title: Operating Systems and BSD
subtitle: OSs
minutes: 5

Operating Systems

Learning objectives

- Explain what operating systems are.
- Explain the history and advantages of Unix.

Operating Systems and Unix

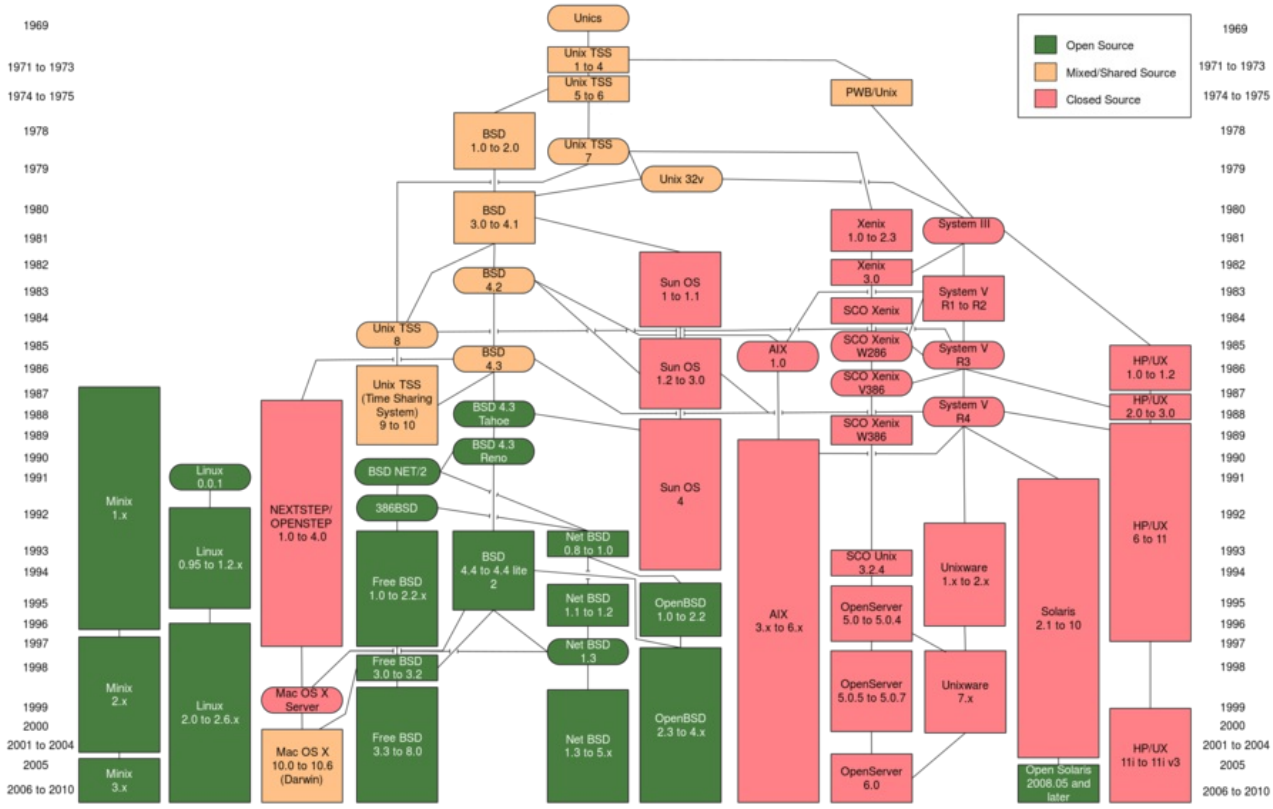
What is an Operating System?

An operating system is a suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

Unix

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since.

UNIX HISTORY
1969 - 2010



Unix v. Windows

Unix has several fundamental differences compared with Windows:

- More rigorous security
- Extremely powerful command-line tools
- Very stable
- Entirely different directory structure

Unix v. Apple OSX

OSX *is* Unix: A version called Darwin, based on [BSD (https://en.wikipedia.org/wiki/Berkeley_Software_Distribution)](https://en.wikipedia.org/wiki/Berkeley_Software_Distribution). It comes packaged with all the necessary tools:

- The full suite of command-line tools in the **Terminal**
- An X11() server for graphics
- Secure shell and secure copy for working over networks
- The C and FORTRAN compilers (in the Xcode toolset)

OSX is probably the best-designed commercial Unix variant for consumer use in operation today.

Unix v. Linux

Linux is a free and open source (https://en.wikipedia.org/wiki/Free_and_open-source_software) Unix-like operating system. It is the leading operating system on servers and other big iron systems such as mainframe computers and supercomputers, but is used on only around 1% of personal computers. Typically, Linux is packaged in a form known as *Linux distribution* such as the one in Ubuntu (http://www.ubuntu.com/about/about-ubuntu).

Key Components of Unix

(Adapted from Indiana University (https://kb.iu.edu/d/agat))

Unix has three main components

Kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the filestore and communications in response to system calls.

Shell

The shell is an interactive program that provides an interface between the user and the kernel. The shell interprets commands entered by the user or supplied by a shell script, and passes them to the kernel for execution.

As an illustration of the way that the shell and the kernel work together, suppose a user types `rm myfile` (which has the effect of removing the file *myfile*). The shell searches the filestore for the file containing the program `rm`, and then requests the kernel, through system calls, to execute the program `rm` on *myfile*. When the process `rm myfile` has finished running, the shell then returns the UNIX prompt `%` to the user, indicating that it is waiting for further commands.

We'll talk more about shells in a little bit.

File system

Unix and Unix-like operating systems employ a hierarchical (i.e., inverted tree) directory structure, with the root directory (/) at the top.

The standard file system has, among others, the following directories:

Directory	Description
/	The root directory, where the whole tree starts
/bin	Contains fundamental executables (i.e., binaries) generally used by all users on the system (e.g., chmod, cp, mv, grep, and tar)
/etc	Contains local configuration files, subdirectories containing configuration files for large software packages (e.g., the X11 window system)
/lib	Contains shared libraries needed to boot the system and run the commands in the root file system
/tmp	Local scratch space for storing temporary files, which may be deleted without notice
/usr/bin	The primary directory for most executables used by normal users on the system (e.g., emacs, make, scp, sftp, ssh, and yum)
usr/lib	Contains static and dynamic libraries, a few executables that usually are not invoked directly, and subdirectories for complex programs

We'll also be talking more about files + directories.