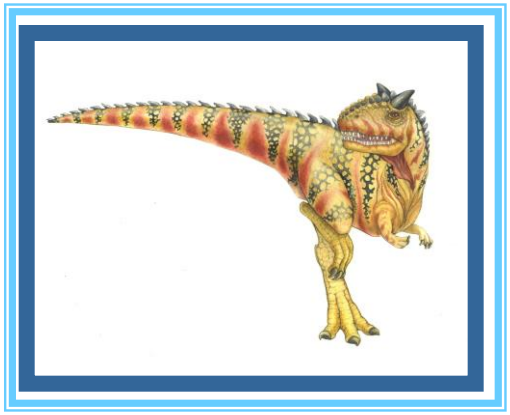
Chapter 1: Introduction

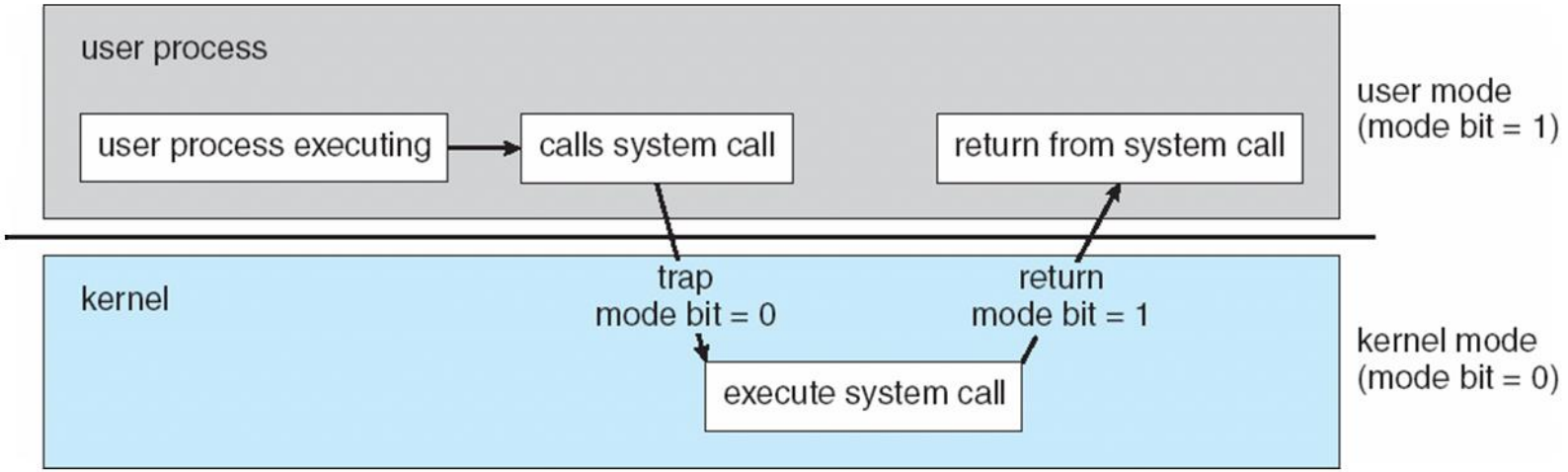


Operating-System Operations

- Interrupt driven by hardware
- Software error or system request creates **exception** or **trap**
  - Division by zero, memory error
  - request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Set interrupt after specific period
  - Operating system decrements counter
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



Transition from User to Kernel Mode

hello32.asm

```
1   ;  hello32.asm shows system calls using old 32-bit assembly
2   ;  The program is compiled for a 64-bit system
3   ;
4   ; assemble:     nasm -f elf64 -l hello32.lst  hello32.asm
5   ; link:         gcc -o hello32  hello32.o
6   ; run:          hello
7   ; output is:    Hello World
8
9       SECTION .data       ; data section
10  msg:    db "Hello World",10 ; the string to print, 10=cr
11  len:    equ $-msg       ; "$" means "here"
12                          ; len is a value, not an address
13
14      SECTION .text       ; code section
15          global main     ; make label available to linker
16  main:                   ; standard  gcc  entry point
17
18      mov edx,len     ; arg3, length of string to print
19      mov ecx,msg     ; arg2, pointer to string
20      mov ebx,1       ; arg1, where to write, screen
21      mov eax,4       ; write sysout command to int 80 hex
22      int 0x80        ; interrupt 80 hex, call kernel
23
24      mov ebx,0       ; exit code, 0=normal
25      mov eax,1       ; exit command to kernel
26      int 0x80        ; interrupt 80 hex, call kernel (trap, system call)
```

Transition from User to Kernel Mode

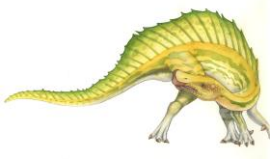
hello32.asm | hello64.asm

```
1  ;  hello64.asm shows system calls using old 32-bit assembly
2  ;  The program is compiled for a 64-bit system
3  ;
4  ; assemble:     nasm -f elf64 -l hello64.lst  hello64.asm
5  ; link:         gcc -o hello64  hello64.o
6  ; run:          hello
7  ; output is:    Hello World
8
9  global main            ; global entry point export for ld
10
11 section .text
12 main:
13
14     ; sys_write (stdout, message, length)
15
16     mov    rax, 1          ; sys_write
17     mov    rdi, 1          ; stdout
18     mov    rsi, message    ; message address
19     mov    rdx, length     ; message string length
20     syscall
21
22     ; sys_exit (return_code)
23
24     mov    rax, 60         ; sys_exit
25     mov    rdi, 0          ; return 0 (success)
26     syscall
27
28 section .data
29     message: db 'Hello World',0x0A ; message and newline
30     length:  equ    $-message       ; NASM definition pseudo-instruction
31
```

Process Management

- A process is a program in execution. It 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
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread

1. Give an example of a multi-threaded process
2. Draw a picture of process and threads

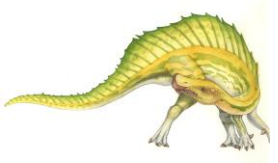


Process Management Activities

The operating system 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
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

1. How can we see active processes on windows?
2. What about Linux?



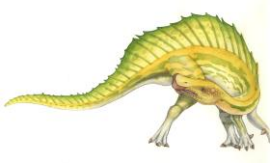
Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
  - Optimizing CPU utilization and computer response to users
- 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

1. Who decides to move data in/out of memory?
2. If the data is not in memory, where is it?
3. How is the decision made?

Storage Management

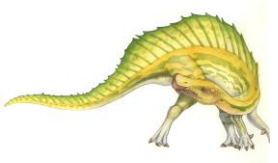
- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Files are mapped onto physical medium
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what



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 hinges on disk subsystem and its algorithms
- 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
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

1. What is tertiary storage?



Performance of Various Levels of Storage

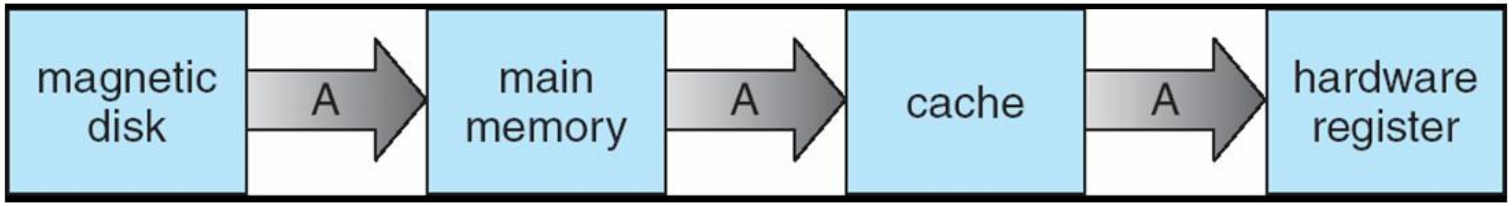
- Movement between levels of storage hierarchy can be
  - explicit - data transfer from cache to registers (usually no OS intervention)
  - implicit - data transfer from disk to memory (usually involves OS)

1. What is .5ns?

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Migration of Integer A from Disk to Register

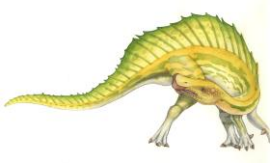
- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- 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
  - Various solutions covered in Chapter 17

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 including 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



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
- 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
- Skipping 1.11 & 1.12 for now

End of Chapter 1