Real Operating Systems

Lecture #12

Department of Computer Science and Technology
University of Bedfordshire

Written by David Goodwin,
based on the lecture series of Dayou Li
and the book *Understanding Operating Systems 4thed.*
by *I.M.Flynn and A.McIver McHoes* (2006).

Operating Systems, 2012

Outline

Introduction to different O/Ss

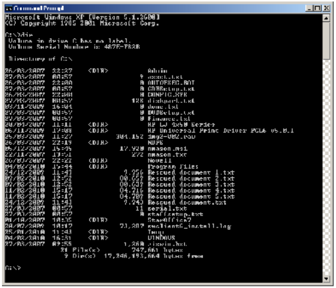
- Three typical operating systems
  - Disk operating system (DOS)
  - Windows
  - Unix/Linux

DOS

- History
  - In 1980, IBM looked for an operating system for its soon-to-be-released 6-bit personal computers
  - Digital Research offered CP/M-86
  - Softech offered P-System
  - MS
    - MS also looked for OS for its 16-bit computers
    - Seattle Computer Products offered 86-DOS
    - MS bought it and renamed it MS-DOS
  - IBM chose MS-DOS in 1981 and called it PC-DOS
  - MS-DOS evolved from v 1.0 to v 6.22 from 1981 to 1994

DOS

- Features
  - Single user, stand-alone desktop
  - Command-line
  - Commands are based on words
  - Examples
    - COPY – copy a file
    - DEL – delete a file
    - PRINT – print files on a printer
    - DIR – list files in this directory
    - MD – make a new directory
    - CHKDSK – check the disk
    - COMP – compare two files



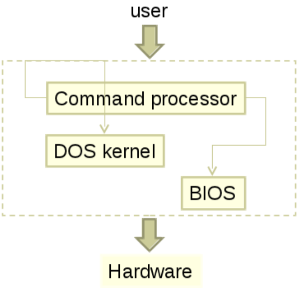
Layers of DOS

- Three layers:
  - Top layer – command processor
    - Sends prompts to user
    - Accepts commands
    - Executes commands (including interpret commands to machine language)
    - Issues responses

- Machine language
  - Only language a bare computer can understand
  - Program to hardware level such as registers, memory and I/O devices
  - Written by 0s and 1s
- Assembly language
  - Also program to hardware level
  - Instructions are English words based
- Advanced programming languages
  - Interpreting
  - Procedural

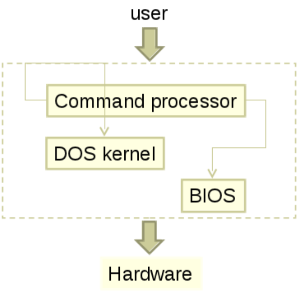
Layers of DOS

- - Middle layer – DOS kernel
    - A program containing routines that are needed for interfacing disk
    - Stored in MSDOS.SYS file
    - Read to memory during initialisation time



Layers of DOS

-
  - Bottom layer – BIOS (Basic Input/Output System)
    - Interfaces I/O devices such as printer, monitor and keyboard
    - Controls data flow to and from these devices
    - Receive statues information about these devices



WINDOWS

- History
  - Initiative
    - To allow users to not have to remember and use system commands but via a user-friendly interface – GUI
    - Not a replacement of DOS
  - Early versions (1985 to 1992)
    - Windows 1.0 to 3.1 are only "interfaces" between GUI and DOS
    - Single-user rather than networked
  - True O/S since 1992
    - Windows 95 is first true O/S
  - Network O/S
    - Windows NT version 3.1 in 1993 led by David Cutler

WINDOWS - Design goals

- Extensibility – be easily enhanced to meet changes over time to support new hardware and software technologies
- Privileged process and non-privileged processes
  - Kernel mode refers to the privileged mode of a processor
    - All instructions are allowed
    - System memory is accessible
  - Use mode refers to the non-privileged mode of a processor
    - Only certain instructions are allowed
    - System memory is not accessible
  - O/S executes in kernel mode
  - Application programs (protected subsystems) run in user mode
- Modular structure
- Drivers for new file systems, devices and networks
- Objects – abstract data types

WINDOWS - Design goals

- Portability – ability for O/S to operate on different machines that use different processors or configurations
  - Code is modular
  - Standard high-level programming language c/c++ is used for implementation
  - Hardware abstraction layer (HAL) providing isolation from hardware dependencies
    - HAL abstracts hardware such as caches with a layer of low-level software so that higher-level code needs no change when moving from one platform to another

WINDOWS - Design goals

- Reliability – predictability in responding to error conditions, including hardware failures
  - Modular design
  - NTFS to recover all types of errors
  - US government-certifiable security
  - Virtual memory strategy to prevent one user from reading or modifying memory that is occupied by another user
- Compatibility – ability of an O/S to execute programs written for other O/Ss
  - Execution environments for applications that differ from Win32 API
  - Source-level compatibility to POSIX (Portable Operating System Interface for Computer Environment)
  - Supporting already-existing file systems

WINDOWS - Design goals

- Performance – fast response times
  - Crucial processes such as system calls and page faults are tested and optimised
  - LPC (local procedure call) to guarantee fast communications among the protected subsystems
  - Carefully designing the environment subsystems to ensure the speed of frequently used systems services
  - Critical elements of Windows' networking software are built in the privileged protion

UNIX AND LINUX

- Advantages shared by Unix and Linux
  - Portable
  - Powerful utilities
  - Device independent

UNIX AND LINUX

- Evolution of UNIX

Year	Release	Features
1971	UNIX V1	Based on MULTICS; introduced shell concept, written in assembly language
1972	UNIX V2	Added pipes and filters
1973	UNIX V3	Kernel and I/O, first version written in C
1973	UNIX V4	
1975	UNIX V6	First version to become commercially available
1979	UNIX V7	More powerful shell added: string variables, structured programming, trap handling
1980	UNIX System III	First version used in 16-bit microcomputers
1981	UNIX System IV	First version available for a mainframe
1983	UNIX System V Release 1	Added small general-purpose programs
1984	UNIX System V Release 2	Added features from Berkeley version: shared memory, more commands, vi editor, termcap database, flex filenames
1988		The Open Group founded
1991	UNIX System V Release 4	Combined features from BSD, SunOS, and Xenix
1991	Solaris 1.0	Sun's version designed to run on Sun workstations, derived from AT&T's UNIX System V, Release 4
1993	Novell UnixWare	Novell's version of System V release 4 designed to work with NetWare
1994	Single UNIX Specification 1	Separates the UNIX trademark from actual code stream, opening the door to standardization
1997	Single UNIX Specification 2	Adds support for realtime processing, threads, and 64-bit processors
2001	Single UNIX Specification 3	Unites IEEE POSIX and industry efforts to standardize
2003	ISO/IEC 9945:2003	International standard approved for the core volumes of Single UNIX Specification Version 3

UNIX AND LINUX

- Evolution of LINUX

Year	Release	Features
1994	Beta versions	First Red Hat Linux product available to the public in a series of beta versions.
1994	RHL 1.0	First non-beta release of Red Hat Linux.
1995	RHL 2.0	Written in Perl for quick development.
1996	RHL 3.0.3	The first approximately concurrent multi-architecture release; supported the Digital Alpha platform.
1996	RHL 4.0	Based on the 2.0.18 kernel and the first release to include documentation freely available in electronic form.
1997	RHL 5.0	Named 1997 InfoWorld Product of the Year.
1999	RHL 6.0	Integrated GNOME desktop GUI.
2000	RHL 7.0	First release that supported Red Hat Network out of the box.
2001	RHL 7.1	Introduced the 2.4 kernel.
2002	RHEL 2.1 AS (Advanced Server)	Launch of Red Hat Enterprise Linux, the first commercial enterprise computing offering. Based on RHL 7.2.
2002	RHL 8.0	Designed to provide a unified look across RHL and RHEL desktops.
2003	RHL 9	First release to include Native POSIX Thread Library (NPTL) support.
2003	RHEL 3	The first Red Hat product made to run on seven chip architectures (by Intel, AMD, and IBM).
2003	Fedora Core 1	Product based on RHL 9 for individual users; created by the Fedora Project in cooperation with Red Hat.
2004	Fedora Core 2	Introduced Security Enhanced Linux (SELinux) an implementation of Mandatory Access Control (MAC) in the kernel.
2004	Fedora Core 3	Supported the 2.6.9 Linux kernel, updated SELinux, and support for the latest popular GUIs including KDE and Gnome.

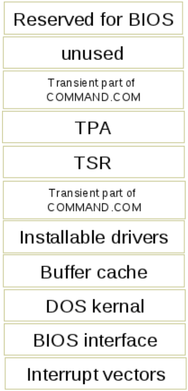
UNIX AND LINUX - DESIGN GOALS

- Supporting software development
- Keeping its algorithms as simple as possible

Function	Purpose
Multiple processes and multiple processors	Linux can run more than one program at a time using one or several processors.
Multiple platforms	Although it was originally developed to run on Intel's processors for microcomputers, it can now operate on many other platforms
Multiple users	Like all UNIX systems, Linux allows several users to work on the same machine at the same time.
Interprocess communications	Linux supports pipes, IPC, sockets, etc.
Terminal management	Terminal management conforms to POSIX standards and it also supports pseudo-terminals as well as process control systems.
Peripheral devices	Supports a wide range of devices including sound cards, graphics interfaces, networks, SCSI, USB, etc.
Buffer cache	Linux supports a memory area reserved to buffer the input and output from different processes.
Demand paging memory management	Linux loads pages into memory only when they're needed.
Dynamic and shared libraries	Dynamic libraries are loaded only when they're needed and their code is shared if several applications are using them.
Disk partitions	Linux allows disk partitions used by file systems such as Ext2 and partitions having other formats (MS-DOS, ISO 9660, etc.).
Network protocol	Supports TCP/IP and other network protocols

DOS MEMORY MANAGEMENT

- ROM and RAM
  - ROM contains a section of BIOS for starting up a computer
  - RAM is mail memory where programs are loaded
    - Interrupt vectors
    - BIOS interface
    - DOS kernel
    - Buffer cache
    - Installable drives
    - Resident part of COMMAND.COM
    - TSR
    - User memory
    - Transient part of COMMAND.COM
    - Reserved for BIOS



DOS MEMORY MANAGEMENT

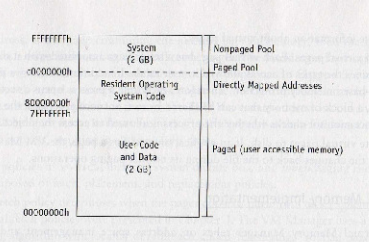
- TPA allocation “policy”
  - Reason for allocating memory blocks in TPA for more than one programs
    - Improving efficiency when executing or accessing the next program/file after executing one program
  - “Policy”
    - Dynamic allocation
    - Modification – modifying (normally giving more) memory blocks to a running program when it requires more for, e.g., I/O purposes
    - Release of main memory– after part of a program is executed
    - For .EXE files – allocating the max memory needed if TPA has enough free memory, otherwise, giving the min memory
    - For .COM files – allocating all memory it may need despite it will use or not

DOS MEMORY MANAGEMENT

- TPA allocation algorithms
  - First-fit was used in early version and Best-fit was used in later version
  - A block can be as small as 16 bytes (also known as paragraph) and as large as the max available memory (TPA)
  - The first five bytes have special usage:
    - Byte 0 – indicator of the last block (90h if yes, 77h otherwise)
    - Byte 1– indicator of status of the block (00h for busy)
    - Byte 2 – pointer to PSP
    - Bytes 3 and 4 – indicator of the number of paragraph contained in the block
  - List of busy/free blocks
    - List is a data structure containing a head and a tail
    - An algorithm searches for a free block in the list for a file

WINDOWS MEMORY MANAGEMENT

- Memory manage challenge and solution
  - Challenge is to run programs written for Windows, DOS and POSIX without clashing each other in memory
  - Solution is to separate system memory and application memory
  - Example
    - 4GB memory with 2GB each allocated for application storage and system storage



WINDOWS MEMORY MANAGEMENT

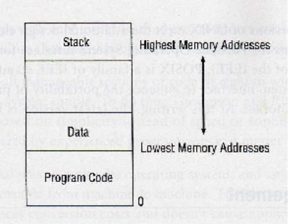
- Virtual memory manager
  - To allow applications to share memory
  - Allocate memory in two stages:
    - Reserving memory
    - Committing memory
  - Read/write protection for virtual memory so read/write performed by one process won't be interrupted by other processes
  - Lock virtual memory pages in physical memory to ensure that a critical page won't be removed from memory while a process is using it
  - Retrieve information
  - Protect virtual pages
  - Rewrite virtual pages to disk

Windows Memory Management

- Implementation
  - Address space management
    - System storage section of the virtual memory can only be accessed by kernel-mode processes
    - Addresses of the lower part of this section are translated by hardware to have a fast access speed
  - Paging
    - Fetch policy determines time when copying a page from memory to disk
    - Placement policy is a set of rules determining where the vurtual pages are loaded in memory
    - Replacement policy determines which virtual page must be removed from memory to make room for new pages

LINUX MEMORY MANAGEMENT

- Swapping and demand paging
  - Swapping a job out of memory
    - Round robin policy is used – jobs/processes are managed by round robin and if a job's time slice is up or when it generate an I/O interrupt, the entire job will be swapped out to secondary storage to make room for another job that is waiting in the READY queue
  - Demand paging
    - Image
    - Program code
    - Data
    - stack



LINUX MEMORY MANAGEMENT

- UNIX kernel
  - Responding system calls issued by processes
  - Set up memory boundary
  - Permanently resides in memory
  - Uses the least recently used (LRU) page replacement algorithm

DOS PROCESS MANAGEMENT

- Designed for single-task and single user environment
  - Parent child processes – parent process calls child process and then goes to sleep and remains asleep while the child process is running
  - One process runs at a time
  - The child process can interrupt the parent process
  - 256 interrupts

DOS PROCESS MANAGEMENT

- Reason for interrupts
  - No need for having any sophisticated process management as MS-DOS is designed for single user in a single task environment
  - A task/process sometimes does need to be interrupted , for example, when it waits for a peripheral device, to improve efficiency
  - Process life cycle: ready – running – waiting – exit
  - Synchronisation among tasks/processes is need
  - Synchronisation is achieved in MS-DOS via interrupts

DOS PROCESS MANAGEMENT

- Different types of interrupts
  - Internal hardware interrupts
    - Generated by certain event during a program's execution, for example, divided by zero
    - Every such event is assigned with a specific interrupt number which is electronically wired into the processor and therefore cannot be modified
  - External hardware interrupts
    - Caused by peripheral device controllers
    - Also assigned with specified numbers and cannot be modified
  - Software interrupts
    - Generated by system and application programs
    - Some are used to activate specialised application programs

DOS PROCESS MANAGEMENT

- Interrupt synchronisation
  - Stack for
    - PSW (Program Statuts Word)
    - code segment register
    - instruction pointer register
  - Disables the interrupt system until the current interrupt has been solved
    - Placing a 8-bit number on the systems bus

WINDOWS PROCESS MANAGEMENT

- Multithreading
  - Elements of a process
    - An executable program
    - Private memory area
    - System resources allocated by an O/S
    - At least one thread of execution
  - Elements of a thread
    - A unique identifier
    - The contents of a volatile set of register indicating the processor's state
    - Two stacks used during the thread exectuion
    - A private storage area used by subsystems and dynamic-link libraries

WINDOWS PROCESS MANAGEMENT

- Multithreading synchronisation
  - Problem
    - Several treads can modify the same global variable independently of each other
    - Competition/racing for single shared resource
  - Synchronisation in Win32
    - Mutexes – only one thread can own the resource at a time
    - Semaphores – multiple threads can own it at a time
    - Critical section – a critical section can only be owned by a process and cannot be shared between processes
    - Event object – it is sent to all threads to alert them of an action occurring

LINUX PROCESS MANAGEMENT

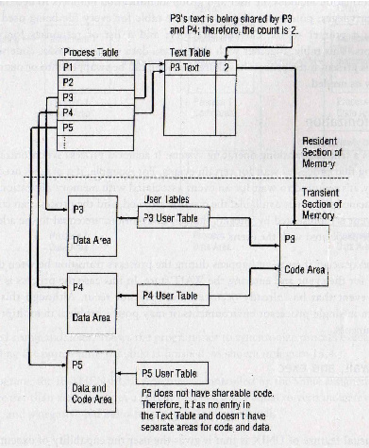
- Priority
  - Priority is largely determined by accumulated CPU time
  - Computer-to-total-time ratio:

$$\frac{\text{CPU time a process has used}}{\text{Total time CPU time required by the process}}$$

  - A process that has used a lot CPU time gets the lowest priority
  - Computer-to-total-time ratio is updated every second
  - Round-robin is used to decide which process will run first among those that have the same priority

Linux Process Management

- Tables
  - Process with sharable code
    - Resident Section of memory has two tables
    - Process Table shows all processes
    - Text Table shows the relationship between the processes and code, i.e. which processes share the same code and the memory address of the code

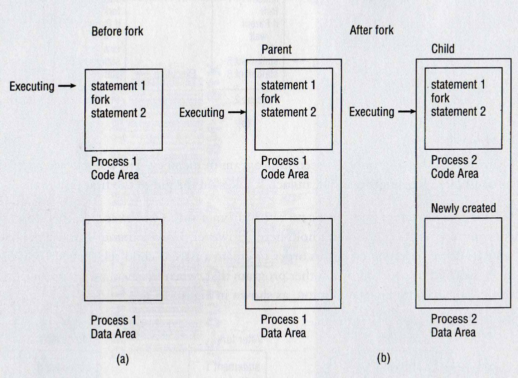


LINUX PROCESS MANAGEMENT

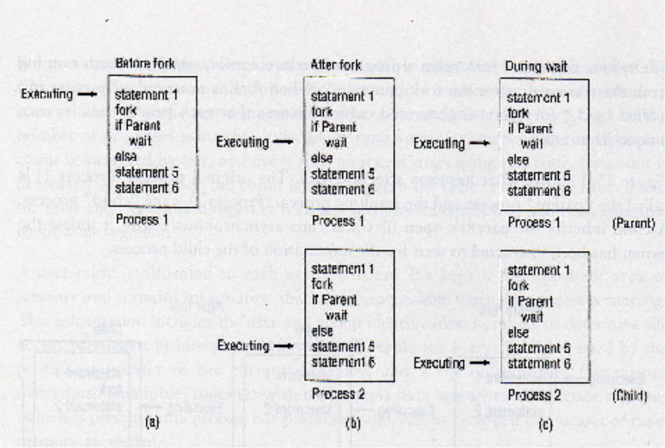
-
  -
    - Transient Section has Data area, Code area and User Table for each process
    - Data and code are stored separately because code is sharable
    - User Table is a map between Data area and Code area so it controls the access privilege of data/files
    - Example – P3 and P4
  - Processes with nonsharable code
    - Data and code are stored in the same area
    - Example – P5

LINUX PROCESS MANAGEMENT

- Fork, wait and exec
  - fork
    - Creates a copy of a process
    - The original one is called parent
    - The copy is called child



- - wait
    - Allows programmers to synchronise process execution



LINUX PROCESS MANAGEMENT

- - exec
    - Is a family of commands – execl, execv, execls, execlp and execvp
    - Are used to start a process from another process