CS341: Operating System

System Call, System Program and Type of System

Lect06 : 13th Aug 2014

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

1

Outline

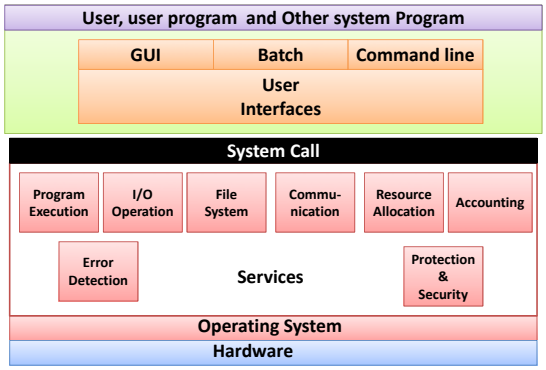
- System Call
- System Program
- Linker/Loader with Examples
- Types of computing Environment

2

Operating System Services

- OS provide
  - An environment for execution of programs
  - And services to programs and users
- Services
  - One set of OS services provides functions that are ***helpful to the user***
  - Another set of OS functions exists for ***ensuring the efficient operation of the system*** itself via resource sharing

A View of OS Services

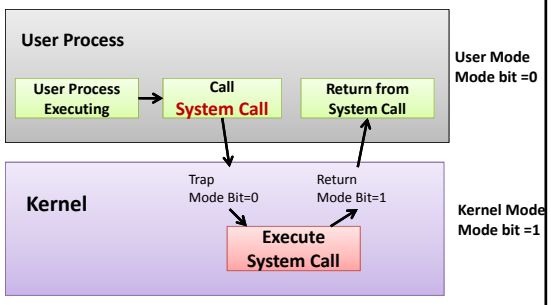


System Calls

- Programming interface to the services provided by the OS
  - Typically written in a high-level language (C or C++)
- Mostly accessed by programs
  - Via a high-level Application Programming 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),
  - **Java API** for the Java virtual machine (JVM)

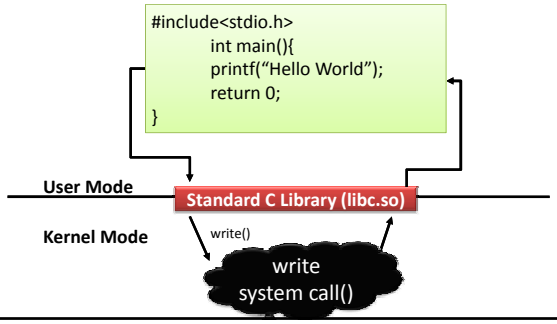
Note that the system-call names used throughout this course are generic

Transition from User to Kernel Mode



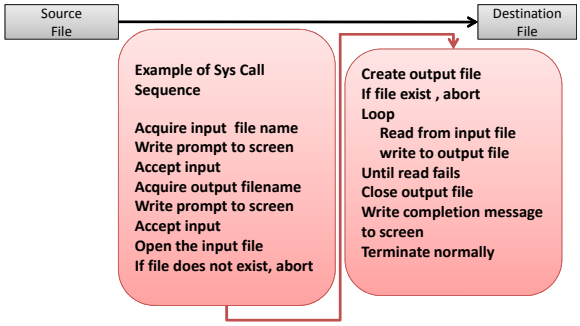
Standard C Library Example

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



Example of System Calls

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



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***

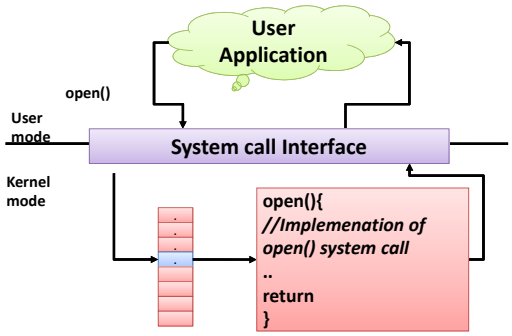
System Call Implementation

- 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



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

13

Process Management & Related System Call

14

Process Management

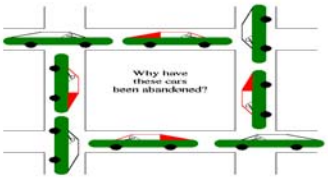
- A process is a program in execution
- **Process** is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.
- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of 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**
  - **No progress**
  - **Circular Wait**



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

21

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 deallocating 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

I/O Subsystem

- **One purpose of OS is to hide peculiarities of hardware devices from the user**
- I/O subsystem responsible for
  - Memory management of I/O
    - 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

File & Device : System Calls

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

Communication: System Calls

- **Communications**
  - create, delete communication connection
  - send, receive messages if **message passing model** to **host name** or **process name:** From **client** to **server**
  - **Shared-memory model** create and gain access to memory regions
  - transfer status information
  - attach and detach remote devices

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
  - **Privilege escalation** allows user to change to effective ID with more rights

Protection and Security

- 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 effective ID with more rights **Ex: sudo**

Protection & Info. Maintenance: System Calls

- **Information maintenance**
  - get time or date, set time or date
  - get system data, set system data
  - get and set process, file, or device attributes
- **Protection**
  - Control access to resources
  - Get and set permissions
  - Allow and deny user access

Windows and Unix System Calls

	Window	Linux
Process Control	CreatePrtocess()	fork()
	Exit Process()	exit()
	WaitForSingleObject()	wrait()
File Manipulation	CreateFile()	open()
	ReadFile	read()
	WriteFile()	write()
	CloseHandle()	close()
Device Manipulation	SetControlMode()	ioctl()
	ReadConsole()	read()
	WriteConsole()	write()

Windows and Unix System Calls

	Window	Linux
Information Maintenance	GetCurrentProcessID()	getpid()
	SetTimer()	alarm()
	Sleep()	sleep()
Commu-nication	CreatePipe()	pipe();
	CreateFileMapping()	shmget()
	MapViewOfFile()	mmap()
Protection	SetFileSecurity()	chmod()
	InitializeSecurityDescriptor()	umask()
	SetSecurityDescriptor Group()	chown()