Lecture 02: Operating Systems Services

The OS provides various services to the user. These services vary with the OS in question but usually include most of the following:

- Protection and Security
- Privileged Instructions
- Interrupt and Branch Management
- Buffering
- Spooling
- Resource Allocation and Job Scheduling
- Processor Modes and Inter-process Communication
- Accounting
- User Interface
- Object Management
- Help Facility (modern systems)
- On-line Manuals (modern systems)
- Interactive Command Prompt (modern systems)
- Useful Utilities and Information
- Programming Language Support
- Messaging: User-User Communication
- Communications Support
- Backup and Recovery
- CPU Scheduling and Resource Allocation
- Memory Management

This lecture briefly discusses these services, but bear in mind that several of them will be revisited as you progress through the course. As usual, the lecture ends with a summary and some concluding remarks.

2.1 Protection and Security

Protection

Processes in the OS must be protected from each other's activities. If processes are allowed to interfere with each other's work space, this will bring the system down.

Files, memory segments, CPU and other resources must be protected so that they are operated on only by those processes which have gained authorization from the OS.

Owners of objects must be able to specify how these objects are to be accessed by other users (& processes) in the system.

Security

Only authorized users must have access to the system. Additionally, it must be possible to grant/revoke specific permissions to/from authorized users of the system.

The OS should allow application developers to dictate authority constraints to applications, files, directories/libraries and other objects created by users.

2.2 Privileged Instructions

The OS has a set of programs (instructions) which are not accessible to the user. These are called privileged instructions. Only the OS can issue privileged instructions.

Privileged instructions cause interrupts to be effected.

2.3 Interrupt and Branch Management

An interrupt is an unexpected signal from the OS (to a running program) to finish the current operation and start something else.

Housekeeping procedures associated with an interrupt include the following:

- Save PC and CPU state (registers) to a stack;
- Load PC with interrupt vector;
- Execute interrupt instructions;
- Restore CPU state (from the stack);
- Restore previous PC value.

2.3 Interrupt and Branch Management (continued)

A similar situation occurs in the case of programmed branching e.g. a subroutine call:

- Save PC and relevant CPU registers to a stack;
- Load PC with branch address;
- Execute branch instruction(s);
- Restore CPU registers;
- Restore PC.

Note:

1. An interrupt is a system call while a branch is a program call.
2. An interrupt results in the entire CPU state being saved; a branch does not.

2.4 Buffering

Buffering is the use of auxiliary storage in such a way as to ensure that the CPU and I/O device(s) are kept busy:

- Buffering on input: The I/O device reads data into memory buffer. CPU gets input from memory buffer.
- Buffering on output: CPU outputs to memory buffer. The I/O (output) device gets output data from the buffer.

How is efficiency of I/O devices and CPU ensured?

- As soon as an I/O device has completed its input/output the CPU is interrupted.
- The CPU responds to the interrupt, which might be to issue another I/O instruction for that process, or move on to some other instruction for that process.
- The CPU then resumes previous activity.

Buffering is necessary but not sufficient to improve CPU performance. If there are several I/O bound or CPU bound jobs, buffering is of little help.

2.5 Spooling

An acronym for *"simultaneous peripheral operating on-line"*, spooling involves the use of magnetic disks. Information can be read from the disk and written to it at the same time (revise Computer Organization & Architecture notes).

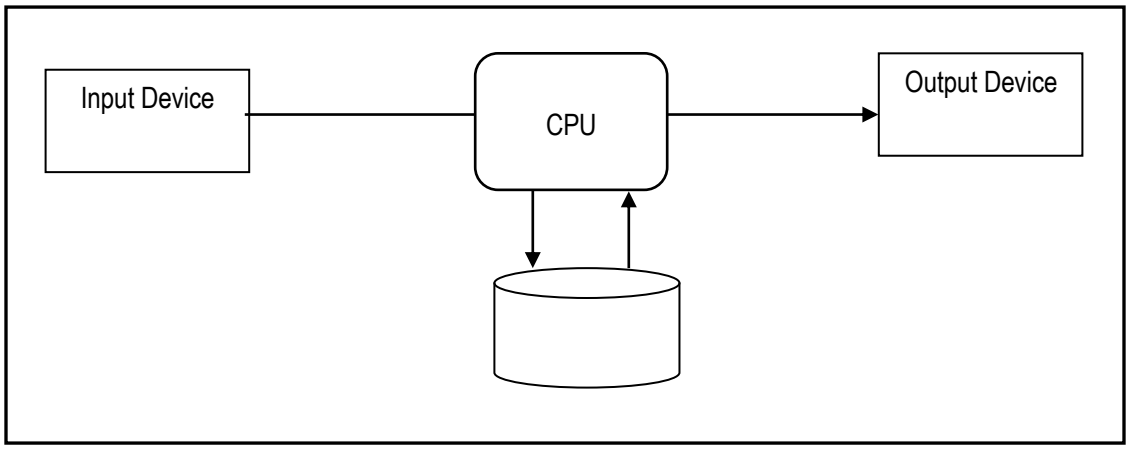
Spooling uses the disk as a large buffer — reading as far ahead as possible on input devices and storing output files until the output device can accept them.

Buffering overlaps the I/O of a job with its own computation; spooling can overlap the I/O and computation of several jobs.

2.5 Spooling (continued)

Spooling provides a very important data structure — the job pool. A pool of jobs allows the OS to select the next job to be run in order to increase CPU utilization. Job selection is normally done by a job scheduler.

Figure 2.1: Illustrating the Concept of Spooling



2.6 Resource Allocation and Job Scheduling

When competing jobs are running in an OS, they need resources in order to complete. The main resources include memory access, CPU time, and access to records in files. The OS presides over the schedule and usage of all system resources by issuing appropriate instructions to the CPU. These are huge topics that will be further discussed in lectures 04 - 07.

2.7 Processor Modes and Inter-process Communication

The OS determines processing modes in some instances automatically, and in other instances by allowing the user to choose.

Example: The IBM i allows users to (choose to) operate either in batch mode or interactive mode.

The OS also takes care of inter-process communication. This is normally facilitated via program calls, message queues. Reentrant code (discussed in lectures 5 and 6) may also be used.

2.8 Accounting

The OS may provide the facility of job accounting by automatically storing information on users on the system, application(s) used, files (and other objects) used, time spent etc.

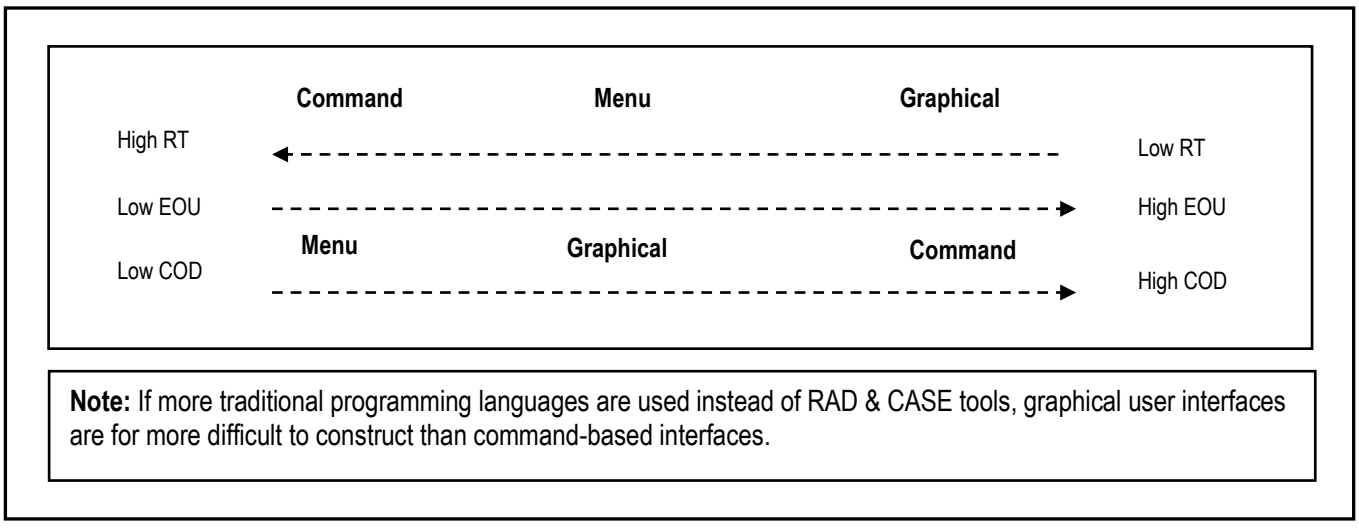
This information may be useful in the situation where access fees are chargeable. It is also useful in the eventuality of a catastrophic system failure.

2.9 User Interface

The user interface is the medium used to facilitate communication be the user and the OS. The user interface for the OS may be any of the three standard types of user interfaces — a command-based interface, menu-driven interface, or a graphical user interface (GUI). Additionally, it may be a combination or two or more types of the standard user interfaces. Usually, there is a control language (i.e. a command language) which can be learnt and mastered by those interested in becoming intimate with the system.

Figure 2.2 compares the three standard user interfaces in terms of relative complexity of design (COD), response time (RT), and ease of use (EOU). Generally speaking, command-based interfaces are the most efficient, and GUI's provide the most convenient. Therefore, OS designers often have to make a tradeoff.

Figure 2.2: Comparison of User Interface Categories



Note: If more traditional programming languages are used instead of RAD & CASE tools, graphical user interfaces are for more difficult to construct than command-based interfaces.

2.9 User Interface (continued)

Examples 1:

1. Early versions of DOS were command based; later versions were command based and graphical.
2. Windows and Mac operating systems tend to be primarily GUI-based, but intermittently provide command-based instructions.
3. Linux and Unix operating systems primarily use command-based interface, but also support GUI access via X-Windows.
4. The IBM i does an excellent job of consistently providing users with the option of using any of the three standard user interfaces, depending on their preferences. Expert users can use the command language, while novices can use the GUI or menu-driven interface.
5. PICK was command-based and menu driven.

2.10 Object Management

A library (or directory or folder) is the holding area for assorted objects - those with which the OS is shipped and those created by the users.

All systems have library (or directory or folder) management strategies and methodologies, as well as file management strategies. These are mandatory services. The strategies used may vary from system to system. Further discussion will ensue in the following lecture.

2.11 Help Facility and Online Manuals

There are three alternatives for structuring the help system in an OS: panel-by-panel, context sensitivity, and hypermedia style.

Panel-by- panel Help: Each panel displayed by the software has a corresponding help panel, invoked when the user makes a request for it. The user request is typically affected by pressing a given function key on the keyboard, or clicking at a help button on the screen.

Context Sensitive Help: Depending on the cursor position at the time of user request, help information, specific to that locality is provided. This approach is difficult to develop and maintain, since there must be close synchronism between cursor location and help information provided. Example: The operating system IBM i has a help facility is completely context sensitive.

Hypermedia-based Help: The help facility is developed as a hypermedia system, invoked by the user making an explicit request (clicking help menu/button or pressing a function key). The user accesses information based on the choices made from a menu, an index or a hyperlink. This approach has become very popular, because it comes close to offering the same conveniences as the context sensitive help, but with much less complexity of design.

The on-line manual, if present, can usually be easily printed. Traditionally, this was sent to the customer in printed. The contemporary practice is to provide the manual in electronic form.

2.12 Interactive Command Prompt

Interactive command prompt is the ability to obtain assistance for completing a command while actually issuing the command. Some operating systems have this feature; of the few that have it, the IBM i stand as the most outstanding.

Interactive command prompt is a very expensive overhead, but it is extremely useful and almost guarantees a user friendly environment.

2.13 Utilities and Programming Language Support

The OS normally provides various utilities for addressing common user problems e.g. copying, connectivity to some other system, provision of date and time etc.

A programming language interface which supports certain high level languages is usually provided.

Examples 2:

Traditionally,
1. DOS supported BASIC, C, COBOL, C++, Pascal, etc.
2. UNIX supported Fortran, Pascal, Java, C, C++, Pascal, etc.
3. OS-400 supported COBOL, C, Java, RPG-400, etc.
4. Windows NT/2000 supported BASIC, C, COBOL, C++, Java, Pascal, etc.

The programming language interface normally supports certain DBMS suites and CASE tools also.
The modern trend is towards OSA (Open System Architecture) standards so that all OSs seek to support a basic set of high level languages, DBMS suites and CASE tools.

Still, a more recent development (since the "Java revolution") is to have compilers which are designed to run on multiple OS platforms, thus shifting the focus from OS design to compiler design.

2.14 Messaging: User-User Communication

Users are able to communicate with each other from different workstations via messages (electronic mail and chat are the most popular examples). Some systems are quite sophisticated, providing both mailing and faxing facilities. Others are fairly simple.

Examples 3:

1. Windows uses **Windows Messenger** and **e-mail**. There is also a third-party products called Net Windows, which piggy-backs off Windows Messenger.
2. Linux and Unix use **mail**, **send-mail**, and **e-mail**.
3. OS-400 uses **Workstation Message**, **User Message**, **Program Message**, and **e-mail**.

2.15 Communications Support

Traditionally, communication support would be omitted from small (microcomputer-based) operating systems, but included in more powerful operating systems. Vendors would use third-party communications software for these small systems.

The contemporary trend is to built-in communications support in the OS. Communications support and interoperability are important OS services for modern systems; in fact they determines to a large extent, the success of the OS in the marketplace. Interoperability is the term used to describe the capacity of the OS to communicate with applications and systems running on different OS platforms.

2.16 The User View

Two methods of providing services are System Calls, and System Programs.

Systems Calls

System Calls provide the interface between a running program and the OS. The categories are as follows:

- Process Control
  - End, Abort
  - Load, Execute
  - Create Process, Terminate Process
  - Get Process Attributes, Set Process Attributes

- File Manipulation
  - Create, Delete
  - Open, Close
  - Read, Write, Reposition
  - Get File Attributes, Set File Attributes

- Device Manipulation
  - Request Device, Release Device
  - Read, Write, Reposition
  - Get Device Attributes, Set Device Attributes

- Information Maintenance
  - Get Time/Date, Set Time/Date
  - Get System Parameters, Set System Parameters
  - Get Process, File, or Device Attributes
  - Set Process, File, or Device Attributes

2.16 The User View (continued)

System Programs

Most systems provide (utility) programs to solve common problems e.g. copying. They can be divided into several categories:

- File manipulation
- Status information
- File modification
- Programming language support
- Application programs and utilities
- The OS command interpreter

2.17 The Operating System View

Operating systems are event-driven (interrupt-driven) systems — if there are no users, no jobs, nothing will happen. Types of interrupts may be:

- System Call which includes:
  - Normal job termination
  - Abnormal job termination
  - Status request
  - Resource request
  - I/O request
- I/O Interrupt
- Program Error

2.18 Summary and concluding Remarks

This lecture provided an overview of the fundamental services provided by an operating system: protection and security; privileged instructions; buffering; processor modes and inter-process communication; resource allocation and job scheduling; accounting; user interface; I/O operation; file management; library/directory management; spooling; interrupt and branch management; help facility; on-line documentation; interactive command prompt; useful utilities and information; programming language support; messaging; backup and recovery; communications support; CPU scheduling; memory management.

As we proceed through the course, we will be delving into these services in more detail. The next lecture discusses object management.

2.20 Recommended Reading

[Bacon & Harris 2003] Bacon, Jean S. and Tim Harris. 2003. *Operating Systems: Concurrent and Distributed Software Design*. Boston, MA: Addison-Wesley. See chapters 1 – 3.

[Davis & Rajkumar 2005] Davis, William S. & T. M. Rajkumar. 2005. *Operating Systems: A Systematic View*, 6th ed. Boston, MA: Addison-Wesley. See chapters 1 – 3.

[Nutt 2004] Nutt, Gary. 2004. *Operating Systems* 3ed. Boston, MA: Addison-Wesley. See chapters 1 – 4.

[Silberschatz 2012] Silberschatz, Abraham, Peter B. Galvin, & Greg Gagne. 2012. *Operating Systems Concepts*, 9th Ed. Update. New York: John Wiley & Sons. See chapter 1-2.

[Stallings 2005] Stallings, William. 2005. *Operating Systems* 5th ed. Upper Saddle River, New Jersey: Prentice Hall. See chapters 1 – 2.

[Tanenbaum & Woodhull 1997] Tanenbaum, Andrew S., & Albert S. 1997. Woodhull. *Operating Systems: Design and Implementation* 2ed. Upper Saddle River, NJ: Prentice Hall. See chapter 1.