SYST13416
Introduction to Linux
Operating

The Essence of Unix

COMPUTERS

Objectives

- Discuss the background and philosophy of using computers
- Overview the basics of digital computer hardware (memory unit, control unit, ALU, Input/output unit, secondary storage)
- Discuss computer use

Computers

Computers then and now

The word computer used to mean a person who computes. Now, a computer is any device used to process information according to a well-defined procedure. Originally, information processing was almost exclusively related to arithmetical problems, but modern computers are used for many tasks unrelated to mathematics. Often the definition is restricted to devices whose primary purpose is information processing rather than being a part of a larger system such as a phone, microwave, or aircraft, and can be adapted for a variety of purposes by the user without physical modification. Mainframes, minicomputers, and personal computers meet this definition.

Today, many household devices, notably video game consoles, and extending to mobile telephones, video cassette recorders, PDA's and myriad other household, industrial, automotive, and other electronic devices, all contain computer-like circuitry. These computers inside other special-purpose devices are commonly referred to as "microcontrollers" or "embedded computers".

Digital Computers

While the technologies used in digital computers have changed dramatically since the first computers of the 1940s, most still use the von Neumann architecture proposed in the early 1940s by John von Neumann. Von Neumann's architecture describes a computer with four main sections: the Arithmetic and Logic Unit (ALU), the control circuitry, the memory, and the input and output devices (I/O). These parts are interconnected by a bundle of wires, a bus.

The ALU is the device that performs elementary operations such as arithmetic operations (addition, subtraction, and so on), logical operations (AND, OR, NOT),

and comparison operations (for example, comparing the contents of two "slots" for equality).

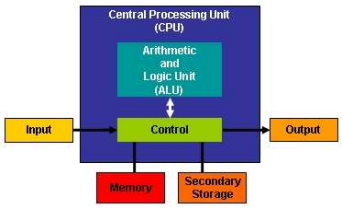


Figure 2: The figure illustrates the computer components that support these capabilities.

The control unit keeps track of which slot contains the current instruction that the computer is performing, telling the ALU what operation to perform and retrieving the information (from the memory) that it needs to perform it, and transfers the result back to the appropriate memory location. Once that occurs, the control unit goes to the next instruction. The I/O allows the computer to obtain information from the outside world, and send the results of its work back there. There is incredibly broad range of I/O devices (keyboards, monitors, disk drives, webcams, etc.).

Contemporary computers put the ALU and control unit into a single integrated circuit known as the Central Processing Unit (CPU). Typically, the computer's memory is located on a few small integrated circuits near the CPU. Some larger computers differ from the above model in one major respect - they have multiple CPUs and control units working simultaneously.

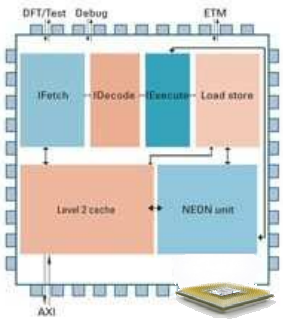
The functioning of a computer is therefore in principle quite straightforward. The computer fetches instructions and data from its memory. The instructions are executed, the results are stored, and the next instruction is fetched. This procedure repeats until the computer is turned off.

Hardware

All computers, from large supercomputers costing millions of dollars to smaller desktop personal computers, must perform a minimum set of functions and provide capability to

- Accept input
- Display output
- Store information in a logically consistent format (traditionally, binary)
- Perform arithmetic and logical operations on either the input or stored data
- Monitor, control, and direct overall operation and sequencing of the system

The physical components are collectively referred to as hardware.



Memory Unit

This unit stores information in a logically consistent format. Typically, both instructions and data are stored in memory, usually in separate and distinct areas. Each computer contains memory of two fundamental types: RAM and ROM.

- **Random Access Memory (RAM)** is usually volatile, which means that whatever is stored there is lost when the computer's power is turned off. Your programs and data are stored in RAM while you are using the computer. The size of a computer's RAM is usually specified in terms of how many bytes are available to the user. Personal computer memories currently consists of from 32 to 512 million bytes.
- **Read-only Memory (ROM)** contains fundamental instructions that cannot be lost or changed by the casual computer user. These instructions include those necessary for loading anything else into the machine when it is first turned on and any other instructions the manufacturer requires to be permanently accessible when the computer is turned on. ROM is non-volatile; its contents are not lost when the power goes off.

Control Unit

The control unit, **Central Processing Unit (CPU)**, directs and monitors the overall operation of the computer. It keeps track of where in memory the next instruction resides, issues the signals needed to both read data from and write data to other units in the system, and executes all instructions.

Arithmetic Logic Unit

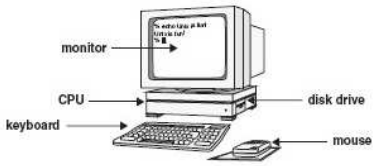
The **Arithmetic Logic Unit (ALU)** performs all the arithmetic and logic functions, such as addition, subtraction, comparison, and so forth, provided

Input/Output Unit

The **Input/Output Unit (I/O)** provides access to and from the computer. It is the interface to which peripheral devices such as keyboard, cathode array screens, and **printers** are attached. Input devices (**keyboard**, **mouse**, **scanner**) trigger instructions to be processed. The most common output device is the **video display terminal (VDT)**, also called **monitor**.

Secondary Storage

Because RAM memory in large quantities is still relatively expensive and volatile, it is not practical as a permanent storage area for programs and data. Secondary or auxiliary storage devices are used for this purpose. Although data have been stored on punched cards, paper tape, and other media in the past, virtually all secondary storage is now done on magnetic tape, magnetic disks, and optical storage media



Computer Use

Computers are **tools that assist** and ease cumbersome and seemingly impossible tasks. Not all tools are physical devices, some are instructions and methods, that instruct us how to accomplish a goal, such as solve mathematical problem. Many of these tasks require only a **human brain, some paper, and a pencil**. Without computers, many tasks would require countless hours of hand calculations.

In any case, **NO ONE should depend completely** on computers. NEVER blindly trust a computer's output. Always check your input. To check your computer software's results, use traditional pencil-and-paper analysis. Always remember that computers are incredibly useful tools that require **diligence** and **understanding**.

dil·i·gence
n.
1. Earnest and persistent application to an undertaking; steady effort; assiduity.
2. Attentive care; heedfulness.

SYST13416
Introduction to Linux Operating

The Essence of Unix

PROGRAMMING LANGUAGES

Objectives

- Overview the basics of computer software

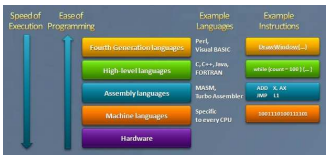
Programming Languages

The instructions discussed previously are not the rich instructions of a human language. A computer only has a limited number of well-defined, simple instructions.

Instructions are represented within the computer as numbers - the code for "copy" might be 001, for example. The particular instruction set that a specific computer supports is known as that computer's machine language.

In practice, people do not normally write the instructions for computers directly in machine language but rather use a "high level" programming language which is then translated into the machine language automatically by special computer programs (interpreters and compilers).

Some programming languages map very closely to the machine language, such as assembler (low level languages); at the other end, languages like Prolog are based on abstract principles far removed from the details of the machine's actual operation (high level languages).



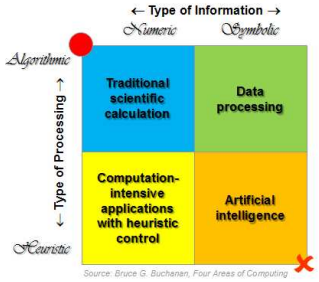
Levels of Programming

Computer Programs

Computer programs are simply large lists of instructions for the computer to execute, perhaps with tables of data.

Many computer programs contain millions of instructions, and many of those instructions are executed repeatedly. A typical modern PC can execute around 2-3 billion instructions per second.

Computers do not gain their extraordinary capabilities through the ability to execute complex instructions. Rather, they do millions of simple instructions arranged by clever people, "programmers."



Good programmers develop sets of instructions to do common tasks (for instance, draw a dot on screen) and then make those sets of instructions available to other programmers.

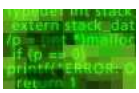
Nowadays, most computers appear to execute several programs at the same time. This is usually referred to as multi-tasking.

In reality, the CPU executes instructions from one program, then after a short period of time, it switches to a second program and executes some of its instructions. This small interval of time is often referred to as a time slice. This creates the illusion of multiple programs being executed simultaneously by sharing the CPU's time between the programs.

This is similar to how a movie is simply a rapid succession of still frames. The operating system is the program that usually controls this time sharing.

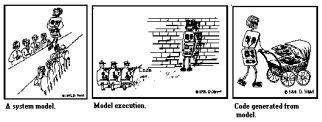
Contemporary computers put the ALU and control unit into a single integrated circuit known as the Central Processing Unit (CPU). Typically, the computer's memory is located on a few small integrated circuits near the CPU. Some larger computers differ from the above model in one major respect - they have multiple CPUs and control units working simultaneously.

The functioning of a computer is therefore in principle quite straightforward. The computer fetches instructions and data from its memory. The instructions are executed, the results are stored, and the next instruction is fetched. This procedure repeats until the computer is turned off.



Code

Code refers to a program written in a programming language. You will encounter several kinds of computer code.



Source Code→
→*Preprocessor*→
→**Modified Source Code**→
→*Compiler*→
→**Object Code**→
→*Linker*→
→**Executable Code**

```
# comparing strings
print ("Enter string1: \n");
$str1=<STDIN>;
chomp $str1;
print ("Enter string2: \n");
$str2=<STDIN>;
chomp $str2;
```

Pseudocode

A common tool in program design, pseudocode is an outline of the program written in abbreviated English. It is laid out to resemble the structure of the actual program.

Source Code

A computer program written in a high-level language is called source code. Source code is stored in an ASCII text file. (Entered by programmer with a text editor)

Assembly Code

A computer program written in assembly language is called, appropriately, assembly code.

Object Code

When source code is translated by a compiler, the resulting machine language is called object code.

Library Code

Some language, such as C, provide libraries of pre-compiled functions and routines. This library code can be linked to object code to make a complete program.

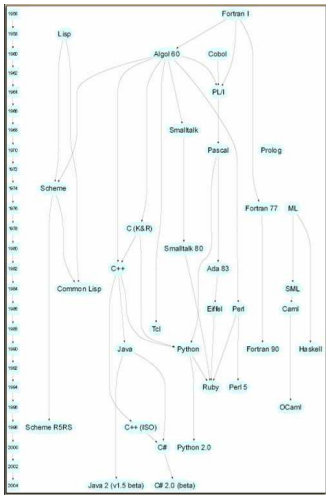
Executable Code

Binary code that can be run directly by the computer is called executable code. Depending on the language, it may consist of both object code and library code.

Scripting Languages

An interpreter is a program that executes other programs. The interpreter steps through the source code, translating and executing each instruction as it encounters it.

A Few Well-known High-level Languages



BASIC (Beginners All-purpose Symbolic Instruction Code)
A general programming language originally designed to be simple enough for the beginners to learn.

FORTRAN (Formula Translator)
A language designed for programming complex mathematical algorithms.

COBOL (Common Business-Oriented Language)
A language designed for business applications.

Pascal
A structured, general-purpose language designed primarily for teaching programming.

C
A structured, general-purpose language developed at Bell Laboratories. C offers both high-level and low-level features.

C++
Based on the C language, C++ offers object-oriented features not found in C. Also invented at Bell Laboratories.

C# (pronounced C sharp)
A language invented by Microsoft for developing applications on Microsoft .NET platform.

Java
An object-oriented language invented at Sun Microsystems. Java may be used to develop programs that run over the Internet, in a Web browser.

Javascript
Javascript can be used to write small programs that run in Web pages. Despite its name, Javascript is not related to Java.

Python
Python is a general purpose language created in the early 1990s. It has become popular in both business and academic applications.

Ruby
Ruby is a general purpose language that was created in the 1990s. It became popular for programs that run on Web servers.

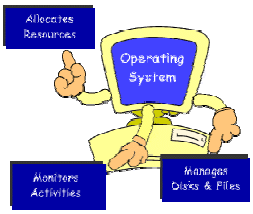
Visual Basic
A Microsoft programming language and development environment used to quickly create Windows-based applications (often used for proto-typing).

OPERATING SYSTEM

Objectives

- Illustrate features of operating system

Operating System



Concept

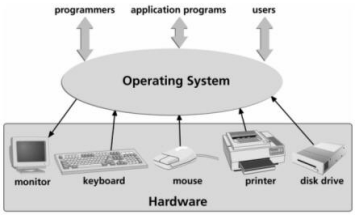
Two logical categories of computer programs are: applications software and system software.

Application software consists of programs written to perform particular tasks required by users.

System software is the collection of programs that must be readily available to any computer system for it to operate at all. In the early computer environments of the 1950s and 1960s, a user initially had to load the system software by hand to prepare the computer to do anything. This was done with rows of switches on a front panel. Those initial hand-entered commands were said to boot the computer, an expression derived from "pulling oneself up by the bootstraps". Today, the so-called bootstrap loader is internally contained in read-only memory (ROM) and is a permanent, automatically executed component of the computer's system software.

Collectively, the set of system programs used to operate and control a computer is called the operating system. Tasks handled by modern operating systems include memory allocation, input/output control, and secondary storage management.

Many operating systems handle very large programs, as well as multiple users concurrently, by dividing programs into segments or pages that are moved between the disk and memory as needed. Such operating systems create a virtual memory, which appears to be as large as necessary to handle any job, and a multi-user environment is produced that gives each user the impression that the computer and peripherals are his or hers alone. Many operating systems, including most windowed environments, permit each user to run multiple programs. Such operating systems are referred to as both multi-programming and multi-tasking systems.



What kinds are there?

Within the broad family of operating systems, there are generally four types, categorized based on the types of computers they control and the sort of applications they support. The broad categories are:

Real-time operating system (RTOS) - Real-time operating systems are used to control machinery, scientific instruments and industrial systems. An RTOS typically has very little user-interface capability, and no end-user utilities, since the system will be a "sealed box" when delivered for use. A very important part of an RTOS is managing the resources of the computer so that a particular operation executes in precisely the same amount of time every time it occurs. In a complex machine, having a part move more quickly just because system resources are available may be just as catastrophic as having it not move at all because the system is busy.

Single-user, single task - As the name implies, this operating system is designed to manage the computer so that one user can effectively do one thing at a time. The Palm OS for Palm handheld computers is a good example of a modern single-user, single-task operating system.

Single-user, multi-tasking - This is the type of operating system most people use on their desktop and laptop computers today. Microsoft's Windows and Apple's MacOS platforms are both examples of operating systems that will let a single user have several programs in operation at the same time. For example, it's entirely possible for a Windows user to be writing a note in a word processor while downloading a file from the Internet while printing the text of an e-mail message.

Multi-user - A multi-user operating system allows many different users to take advantage of the computer's resources simultaneously. The operating system must make sure that the requirements of the various users are balanced, and that each of the programs they are using has sufficient and separate resources so that a problem with one user doesn't affect the entire community of users. Unix, VMS and mainframe operating systems, such as MVS, are examples of multi-user. ..operating systems.

What is multiprocessing?

Multiprocessing - means that different processes can run simultaneously on multiple processors. If a system has eight processors, it can execute eight processes simultaneously. These processes may belong to different programs or to the same program. If different processes in the same program can run concurrently on different processors, the program is said to be multi-threaded.

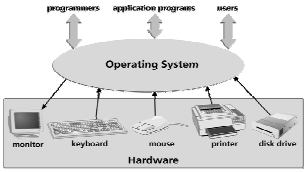
Multiprocessing has the advantage of increasing system-wide throughput and speeding up multi-process applications such as databases, servers and background processes. There are two types of multiprocessing architectures: Asymmetrical and symmetrical. Both architectures can use one of two designs for processor-memory association: loosely coupled or tightly coupled.

Asymmetrical architecture is one where one processor acts as a master to other slave processors. The master processor executes all kernel code while the slave processors execute user code delegated by the master.

Symmetrical architecture: all processors in the system function as peers and all processors have equal access to all resources such as memory and devices. All I/O and interrupts are available equally from any processor so that no single processor becomes a bottleneck when the system is executing I/O intensive loads.

Tightly Coupled design: all processors access a single pool of memory over a shared processor-memory bus. There is no message passing; processors communicate via physically shared memory and cross-processor interrupts. All processors have access to one copy of the shared region of memory. This design permits efficient execution of shared memory applications such as relational databases.

Loosely Coupled design: all processors have dedicated memory regions attached via a dedicated interconnect bus. Memory accesses occur over the dedicated bus at each processor. The system bus handles interprocessor communications and I/O. This architecture provides high performance for applications that do not need access to other processors' memory and isolated failures of dedicated memory regions.

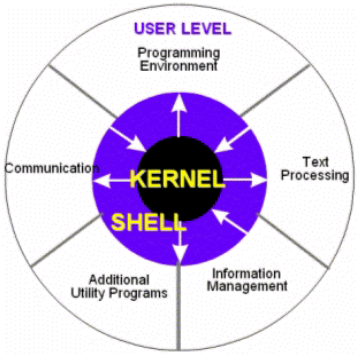


Operating System manages the hardware and software resources of the system as various programs compete for the attention of the central processing unit (CPU) and demand memory, storage and input/output (I/O) bandwidth for their own purposes

- In a desktop computer, 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.

Operating System provides a stable, consistent way for applications to deal with the hardware without having to know all the details of the hardware.

- 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.



Operating System Components

What is a kernel?

Kernel is the heart of the operating system where the real work is done. It decides how much RAM space is to be allocated to a program before it is loaded and executed. It ensures that only one file is printed on the printer at a time. It prevents an existing file on the disk from being accidentally overwritten by another file. It guarantees that when execution of a program given to the CPU for processing has been completed, the program relinquishes the CPU so that other programs can be executed. In other words, it manages the resources, which include all hardware components, including keyboard, display, memory, storage, modem, CPU, etc.

What is a shell?

Shell forms the application user's interface that protects the kernel from users. It provides a set of functions that can be used by the user and applications to access the kernel's services. A SHELL IS A LANGUAGE. Most users know the "click-and-point language of the desktop". BUT in that language the user is treated as a baby, asked to pick from what is presented to her. A shell, on the other hand, is an advanced way of communicating with the system, because it allows for conversation and taking initiative. Both partners are equal, so new ideas can be tested. The shell allows the user to handle a system in a flexible way. Don't you want to be treated as an adult?

Interacting with an operating system

Operating system is a program that allows you to interact with the computer—all of the software and hardware on your computer

With a command-line interface (CLI), you type a text command and the computer responds according to that command.

With a graphical user interface (GUI), you interact with the computer through a graphical interface with pictures and buttons by using the mouse and keyboard.

Two Major Classes: Unix and Windows

And they have a competitive history and future.

- **Unix** has been in use for more than four decades. Originally it rose from the ashes of a failed attempt in the early 1960s to develop **a reliable timesharing operating system**. A few survivors from Bell Labs did not give up and developed a system that provided **a work environment described as "of unusual simplicity, power, and elegance"**.
- Since the 1980's Unix's main competitor **Windows** has gained popularity due to the increasing power of micro-computers with Intel-compatible processors.
- Windows, at the time, was the only major OS designed for this type of processors.
- In recent years, however, a new version of Unix called **Linux**, also specifically developed for micro-computers, has emerged. It can be obtained for free and is therefore a lucrative choice for individuals and businesses.
- On the server front, Unix has been closing in on Microsoft's market share.
- In 1999, Linux scooted past Novell's Netware to become the No. 2 server operating system behind Windows NT.
- In 2001 the market share for the Linux operating system was 25 percent; other Unix flavors 12 percent.
- On the client front, Microsoft is currently dominating the operating system market with over 90% market share.
- Because of Microsoft's **aggressive marketing practices**, millions of users who have no idea what an operating system is have been using Windows operating systems given to them when they purchased their PCs.
- **Apple** adopts Linux , **Mac OS X** is a Unix-based operating system

Sheridan

SYST13416
Introduction to Linux Operating

The Essence of Unix

WHY LINUX?

Objectives

- Explain why you are learning about Unix
- Illustrate features of Linux/Unix operating system
- Illustrate advantages of using Linux operating system
- Explain the need to use command line interface
- Give examples of when to use GUI and when CLI

Why Linux?

- ***Openness***
  The source code is publicly available. Easily adapted to suit needs.
- ***Utilities***
  Multitude of programs available, both commercial and public domain.
- ***Portability***
  Virtually every kind of computer supports Unix.
- ***Multitasking***
  Users can run different programs simultaneously.
- ***Networking***
  Unix protocols form basis of the Internet
- ***Prevalence***
  Workstations help solve many complicated problems

UNIX is a **popular time−sharing** operating system originally intended for program development and document preparation, but later widely accepted for a number of implementations.

UNIX is **today's most ubiquitous multi−user** operating system, with no indication of any diminishment in the near future. Today, when a period of several years **represents the lifetime of many successful IT products**, UNIX is still considered **the most stable and the most secure** operating system on the market, four decades after its appearance. Of course, during 40 years of existence UNIX has changed a great deal, adapting to new requirements; it is hard to compare today's modern UNIX flavours with initial (now obsolete) UNIX versions.

In fact, these changes and adaptations are unique to the UNIX operating system; **no other operating system has so successfully evolved**, time and again, **to meet modern needs**. The concept and basic design of UNIX deserve the credit for this remarkable **longevity**, as they provide the necessary **flexibility** for the permanent changes required to make UNIX **suitable** for many new applications.

Source: Levi, Bozidar, UNIX administration :a comprehensive sourcebook for effective systems and network management, CRC Press, 2002.

- Linux is a multi-user, multi-tasking system.
- Linux is an "**open source**" system.
- Linux will run on both large and small computer systems.
- Linux can be used to control computer networks.
- Linux is available both as a **free** download from many Internet sources and as a **low-cost**, packaged "distribution".
- Linux supports an **effective, built-in security** system. This security system is designed, first of all, to protect the system itself. Second, and equally important, Linux security protects your data from unauthorized access.

A Very Brief History of Unix…

The 9 Unix Philosophy Tenets

1. Small is beautiful
2. Make each program do one thing well
3. Build a prototype as soon as possible
4. Choose portability over efficiency
5. Store numerical data in flat ASCII files
6. Use software leverage to your advantage
7. Use shell scripts to increase leverage and portability
8. Avoid captive user interfaces
9. Make every program a filter

Top 10 Lesser Tenets

1. Allow the user to tailor the environment
2. Make operating system kernels small and lightweight
3. Use lower case and keep it short
4. Save trees
5. Silence is golden
6. Think parallel
7. The sum of the parts is greater than the whole
8. Look for the 90 percent solution
9. Worse is better
10. Think hierarchically

1969 - Dennis Ritchie, Ken Thompson rewrite SPACE TRAVEL from the GE-645 to the DEC PDP-11/20 Unix is born

1970 - Bell offers minimal funding of Unix as an operating system if a text processor is developed: Unix is ported to a larger machine: PDP-11/20

1973 - Kernel rewritten in C: Unix released to universities, commercial firms, and the government for a nominal fee

1974 - Fourth edition of Unix marks a wide use of the operating system throughout Bell Labs

Thompson

1975 - Thompson takes a copy of the 6th edition to UC Berkeley and encourages development (birth of BSD - Berkeley Software Distribution).

1978 - Over 600 machines running Unix - primarily throughout Bell Labs and universities

1980 - Microsoft releases XENIX - Unix capabilities designed for microcomputers; Unix hits the desktop world

Ritchie

1982 - Unix System III released commercially by AT&T

1983 - Unix System V Release 1 - AT&T promises to maintain upward compatibility in future releases

1985 - Release 2 - incorporates UC Berkeley features, including vi

1987 - Release 3 - incorporates consistent approach to networking: Release 3.2 merges System V and XENIX

1990 - Release 4 - unifies various versions of Unix developed inside and outside AT&T: Berkeley, SunOS, XENIX, and AT&T Unix

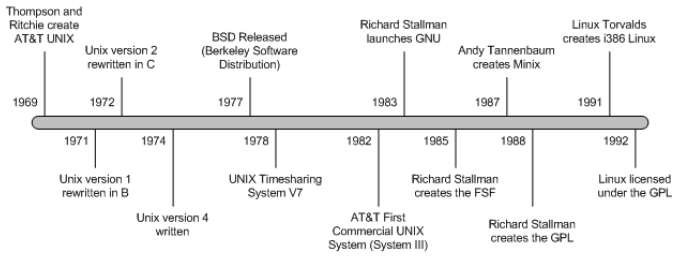
1991 - Linus Torvalds, as student at University of Helsinki, released Linux to the community of hackers on the Internet to work with and enhance it

Torvalds

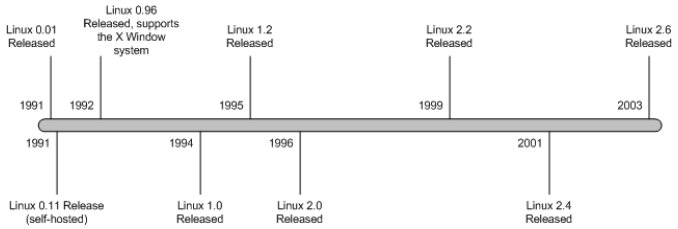
A VERY BRIEF HISTORY OF UNIX ...

X

Unix Timeline



Linux Timeline

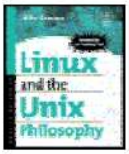


ADDITIONAL READING

Linux and the Unix Philosophy

By Mike Gancarz
Digital Press 2003
(220 pages)
ISBN: 1555582737

Unlike so many books that focus on how to use Linux, this text explores why Linux is a superior implementation of Unix's highly capable operating system.



Linux Philosophy

These ideas come from the UNIX philosophy. They serve as examples in the industry and are adopted as international standards:

- **Rule of Modularity**: Write simple parts connected by clean interfaces.
- **Rule of Clarity**: Clarity is better than cleverness.
- **Rule of Composition**: Design programs to be connected to other programs.
- **Rule of Separation**: Separate policy from mechanism; separate interfaces from engines.
- **Rule of Simplicity**: Design for simplicity; add complexity only where you must.
- **Rule of Parsimony**: Write a big program only when it is clear by demonstration that nothing else will do.
- **Rule of Transparency**: Design for visibility to make inspection and debugging easier.
- **Rule of Robustness:** Robustness is the child of transparency and simplicity.
- **Rule of Representation:** Fold knowledge into data so program logic can be stupid and robust.
- **Rule of Least Surprise:** In interface design, always do the least surprising thing.
- **Rule of Silence:** When a program has nothing surprising to say, it should say nothing.
- **Rule of Repair:** When you must fail, fail noisily and as soon as possible.
- **Rule of Economy:** Programmer time is expensive; conserve it in preference to machine time.
- **Rule of Generation:** Avoid hand-hacking; write programs to write programs when you can.
- **Rule of Optimization:** Prototype before polishing. Get it working before you optimize it.
- **Rule of Diversity:** Distrust all claims for "one true way".
- **Rule of Extensibility:** Design for the future, because it will be here sooner than you think.

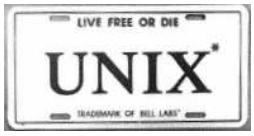
If you're new to Unix, **these principles are worth some meditation**. Software-engineering texts recommend most of them; but *most other operating systems lack the right tools and traditions to turn them into practice*, so most programmers can't apply them with any consistency. They come to accept blunt tools, bad designs, overwork, and bloated code as normal - and then wonder what Unix fans are so annoyed about.

To do the Unix philosophy right, **you have to be loyal to excellence**. You have to believe that software design is a craft worth all the intelligence, creativity, and passion you can muster. Otherwise you won't look past the easy, stereotyped ways of approaching design and implementation; you'll rush into coding when you should be thinking. You'll carelessly complicate when you should be relentlessly simplifying - and then you'll wonder why your code bloats and debugging is so hard.

To do the Unix philosophy right, **you have to value your own time enough never to waste it**. If someone has already solved a problem once, don't let pride or politics suck you into solving it a second time rather than re-using. And never work harder than you have to; work smarter instead, and save the extra effort for when you need it. Lean on your tools and automate everything you can.

To do the Unix philosophy right, **you need to have the right attitude.** You need to care. You need to play. You need to be willing to explore.

Linux Components



AIX	IBM's Advanced Interactive Executive OS
BSD	University of California's Berkeley Software Distribution
HP-UX	Hewlett-Packard's Unix
IRIX	Silicon Graphics's Unix
Linux	Linus Torvalds's Unix—very popular among personal computer users
Mach	Carnegie Mellon's version—adopted by OSF/1
OSF/1	Open Software Foundation's attempt to standardize Unix
Solaris	Sun Microsystems's current and very powerful version of Unix, based on SVR4
SunOS	Sun Microsystems's version of Unix that is being replaced by Solaris
System V	AT&T's version of Unix, also called System V Release 4 or SVR4
XENIX	Microsoft's Unix—also called Santa Cruz Operation (SCO) Unix. UnixWare is also owned by SCO

Kernel

- the heart or control section of the Unix system. It is the task manager and data storage manager for the processor (computer).
- contains most of the hardware-dependent operating system code.
- Much like the processor's inner-workings, the function of the kernel is invisible to the user.

Shell

- the window into the system. It acts as an interface between the user and the Unix system kernel.
- also acts as a command line interpreter. It interprets typed requests (commands), and then spawns and executes them.
- **Bourne shell (/sbin/sh)** written by Steve Bourne at Bell Laboratories;
- **C shell (/usr/bin/csh)** taken from the BSD system; provides command history, job control and command line editing capabilities; tailored for a C programming environment
- **Korn shell (/usr/bin/ksh)** written by David Korn at Bell Laboratories; offers a superset of the features of the Bourne shell, including command history, command line editing, and greatly enhanced programming features
- **job control shell (/sbin/jsh)** adds job control features to the Bourne shell

Language

- a set of rules and conventions that tell the computer how to perform certain operations. There are many standard programming languages available (Cobol, Fortran77, C language, etc.)

Utility

- a set of instructions that assist in the operations of the computer and the upkeep of the OS

File System

- the method that the Unix system uses for organizing and storing your information on the system.

SYST13416
Introduction to Linux Operating

The Essence of Unix

OPEN SOURCE AND STANDARDS

Objectives

- The open source definition
- International standards

The Open Source Definition

Open source doesn't just mean access to the source code. The distribution terms of open source software must comply with the following criteria:

1. Free Redistribution

- The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

- The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

- The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

- The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

- ***Internet Resources***
  http://opensource.org

- ***Openness***
  The source code is publicly available. Easily adapted to suit needs.
- ***Utilities***
  Multitude of programs available, both commercial and public domain.
- ***Portability***
  Virtually every kind of computer supports Unix.
- ***Multitasking***
  Users can run different programs simultaneously.
- ***Networking***
  Unix protocols form basis of the Internet
- ***Prevalence***
  Workstations help solve many complicated problems

5. No Discrimination Against Persons or Groups

- The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavour

- The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

- The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

- The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

- The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

- No provision of the license may be predicated on any individual technology or style of interface.

GNU
http://www.gnu.org/

The GNU Project was launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU system.

The Free Software Definition

- Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer.
- Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it means that the program's users have the four essential freedoms:
  - The freedom to run the program, for any purpose (freedom 0).
  - The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.
  - The freedom to redistribute copies so you can help your neighbor (freedom 2).
  - The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.
- **A program is free software if users have all of these freedoms**. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission to do so.

Standards

- Standards set the basic rules for quality systems--from conception to implementation--whether product or service. They are set of rules for manufacturing a product or delivering a service. They should ensure that a supplier has the capability to produce the required goods and services, showing him how to proceed to make sure what he delivers fully meets customer expectations.

Quality Standards:
"employees will show *a smile 2.75 inches in length at all times"*
- consistency**,** predictability, count-ability, and accountability
- Quality is defined by ISO standards.

- **Cause and Effect**
  There are four major categories for changing the outcome:
  - **People**: the individual effort
  - **Materials**: factors of production
  - **Methods**: methods of production
  - **Machines**: offer change in methods, such as automation but increase complexity (more things can go wrong).

Universal Understanding

- Universal understanding means that things are the **same** or **consistent** in its look and feel and we use consistent concepts
- **Example**: financial statements in every organization are consistent and look the same. A Balance Sheet in manufacturing is same as in Insurance.

Portable Operating System Interface for Unix (POSIX)

- is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system, although the standard can apply to any operating system

IEEE Std 1003.1,2004 Edition

- The core of the Single UNIX Specification, Version 3 is also IEEE Std 1003.1. The latest 2004 edition incorporates IEEE Std 1003.1-2001 and two technical corrigendum. IEEE Std 1003.1-2001 is a major revision and incorporates IEEE Std 1003.1-1990 (POSIX.1) and its subsequent amendments, and IEEE Std 1003.2-1992 (POSIX.2) and its subsequent amendments, combined with the core volumes of the Single UNIX Specification, Version 2. It is technically identical to The Open Group, Base Specifications, Issue 6; they are one and the same documents, the front cover having both designations. The final draft achieved 98% approval by the IEEE ballot group and was officially approved by the IEEE-SA Standards Board on December 6th 2001.
- During 2002, Technical Corrigendum 1 was developed. It was approved by the IEEE-SA Standards Board on December 11th 2002. In 2004 , IEEE Std 1003.1-2001/Cor 2-2004 was approved, and IEEE Std 1003.1 is officially designated as including IEEE Std 1003.1-2001, IEEE Std 1003.1-2001/Cor 1-2002. and IEEE Std 1003.1-2001/Cor 2-2004.

International Standards

International Organization for Standardization

http://www.iso.org/iso/home.htm

CSA STANDARDS

Canadian Standards Association

http://www.csa.ca/cm/ca/en/home

Open Systems Interconnection (OSI)

- Collective work of
  - ISO is a voluntary, non-treaty organization since 1946, with 89 member countries, creates world-wide standards
  - IEC coordinates activities on Country level—Telecomms (voice+data)
  - Local standards organizations of over 89 countries, including ANSI (US), BSI(Great Britain), AFNOR(France), DIN(Germany), CSA(Canada
  - ANSI works with EIA, trade group of manufacturers, focuses on hardware, 400 standards and publications
  - IEEE is the largest professional organization in the world, ANSI-accredited standards, development organization, defines LAN standards—Project 802
  - ITU is agency of the United Nations, inter-country communications
  - ITU-TSS makes technical recommendations to ITU on telephone, telegraph, and data communication interfaces (formerly CCITT)

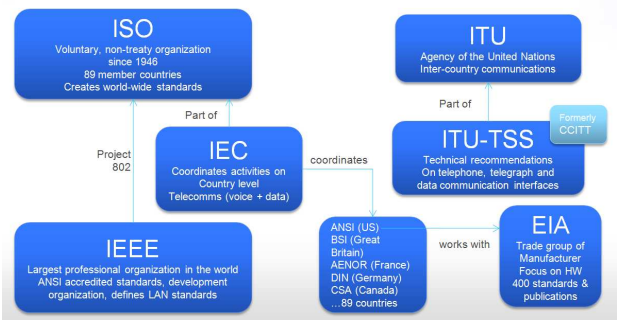


Table of Contents