Operating Systems in Practice

What is an Operating System (OS)?

An operating system is software (i.e. computer programs) that enables users and other programs to communicate with hardware. An operating system means it's relatively easy to use a computer for example if we want to print a document – it's as easy as clicking on a print icon on the screen.

Early computer systems had no operating systems. Programs were entered manually (in binary), ran and were monitored by the programmer.

This was very time consuming, required specialist knowledge and prone to errors. In time, new hardware (printers, tape & punched cards) and software (device drivers, compilers, etc) were developed.

Operating Systems ….

Operating systems have two main roles:

- **An Interface**
- **A Manager**

Operating Systems Objectives

Convenience – An OS helps to make it easy to do tasks. Users can interface with it.

Efficiency – An OS makes efficient use of resources such as memory, CPU, etc.

Evolution – An OS should enable development of new systems.

Convenience - User Interface

Most users are not concerned with hardware.

Users generally view a system as a set of applications.

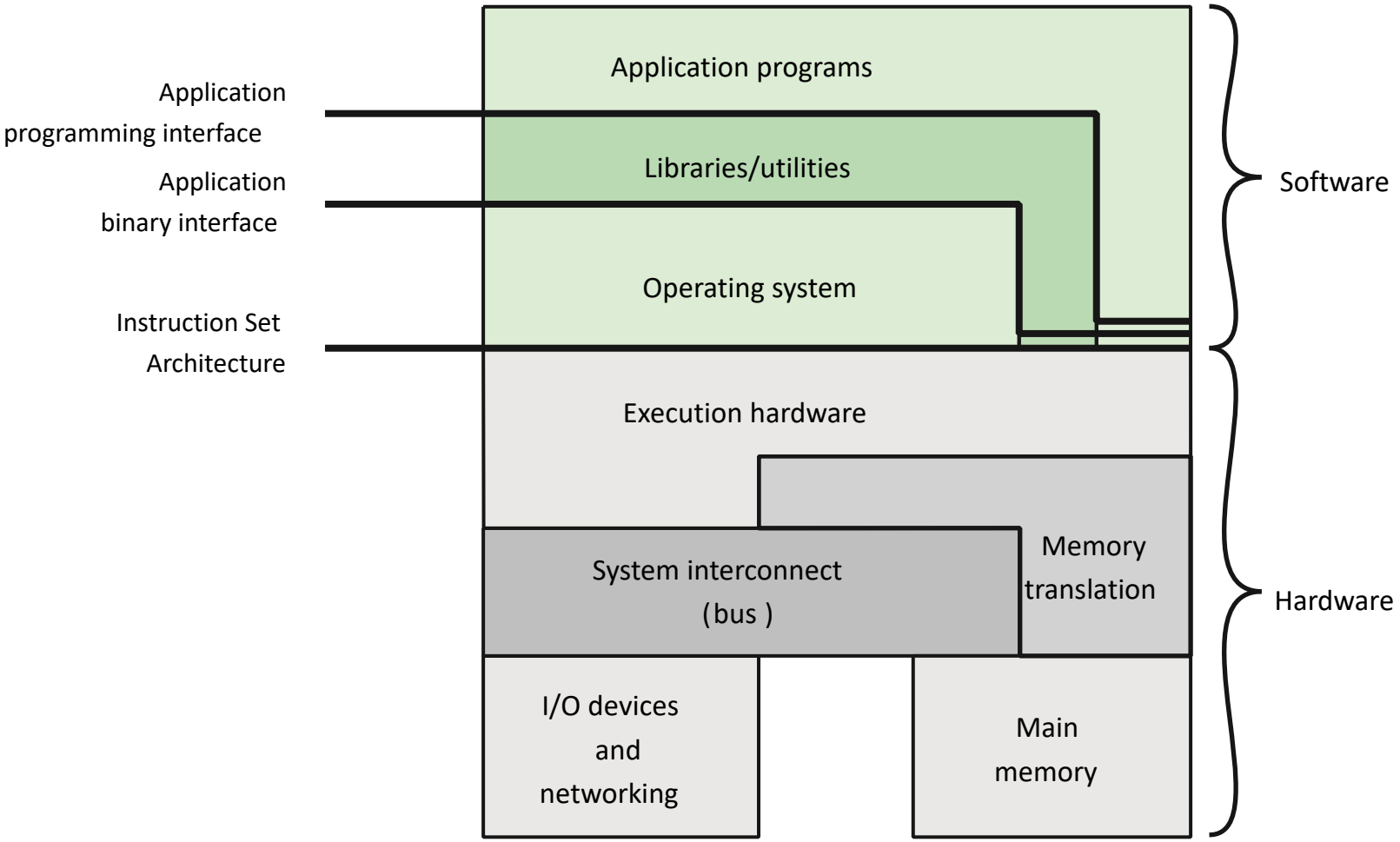
Applications are developed by programmers who are facilitated by operating system programs.

The OS hides hardware details from the programmer and provide an interface for using the system.

Interface

- An OS is a suite of programs which provides an *interface* between the user and the hardware.
- The OS enables complex interactions required by a running program when dealing with processor(s) (CPU), main memory (RAM), secondary storage (disks, tapes), printers, keyboards, screens, network cards, etc.
- It also presents the user/application programmer with a simplified (but effective) view.

Hardware & Software Structure



Operating System Services

The OS provides services to users and programmers. The user accesses these services through either systems programs or application programs. The programmers (both systems programmers and application programmers) call on these services. In general, the services provided include:

- **Program Development** – The OS offers a variety of services and facilities such as editors and debuggers. These assist developers in creating programs and although strictly not part of the core of the OS – they are generally supplied with the operating system.

Services (cont.)

- **Program Execution** – The OS will help to compile, link, load, run and debug programs. OS will allow normal and abnormal termination of programs. It generally handles the scheduling of these tasks.

- **I/O operations** - All input and output from/to devices is handled by OS. The OS provides a uniform interface to these.

- **Controlled access to files** – the OS will open, close, create, delete, and move files about on disk/tape. On a system with multiple users, it may provide protection to individual users' data.

Services (cont.)

- **System Access –** For shared or public systems, the OS controls access to the system as a whole. This provides protection from unauthorized users

- **Error detection** – the OS watches for errors in: - h/w (e.g. power failure, disk head crash); i/o devices (e.g. printer out of paper); - programs (e.g. arithmetic overflow, division by zero).

- **Resource allocation** - Any resources required by programs are requested from OS.

Services (cont.)

- **Accounting** - The OS keeps track of resource use for either payment, protection or statistical purposes. Protection accounting allows system to record damage done. Statistics are used in trying to improve service.

- **Protection** – As well as ensuring private data is kept private (e.g. mailboxes, personnel records) the OS protects programs from each other (especially other OS programs).

Users and Services

Users access these services by using programs. These system programs use *system calls* to get work done.

This means that they call on the OS to perform services on their behalf; the programs do not access the hardware directly, only through the OS.

Users and Services (cont.)

The systems programs allow the user to: (examples are from UNIX / Linux)

- **Control processes** - Users can manipulate processes. For example create (**su**), destroy (**kill**), suspend (**ctrl-Z**), send to background/foreground (**bg/fg**), etc.

- **Manipulate files** - Users can edit (**vi**), copy (**cp**), delete (**rm**), display (**more**), examine attributes (**ls -l**), etc. All of these operations involve lower level actions in programs such as open, close, read, write, etc.

Users and Services (cont.)

- **Use devices** - Users can print (**lp**), clear screens (**clear**), etc.
- **Get information** – Run commands such as **who, date, ls, ps** (list processes), **du** (disk usage), etc.
- **Write and run programs** - Use editors (**vi**), compilers (for **Java, C++,** etc.), assemblers, linkers, loaders, debuggers, etc. are all provided.
- **Run application programs** – such as browsers, office programs, etc.

Multiprogramming

- Multiprogramming refers to the task of running several programs at the same time. For example, a payroll program, a printing task, a statistics program, all doing their separate work. It is up to the OS to divide its time between the jobs.

- The OS is capable of dealing with more than one program running at the same time. For example, while one job is waiting for input, another could be running some computation, while yet another could be waiting for an output device, and so on. All of these programs are said to be running *concurrently*. The job scheduling task of the OS was to juggle the CPU time so that each job got access to it in some fair and efficient manner. This task is referred to as *CPU Scheduling*. CPU scheduling depends on being able to interrupt one program and resume another.

Multiprogramming (cont.)

- When several jobs are ready to run on the CPU they must all be stored in main memory. In a multiprogramming system there must be some way of managing the main memory so that each program gets the space it needs and doesn't interfere with the others. This management task is known as *Memory Management.*

- Because multiprogramming is allowed then the OS must cope with *concurrency issues.* Programs running concurrently could be half way through printing a document or writing data to a database when interrupted – they must be guaranteed that no other program will interfere with the work they are doing by printing on the same paper or reading half-baked data from the database. These issues are *concurrency issues.*

Time Sharing

In addition to multiprogramming, the OS facilitates time sharing.

A *time sharing* system allows several users/programmers access to the machine simultaneously. To each it looks as if they have the machine to themselves.

In reality the CPU is switching from one job to the next very quickly. All of this is possible because the CPU can multi-program.

Manager

- The OS also *manages* the allocation of machine resources to running programs.

- These resources include the processor, memory and storage. It must manage this allocation fairly and efficiently.

Managing Processes

Multiple processes run concurrently.

The CPU time has to be allocated fairly and efficiently.

Processes need to be interrupted and resumed at a later time.

Managing Memory

Each process requires memory.

Each process's memory needs to be protected.

Data and processes constantly move around cache, RAM and virtual memory

Managing Storage

Data and programs can be stored in different non-volatile devices.

Space needs to be allocated in an efficient manner.

Data and programs need to be retrieved in an efficient manner

Kernel and System calls

- The *kernel* is the core of the OS. Kernel software has exclusive access to system data and h/w. Remaining software running in *user mode* has limited direct access to system data and no direct access to h/w.

- All application and utility programs access the fundamental OS operations through OS system calls. These *system calls* are the only way to access the h/w; that is, they provide the interface between the programs and the OS.

Kernel and System calls (cont.)

Example: write a program to output 'Hello World' to a screen the program will contain some instruction like: **system.out ('Hello World');**

This call is translated by the compiler/assembler into code which will call the OS giving it information about the data to be output, and the output device. That is something like:

system_i/o_display(data_address, no_of_bytes, device_identifier);

A system call to the OS will ask the kernel to take a certain number of bytes from RAM starting at data_address and have them displayed on a particular device. The kernel will arrange to call the correct device driver software to communicate with the screen.

Command Interpreter

- All operating systems must have some way of understanding and carrying out instructions from users. This is achieved through a command interpretation program which runs continuously. The user types a command, the interpreter translates it into something meaningful and then calls up the necessary system programs, utilities, or applications that will carry out the work. The interpreter is less obvious in a GUI environment (e.g. Windows) and more so in a command based environment (e.g. UNIX shell).

- These interpreters usually understand more than simple commands. They can interpret and carry out 'command programs'. That is, you can write programs which instruct the system to carry out a complex series of tasks. In UNIX the command interpreter is called the 'shell' and we will be doing some programming with the UNIX shell.

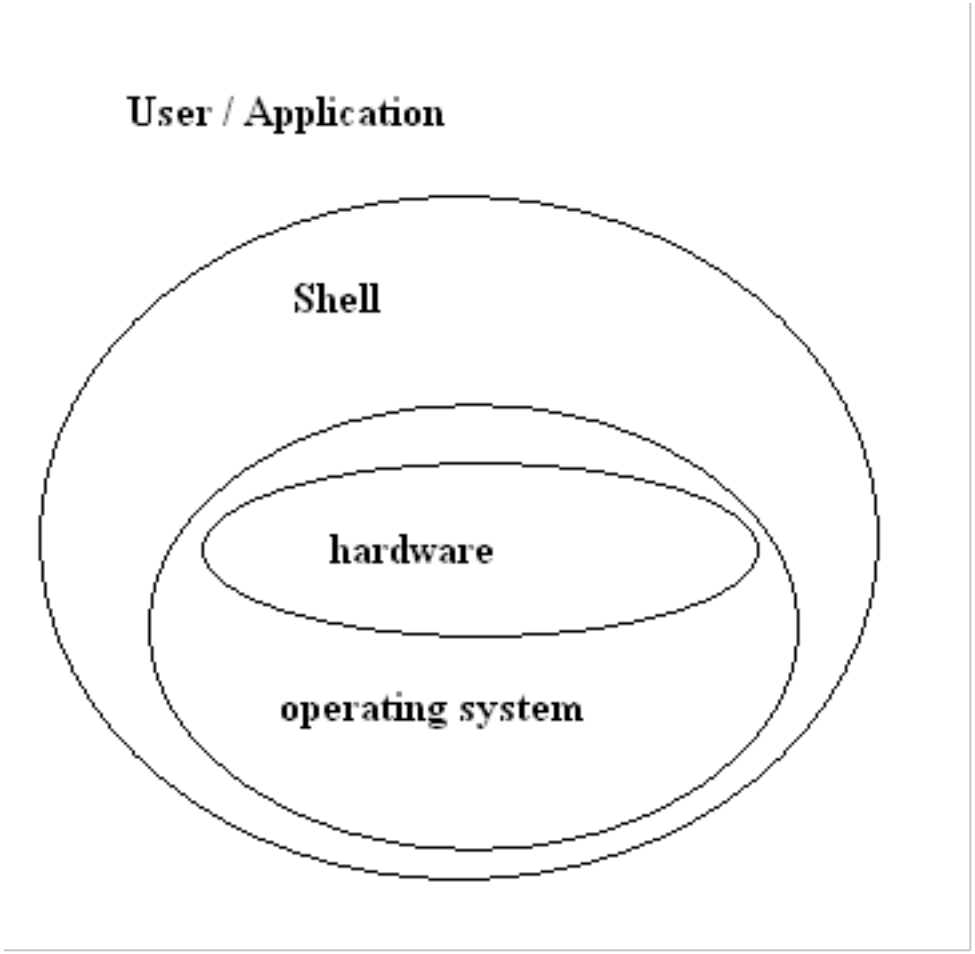
Command Interpreter (cont.)

The following represents a typical system:

The hardware is controlled (and can only be accessed) by the operating system.

The shell acts as an interface to the operating system by users and applications.

The Shell



Introduction to UNIX / Linux

UNIX was first developed about forty years ago in Bell Laboratories, U.S.A. It is a popular and robust system that has survived over the years. UNIX provides an environment that's powerful and flexible and found in business, science, educational institutions and industry.

UNIX Features

- Multitasking capability – capable of doing several things at once.
- Multi-user capability – many users can use the system concurrently.
- Portability – it was relatively easy to move UNIX from one type of computer to another.
- E-Mail – was the first operating system to offer email.

UNIX

The UNIX system is functionally organized at three levels:

- The **kernel**, which schedules tasks and manages storage
- The **shell**, which connects and interprets users' commands, calls programs from memory, and executes them
- The **tools** and **applications** that offer additional functionality to the operating system

The Kernel

The heart of the operating system, the kernel controls the hardware and turns part of the system on and off at the programmer's command.

If you ask the computer to list (*ls*) all the files in a directory, the kernel tells the computer to read all the files in that directory from the disk and display them on your screen

The Shell

- There are several types of shell, most notably the command driven Bourne, BASH and the C Shells. Whatever shell is used, its purpose remains the same - to act as an interpreter between the user and the computer

- The shell also provides the functionality of "pipes," whereby a number of commands can be linked together by a user, permitting the output of one program to become the input to another program.

The Shell (cont.)

The shell enables the writing of operating systems programs known as shell scripts.

A script consists of commands which are interpreted and executed.

UNIX tools & applications

UNIX tools are not strictly part of the OS but are always included. These include:

Editors – used to create scripts, data files, etc. The vi editor (‘though not user friendly) comes with every version of UNIX and Linux.

Compilers and libraries are generally included to facilitate program development.

Most Linux systems come with a graphical user interface and often a suite of office programs.

UNIX Advantages

- Already exists for over 40 years
- Available for almost any hardware platform
- Multi tasking
- Multi user
- Made to keep on running (servers)
- Secure and versatile
- Scalable

Disadvantages of UNIX

- UNIX commands can be difficult to remember – wasn't user friendly
- Not enough desktop or Office applications
- Has to compete with popularity of Windows
- Editors like **vi** difficult to use

Linux

- In 1991 Linus Torwalds at the University of Helsinki build a UNIX-like operating system for IBM compatible PCs. He used freely available source code to develop what would become known as "Linux" ("**Linu**s' Uni**x**"). By 1994 he released the first stable Linux Kernel (version 1.0). This was further developed by others.

- One of the reasons for Linux's popularity is that the source code is freely available to everyone, and everyone can contribute to its development. This effectively added thousands of programmers to the Linux development team. The large number of Linux developers yielded an operating system of unprecedented efficiency and robustness, with countless freely available software packages for both business and pleasure.

Linux Advantages

- **Low cost:** You don't need to spend time and money to obtain licenses.
- **Stability:** Linux doesn't need to be rebooted periodically to maintain performance levels. It doesn't freeze up or slow down over time due to memory leaks and such.
- **Performance:** Linux provides persistent high performance on workstations and on networks. It can handle unusually large numbers of users simultaneously.
- **Network friendliness:** Linux was developed by a group of programmers over the Internet and has therefore strong support for network functionality.
- **Flexibility:** Linux can be used for high performance server applications, desktop applications and embedded systems. You can save disk space by only installing the components needed for a particular use.

Linux Advantages (cont.)

- **Compatibility:** It runs all common UNIX software packages and can process all common file formats.
- **Choice:** The large number of Linux distributions gives you a choice. Each distribution is developed and supported by a different organization. You can pick the one you like best; the core functionalities are the same; most software runs on most distributions.
- **Easy installation:** Most Linux distributions come with user-friendly installation and setup programs.
- **Multitasking:** Linux is designed to do many things at the same time; e.g., a large printing job in the background won't slow down your other work.
- **Security:** Linux is one of the most secure operating systems. "Walls" and flexible file access permission systems prevent access by unwanted visitors or viruses.
- **Open source:** If you develop software that requires knowledge or modification of the operating system code, Linux's source code is at your fingertips.

Linux Advantages (cont.)

Linux GUIs – there are a variety of Linux graphical user interfaces which make using Linux easier. These can be changed by downloading and installing the desktop environment of your choice.

Popular choices are KDE and GNOME.

Linux Advantages (cont.)

Software

Often when Linux is installed it comes with free office software such as openOffice or LibreOffice.

These contain sophisticated word processing, spreadsheet, database and presentation graphics programs.

Other software – programs for a variety of applications (many are free) are available.

Linux Distributions

Linux Distributions

Linux distribution is a coherent collection of free software with the kernel as its center. To run this you normally need a Linux distribution CD.

- **Red Hat** Linux has been around for a while and has acquired a reputation for consistency and reliability.
- **Mandrake** Linux has become very popular in recent years, especially among new and home users.
- **SuSE** Linux is a serious alternative for Windows users, with solid, user-friendly installation and configuration tools
- Others are **Lycoris, Xandros**, **Ubunto, Lindows, Knoppix** and **Slackware**

Acquiring Linux

There are a number of sites from where you download and install Linux.

Select the Linux distribution of your choice.

Download the iso which can be installed on your computer.

Installing Linux

Linux can be installed on your hard disk on a single partition (also creates a swap partition)

Install on a separate partition on the computer's hard disk e.g. have Windows on one partition and Linux on another

Install on a CD / DVD

Install on a USB device

Install on a virtual machine.

Linux Software

Once Linux is installed, software can be downloaded and installed. There are various methods of installation – software may be available in the following formats:

tar.gz – a compressed archive

tar.Z – a different compressed archive

rpm – requires installing using the red hat package manager

Example -

Example :the java jdk environment may offer the following options:

now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.92 MB	jdk-8u161-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.88 MB	jdk-8u161-linux-arm64-vfp-hflt.tar.gz
Linux x86	168.96 MB	jdk-8u161-linux-i586.rpm
Linux x86	183.76 MB	jdk-8u161-linux-i586.tar.gz
Linux x64	166.09 MB	jdk-8u161-linux-x64.rpm
Linux x64	180.97 MB	jdk-8u161-linux-x64.tar.gz
macOS	247.12 MB	jdk-8u161-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	139.99 MB	jdk-8u161-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.29 MB	jdk-8u161-solaris-sparcv9.tar.gz
Solaris x64	140.57 MB	jdk-8u161-solaris-x64.tar.Z
Solaris x64	97.02 MB	jdk-8u161-solaris-x64.tar.gz
Windows x86	198.54 MB	jdk-8u161-windows-i586.exe
Windows x64	206.51 MB	jdk-8u161-windows-x64.exe

Software Package Management + Repositories

A repository is a collection of software that can be downloaded from a server. It requires a package management tool to install.

This provides a significant amount of free software across a variety of applications.

Software Package Management

The Package Manager

Package downloading: allow users to download from a trusted provider.

Dependency resolution: provides information about what other files are required

Standard format: makes installation easier

Quality control: tests software is secure and stable

The Package Management apt

apt is a popular and powerful method of package management.

It is the usual package management system used with Ubuntu.

The apt-get command uses the tool to interact with the operating system.

The Package Management apt

Useful Commands:

apt-get install: Installs the package with dependencies

apt-get remove: removes package

apt-get update: updates list of available packages

apt-get upgrade: (run both) – upgrades if upgrades available

apt-get example

- To install a new service (e.g. at)

apt-get install at

The at daemon can now be used.

Other Package Managers

RPM – Red Hat Package Manager

Originally created for Red Hat Linux, it is now used in many distributions.

Zypper

This is used for package management in openSuse and Suse.