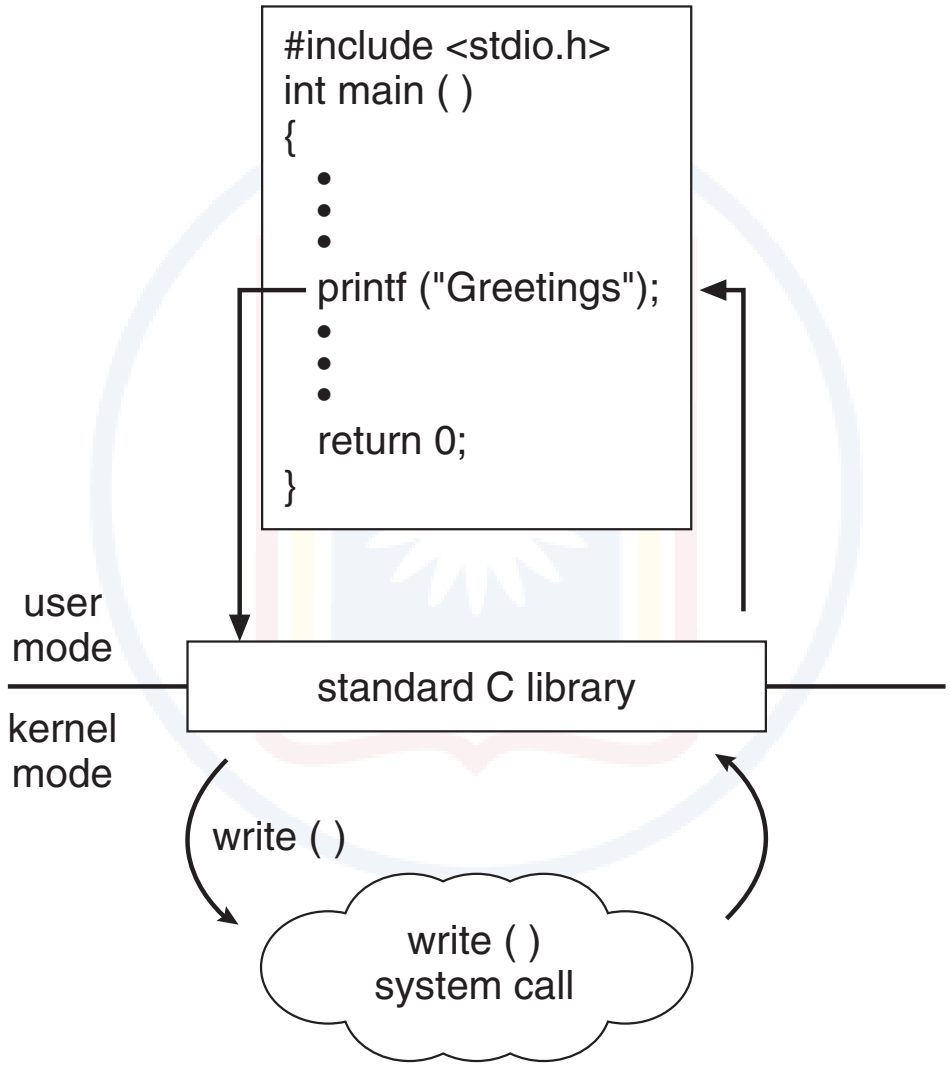


System Call Interface (SCI) [1]

- Most programming languages provide a System-Call Interface (SCI):
  - maintains a table indexed according to the numbers associated with system calls.
  - invokes the intended system call in the OS kernel.
  - returns the status of the system call and any return values.

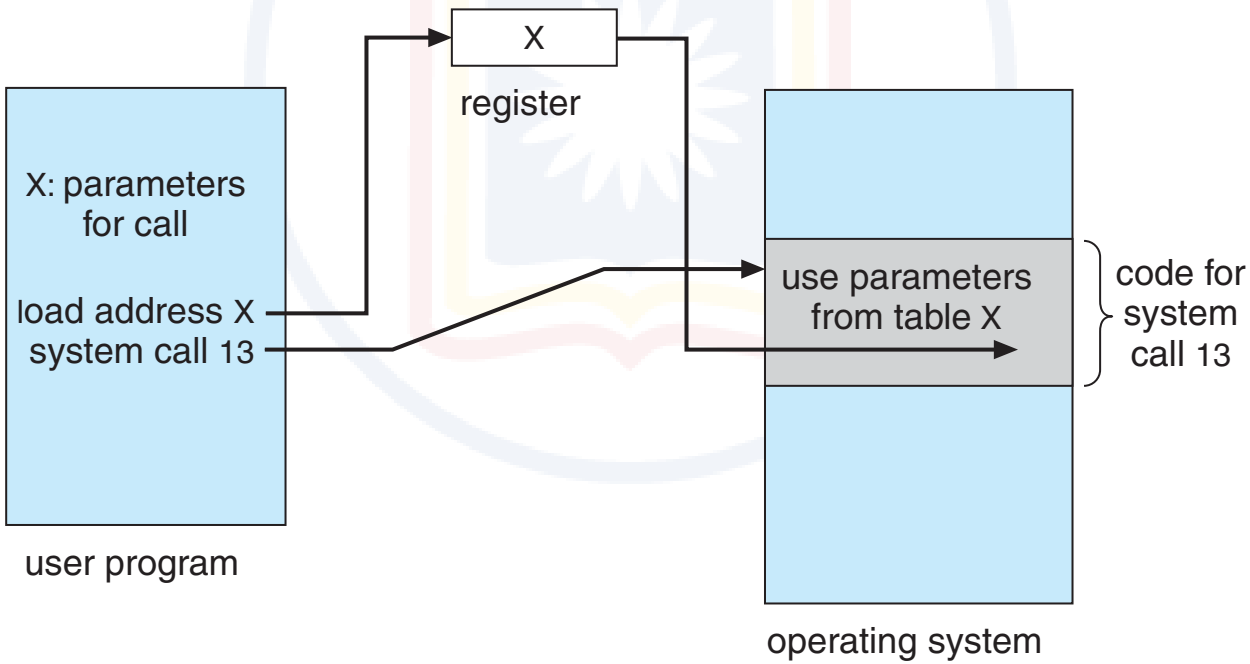


Standard C Library Handling of write()



Passing Parameters to Kernel Process [1]

- In a general way, argument(s) of a system call are:
  1. placed in registers (say rdi, rsi, rdx, rcx, r8, r9).
  2. pushed onto the stack by the SCI and popped off the stack by the kernel process.
  3. stored in a block, or table, in memory, and the address of the block is passed as a parameter in a register.



Types of System Call I

- OS provides certain services to user programs.
- System calls can be grouped into 6 categories:
  1. Process Managment
  2. File Managment
  3. Device Managment
  4. Information Maintenance
  5. Communications
  6. Protections

Types of System Call II

- Process Management
  - end, ebort
  - load, execute
  - create process, terminate process
  - get process attributes, set process attributes
  - wait for time
  - wait event, signal event
  - allocate and free memory
- Information Maintenance
  - get time or date, set time or date
  - get system data, set system data

Types of System Call III

- Device Management
  - request device, release device
  - read, write, reposition
  - get device attributes, set device attributes
  - logically attach or detach devices
- File Management
  - create file, delete file
  - open, close
  - read, write, reposition
  - get file attributes, set file attributes

Types of System Call IV

- Communications
  - create, delete communication connection
  - send, receive messages
  - transfer status information
  - attach, detach remote devices
- Protections
  - set permission, get permission
  - allow user, deny user

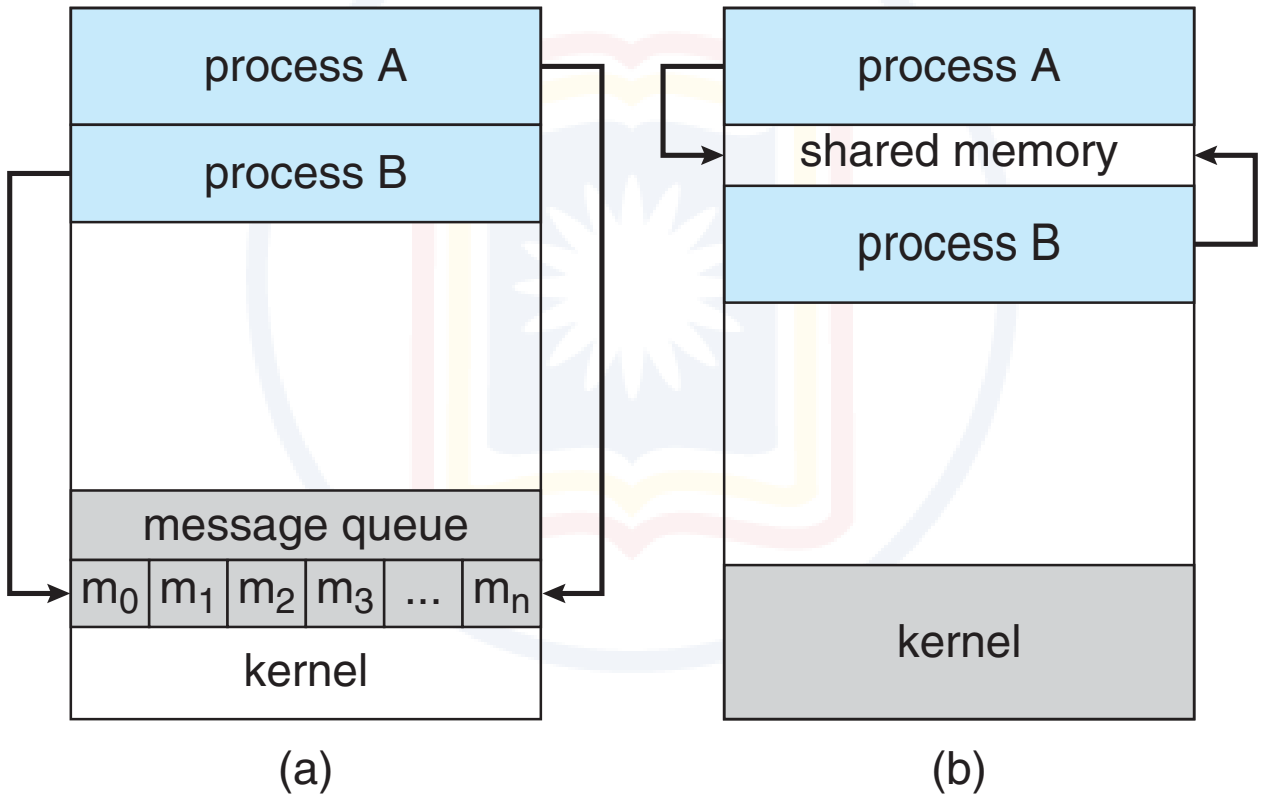
Inter-Process Communication (IPC)

- Processes can be categorized into the following 2 categories:
  1. **Independent Process**: it cannot affect or cannot be affected by other processes.
  2. **Cooperating Process**: it can affect or be affected by other processes.

- Cooperating processes require IPC. Two models for IPC:
  1. **Message Passing**:
     - a communication link is established between 2 processes.
     - easier to implement and useful for exchanging smaller amounts of data.
  2. **Shared Memory**:
     - a shared memory segment is created in the address space of a process which can be shared by other processes.
     - faster, since system calls are required only to establish shared memory regions.

Communication Models [1]

Figure: (a) Message Passing, (b) Shared Memory



System Calls for Process Creation

- Header Files: unistd.h, sys/wait.h and stdlib.h
- System Calls:
  1. **fork()**: for creating a child process.

     childPID = fork()
  2. **shmat()**: for getting PID of the current process.

     myPID = getpid()
  3. **wait()**: for waiting for the termination of child process.

     deadChildPID = wait(NULL)
  4. **exit()**: for terminating a normal process.

     exit(0)
  5. **execlp()**: for replacing the process's memory with a new program.

     execlp(exeFile, arg0, arg1,...)

System Calls for Shared Memory Model

- Header Files: sys/shm.h and sys/stat.h
- System Calls:
  1. **shmget()**: for allocating a shared memory segment into the address space of a process.

     shrSegID = shmget(IPC_PRIVATE, size, S_IRUSR | S_IWUSR)
  2. **shmat()**: for attaching the shared memory segment with a process.

     shrSegMem = (char *) shmat(shrSegID, NULL, 0)
  3. **shmdt()**: for detaching the shared memory segment with a process.

     shmdt(shrSegMem)
  4. **shmctl()**: for removing the shared memory segment from a process.

     shmctl(shrSegID, IPC_RMID, NULL)

References


P. B. Galvin A. Silbeschatz and G. Gagne.
Operating System Concepts.
John Wiley & Sons, 9 edition, 2012.