Unix OS Introduction

August 27

2012

Basically It Talks about Information of Unix OS and Advamntage over other OS. Gives Blocks of UNIX and Principles.

SHEEBA DM

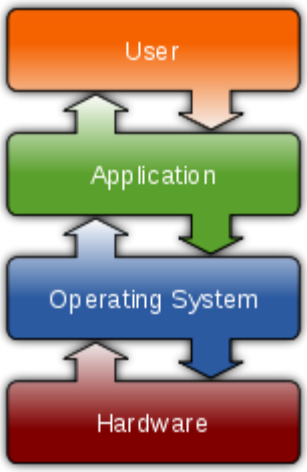
Unix Operating System Introduction

Table of Contents

1. What Is Operating System(OS) and Need for OS

Definition:

An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is a vital component of the system software in a computer system. Application programs require an operating system to function.



- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - ✓ Execute user programs and make solving user problems easier.
  - ✓ Make the computer system convenient to use.
- Use the computer hardware in an efficient manner.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware, although the application code is usually executed directly by the hardware and will frequently make a system call to an OS function or be interrupted by it. Operating systems can be found on almost any device that contains a computer—from cellular phones and video game consoles to supercomputers and web servers.

1.1 Need for OS

At the simplest level, an operating system does two things:

1. It manages the hardware and software resources of the system. In a desktop computer, these resources include such things as the processor, memory, disk space, etc. (On a cell phone, they include the keypad, the screen, the address book, the phone dialer, the battery and the network connection.)
2. It provides a stable, consistent way for applications to deal with the hardware without having to know all the details of the hardware.

The first task, managing the hardware and software resources, is very important, as various programs and input methods compete for the attention of the central processing unit (CPU) and demand memory, storage and input/output (I/O) bandwidth for their own purposes. In this capacity, the operating system plays the role of the good parent, making sure that each application gets the necessary resources while playing nicely with all the other applications, as well as husbanding the limited capacity of the system to the greatest good of all the users and applications.

The second task, providing a consistent application interface, is especially important if there is to be more than one of a particular type of computer using the operating system, or if the hardware making up the computer is ever open to change. A consistent application program interface (API) allows a software developer to write an application on one computer and have a high level of confidence that it will run on another computer of the same type, even if the amount of memory or the quantity of storage is different on the two machines.

Even if a particular computer is unique, an operating system can ensure that applications continue to run when hardware upgrades and updates occur. This is because the operating system and not the application is charged with managing the hardware and the distribution of its resources. One of the challenges facing developers is keeping their operating systems flexible enough to run hardware from the thousands of vendors manufacturing computer equipment. Today's systems can accommodate thousands of different printers, disk drives and special peripherals in any possible combination.

The operating system is the interface between the hardware and the user. If there were no O/S, the computer would be an expensive door stop.

2. Types of OS

2.1 Real-time

A real-time operating system is a multitasking operating system that aims at executing real-time applications. Real-time operating systems often use specialized scheduling algorithms so that they can achieve a deterministic nature of behavior. The main objective of real-time operating systems is their quick and predictable response to events. They have an event-driven or time-sharing design and often aspects of both. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

2.2 Multi-user

A multi-user operating system allows multiple users to access a computer system concurrently. Time-sharing systems and Internet servers can be classified as multi-user systems as they enable multiple-user access to a computer through the sharing of time. Single-user operating systems, as opposed to multi-user operating systems, are usable by a single user at a time. Being able to use multiple accounts on a Windows operating system does not make it a multi-user system. Rather, only the network administrator is the real user. But for a UNIX-like operating system, it is possible for two users to log in at a time and this capability of the OS makes it a multi-user operating system.

2.3 Multi-tasking vs. single-tasking

When only a single program is allowed to run at a time, the system is grouped as a single-tasking system. However, when the operating system allows the execution of multiple tasks at one time, it is classified as a multi-tasking operating system. Multi-tasking can be of two types: pre-emptive or co-operative. In pre-emptive multitasking, the operating system slices the CPU time and dedicates one slot to each of the programs. Unix-like operating systems such as Solaris and Linux support pre-emptive multitasking, as does AmigaOS. Cooperative multitasking is achieved by relying on each process to give time to the other processes in a defined manner. 16-bit versions of Microsoft Windows used cooperative multi-tasking. 32-bit versions, both Windows NT and Win9x, used pre-emptive multi-tasking. Mac OS prior to OS X used to support cooperative multitasking.

2.4 Distributed

A distributed operating system manages a group of independent computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they make a distributed system.

2.5 Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

2.6 Examples of popular modern operating systems includes

1. Android
2. BSD
3. iOS
4. Linux
5. Mac OS X
6. Microsoft Windows
7. Windows Phone
8. IBM z/OS

All these, except Windows and z/OS, share roots in UNIX.

3. About Unix

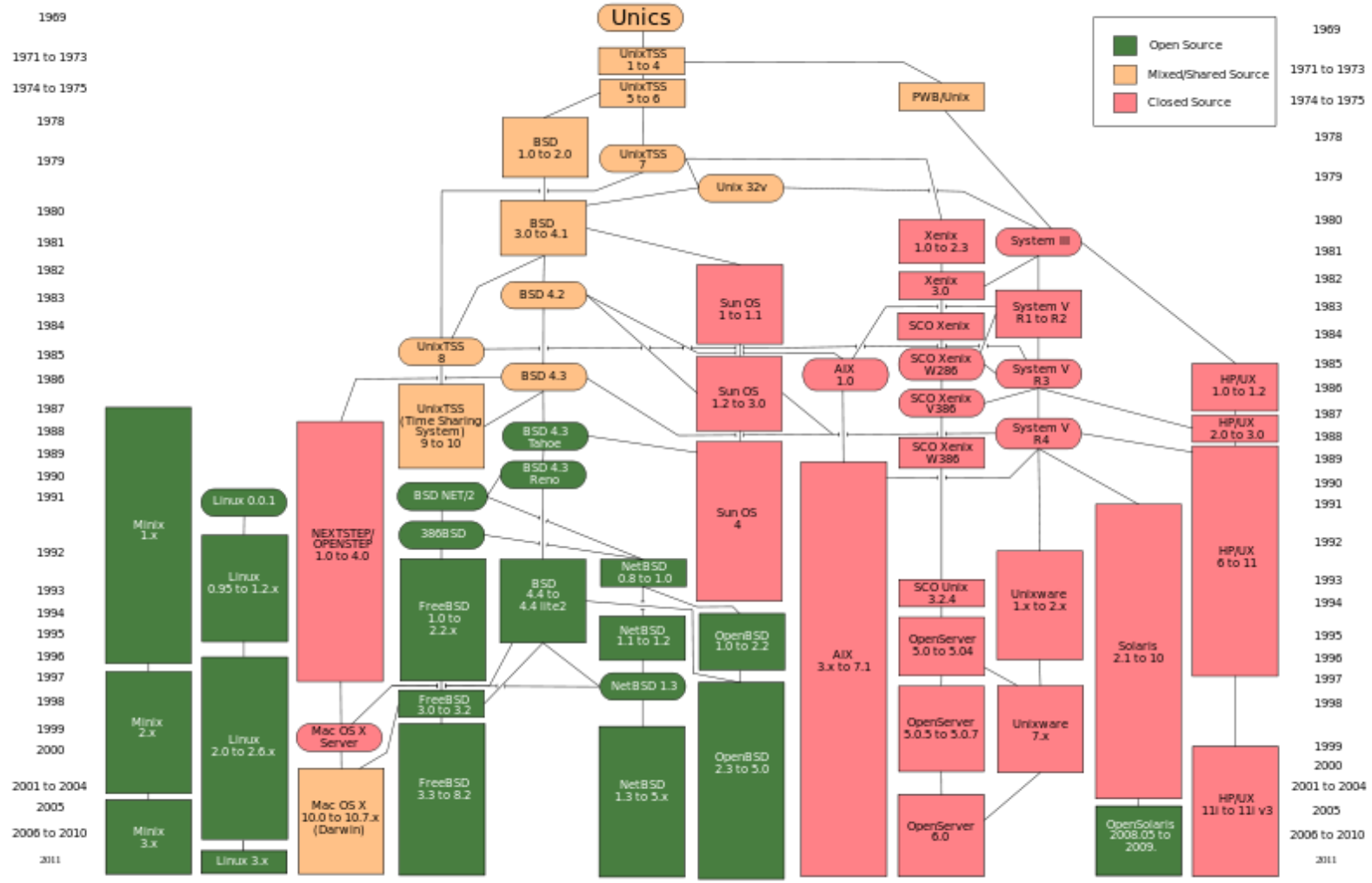
Unix (officially trademarked as UNIX, sometimes also written as Unix) is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk and Joe Ossanna.

The Unix operating system was first developed in assembly language, but by 1973 had been almost entirely recoded in C, greatly facilitating its further development and porting to other hardware. Today's Unix system evolution is split into various branches, developed over time by AT&T as well as various commercial vendors, universities (such as University of California, Berkeley's BSD), and non-profit organizations.

The Open Group, an industry standards consortium, owns the UNIX trademark. Only systems fully compliant with and certified according to the Single UNIX Specification are qualified to use the trademark; others might be called Unix system-like or Unix-like, although the Open Group disapproves of this term. However, the term Unix is often used informally to denote any operating system that closely resembles the trademarked system.

During the late 1970s and early 1980s, the influence of Unix in academic circles led to large-scale adoption of Unix (particularly of the BSD variant, originating from the University of California, Berkeley) by commercial startups, the most notable of which are Solaris, HP-UX, Sequent, and AIX, as well as Darwin, which forms the core set of components upon which Apple's OS X, Apple TV, and iOS are based. Today, in addition to certified Unix systems such as those already mentioned, Unix-like operating systems such as MINIX, Linux, and BSD descendants (FreeBSD, NetBSD, OpenBSD, and DragonFly BSD) are commonly encountered. The term traditional Unix may be used to describe an operating system that has the characteristics of either Version 7 Unix or UNIX System V.

3.1 History


Unics
UnixTSS 1 to 4
UnixTSS 5 to 6
UnixTSS 7
PWB/Unix
BSD 1.0 to 2.0
Unix 32v
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
Xenix 3.0
System
BSD 4.2
Sun OS 1 to 1.1
SCO Xenix
System V R1 to R2
UnixTSS 8
AIX 1.0
SCO Xenix W286
System V R3
BSD 4.3
Sun OS 1.2 to 3.0
SCO Xenix V386
HP/UX 1.0 to 1.2
UnixTSS (Time Sharing System) 9 to 10
BSD 4.3 Tahoe
SCO Xenix W386
System V R4
HP/UX 2.0 to 3.0
BSD 4.3 Reno
Sun OS 4
BSD NET/2
386BSD
NetBSD 0.8 to 1.0
HP/UX 6 to 11
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
AIX 3.x to 7.1
SCO Unix 3.2.4
Unixware 1.x to 2.x
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
Solaris 2.1 to 10
FreeBSD 3.0 to 3.2
NetBSD 1.3
OpenServer 5.0.5 to 5.0.7
Unixware 7.x
OpenBSD 2.3 to 5.0
FreeBSD 3.3 to 8.2
NetBSD 1.3 to 5.x
OpenServer 6.0
OpenSolaris 2008.05 to 2009.
HP/UX 11i to 11i v3
Minix 1.x
Linux 0.0.1
NEXTSTEP/ OPENSTEP 1.0 to 4.0
Linux 0.95 to 1.2.x
Minix 2.x
Linux 2.0 to 2.6.x
Mac OS X Server
Mac OS X 10.0 to 10.7.x (Darwin)
Minix 3.x
Linux 3.x
Open Source
Mixed/Shared Source
Closed Source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2010
2011

4. Advantage of Unix

4.1 Advantages

- ✓ Full multitasking with protected memory. Multiple users can run multiple programs each at the same time without interfering with each other or crashing the system.
- ✓ Very efficient virtual memory, so many programs can run with a modest amount of physical memory.
- ✓ Access controls and security. All users must be authenticated by a valid account and password to use the system at all. All files are owned by particular accounts. The owner can decide whether others have read or write access to his files.
- ✓ A rich set of small commands and utilities that do specific tasks well -- not cluttered up with lots of special options. Unix is a well-stocked toolbox, not a giant do-it-all Swiss Army Knife.
- ✓ Ability to string commands and utilities together in unlimited ways to accomplish more complicated tasks -- not limited to preconfigured combinations or menus, as in personal computer systems.
- ✓ A powerfully unified file system. Everything is a file: data, programs, and all physical devices. Entire file system appears as a single large tree of nested directories, regardless of how many different physical devices (disks) are included.
- ✓ A lean kernel that does the basics for you but doesn't get in the way when you try to do the unusual.
- ✓ Available on a wide variety of machines - the most truly portable operating system.
- ✓ Optimized for program development, and thus for the unusual circumstances that are the rule in research.

4.2 Disadvantages

- ✓ The traditional command line shell interface is user hostile -- designed for the programmer, not the casual user.
- ✓ Commands often have cryptic names and give very little response to tell the user what they are doing. Much use of special keyboard characters - little typos have unexpected results.
- ✓ To use Unix well, you need to understand some of the main design features. Its power comes from knowing how to make commands and programs interact with each other, not just from treating each as a fixed black box.
- ✓ Richness of utilities (over 400 standard ones) often overwhelms novices. Documentation is short on examples and tutorials to help you figure out how to use the many tools provided to accomplish various kinds of tasks.

5. How Unix OS Works

The architecture of UNIX is divided into three levels. On the outer crust reside the Application, Programs and other utilities. At the heart of UNIX, on the other hand, is the Kernel, which interacts with actual hardware in machine language. The stream lining of these two modes of communication is done by the middle layer called Shell.

Thus the three basic layers are:-

1. HARDWARE
2. KERNEL
3. SHELL

At the core is the physical component of the computer system i.e. the hardware and the central part of UNIX system is Kernel which has record of all the hardware and knows how to communicate with every piece of hardware. The shell is the command interpreter. The uppermost layer consists of users creating the application programs.

5.1 HARDWRE

The physical component of the computer system is called hardware. In a UNIX, physical components (hardware) are used like as CPU, ALU, main memory (RAM, ROM), I/O devices, external memory (magnetic tapes, CD's etc).

5.2 UNIX SHELL

The shell or the command interpreter is the mediator which interprets the commands we give and then conveys them to the Kernel which ultimately executes them. It provides the user interface to the Kernel. It is basically an expensive program running on the computer all the time and provides an interactive interface between the user and the computer functions.

There are three types of shells:-

1. Bourne Shell
2. C Shell
3. Korn Shell

5.2.1 Bourne Shell: - It is the original UNIX Shell written by Steve Bourne and John Mashey from AT & T's Bell laboratories. It is the standard shell and is available on all UNIX operating system.
It is known as "sh".

5.2.2 C Shell: - It was developed by Bill Joy from university of California. It does not have the programming capabilities of Bourne Shell but is better for interactivity. It is known as "csh".

5.2.3 Korn Shell: - It was developed by David Korn at the Bell labs. It has the programming capabilities of Bourne Shell and Interactive feature of C Shell. It is known as "ksh".

5.3 KERNEL

It is that part of the operating system that carries out the basic functions such as accessing files, allocating memory and handling communications. Its main function is to manage the resources of the computer's hardware such as CPU, memory, I/O devices and network communication.

It has a record of all the hardware and knows how to communicate with every piece of hardware. Each kernel is built for a specific computer and is not portable to another computer platform. The kernel is loaded into the main memory when the computer system starts up. It resides in the main memory in order to execute and direct activities inside the computer.

5.4 USERS

The human beings that use the computer system are called the users. The application programs and system programs are written by the user to interact and work with UNIX operating system. The programs like shell and editor that are the topmost layers interact with kernel by invoking well defined set of system calls. These various calls instruct the kernel to perform various functions for calling system program or application program.