Linux History

# Unix History

See Wikipedia article: History of Unix: (<https://en.wikipedia.org/wiki/History_of_Unix>)

* Unix was originally written in 1969 to run on a PDP-7 minicomputer.
  + Written in assembly language.
  + Written at Bell Labs
  + An alternative to a time-sharing system for a mainframe, called Multics.
  + Mostly written by Ken Thompson and Dennis Ritchie
  + [History of Unix - Wikipedia](https://en.wikipedia.org/wiki/History_of_Unix#/media/File:Ken_n_dennis.jpg)
  + Moved to PDP-11/20. Used for text formatting. Still written in assembly language.
* In 1972, Unix was re-written in C, a high-level language created by Dennis Ritchie for this purpose. This was revolutionary; operating systems were always written in assembly so that they would be fast. By writing in a high-level language, it was (somewhat) portable, and the language was designed to be fast.
* The long-time early standard for C was described in the book “The C Programming Language” by Brian Kernighan and Dennis Ritchie.
  + [Brian Kernighan - Wikipedia](https://en.wikipedia.org/wiki/Brian_Kernighan#/media/File:Brian_Kernighan_in_2012_at_Bell_Labs_1.jpg)

# Unix Philosophy

Key Point: (<https://en.wikipedia.org/wiki/Unix_philosophy>)

Described in 1978 as:

* Make each program do one thing well. To do a new job, build a fresh program rather than complicate old programs by adding new “features”
* Expect the output of every program to become the input to another, as yet unknown, program. Don’t clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don’t insist on interactive input.
* Design and build software, even operating systems, to be tried early, ideally within weeks. Don’t hesitate to throw away the clumsy part and rebuild them.
* Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of then out after you’ve finished using them.

Later summarized as:

* Write programs that do one thing and do it well.
* Write programs to work together
* **Write programs to handle text streams, because that is the universal interface**.

Essentially, Unix programs were small programs that did one thing well, then you could **JOIN** them together (use the output of one program as the input to another) to accomplish what it was you wanted to do.

If you don’t like the command line, this might change your mind. Rather than writing a new program, you take a bunch pre-existing utility programs and join the together to do what you want. And, then, you can put these into a script to do what you want to do! It makes your job as a system administrator MUCH easier.

# GNU

Richard Stallman, famous for writing Emacs (editor war: emacs vs vi), want to create a completely free version of Unix, unencumbered by licensing agreements. So, he started the GNU program (Gnu’s Not Unix) to build such an operating system. This project re-wrote many of the Unix utilities and used the General Public License (GPL) for them. Stallman also wrote a C compiler (GCC) to compile the programs. See Wikipedia article Gnu: (<https://en.wikipedia.org/wiki/GNU>)

Despite writing many, many utilities, the Gnu Project had (has?) one major problem. They have never finished their operating system kernel, the Hurd. Gnu utilities could be run on many operating systems, though, and are still widely used.

# Minix

A professor in Amsterdam, Andrew S Tanenbaum, famous at the time having written an excellent textbook on Computer Networking, wrote an operating system textbook and wrote an operating system to demonstrate the principles in the textbook. This was called Minix (mini-Unix). It was good to play with, and the source code was widely available, but could only be used for educational purposes.

Picture of Tanenbaum:

<https://en.wikipedia.org/wiki/Andrew_S._Tanenbaum#/media/File:AndrewTanenbaum2.png>

# Linux

See Wikipedia article: History of Linux (<https://en.wikipedia.org/wiki/History_of_Linux>)

A student at the University of Helsinki, Linus Torvalds decided to write an a new operating system kernel. He liked Minix, but didn’t like the licensing requirements. So, he wrote his own kernel. By combining it with tools from the Gnu project, he was able to put together an operating system.

Picture of Linus: (Not Brendan Frasier) <https://en.wikipedia.org/wiki/History_of_Linux#/media/File:Linus_Torvalds.jpeg>

# Key Linux Websites

Kernel.org: Main site for the Linux kernel. Can download the source code from here. Can also browse source code via Git repository. Go to git.kernel.org, look for Torvalds.

KernelNewbies.org: A good place if you want to learn about the kernel. Usually has a page that gives a good summary of what’s new in the latest release (or older releases)

Lkml.org: Discussions on kernel development are done over email. LKML = Linux Kernel Mailing List. It’s an archive of the mailing list. There are TONS of email messages every day.

LWN.net: Linux Weekly News. Linux news stories. Paid subscribers get access to news articles sooner than cheapos like me.

Unix.stackexchange.com: Stack Overflow.

<https://distrowatch.com>: A list of the popular Linux distributions

# Difference between Kernel and Operating System / Distributions

Kernel is the basic part of the OS: it looks after scheduling tasks, talking to devices, memory management, etc. OS is everything: Kernel, libraries, utilities, etc.

People don’t’ typically build their own kernel. They usually use a distribution, which includes the kernel, utilities, applications, etc. Distributions look after compiling everything and making sure it all works together.

Typical distributions:

* Debian (volunteer drive, very risk-averse, slow to incorporate new versions)
* Linux Mint (based on Debian, past colleagues have told me they really like it)
* **Ubuntu (builds on Debian, sponsored by Canonical, release every 6 months)**
* Fedora (another volunteer distribution, sponsored by Red Hat)
* Red Hat Enterprise Linux (RHEL) (paid version of Fedora from Red Hat Software)
* CentOS (free version RHEL)
* SUSE Linux (from SUSE)
* Open SUSE (free version based on SUSE)

In this class, we will use Ubuntu distribution.

# Our Linux Environment

Our Linux environment, for the most part, will be installed on our local machines as a virtual machine.

We will set up the Linux VM with 3 initial accounts (1 root, 2 user accounts).

|  |  |  |
| --- | --- | --- |
| Account Type | Username | Password |
| Root | insacc | Polytech123! |
| User | cstUser1 | cstuserpass |
| User | cstUser2 | cstuserpass |