Lecture on UNIX-like operating systems

Daniel Bosk[1]

Department of Information Technology and Media (ITM),
Mid Sweden University, Sundsvall.

unix.tex 537 2012-12-28 00:06:27Z danbos

[1] This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported license. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/3.0/.

Overview

Mittuniversitetet
MID SWEDEN UNIVERSITY

Overview

Development timeline
The 1960s

1965 Multiplexed Infomation and Computing Service (MULTICS) was a joint effort between MIT, Bell Labs and GE to

> *"develop a convenient, interactive, useable computer system that could support many users." [Lab02a]*

1969
- Bell Labs withdrew from the project, but Ken Thompson, Dennis Ritchie, Douglas McIllroy, and J. F. Ossanna continued on their own.
- Started to write the system on a PDP-7, at first simply as a file system.
- The system then got a shell, an editor, and an assembler. [Lab02a]

Development timeline

The 1960s

1965 Multiplexed Infomation and Computing Service (MULTICS) was a joint effort between MIT, Bell Labs and GE to

> *"develop a convenient, interactive, useable computer system that could support many users."* [Lab02a]

1969
- Bell Labs withdrew from the project, but Ken Thompson, Dennis Ritchie, Douglas McIllroy, and J. F. Ossanna continued on their own.
- Started to write the system on a PDP-7, at first simply as a file system.
- The system then got a shell, an editor, and an assembler. [Lab02a]

Development timeline

The 1970s

1970
- Brian Kernighan suggests the name UNIX.
- They port the current code to a PDP-11.
- Focused for use in text-processing, patent applications for Bell Labs.

1971 Ritchie improved Thompson's B programming language into the C programming language.

1972 Thompson started rewriting UNIX in C. (And continuous improvement of C.) [Lab02a]

Development timeline
The 1970s

1970	• Brian Kernighan suggests the name UNIX. • They port the current code to a PDP-11. • Focused for use in text-processing, patent applications for Bell Labs.
1971	Ritchie improved Thompson's B programming language into the C programming language.
1972	Thompson started rewriting UNIX in C. (And continuous improvement of C.) [Lab02a]

Mittuniversitetet
MID SWEDEN UNIVERSITY

Development timeline

The 1970s

1970
- Brian Kernighan suggests the name UNIX.
- They port the current code to a PDP-11.
- Focused for use in text-processing, patent applications for Bell Labs.

1971 Ritchie improved Thompson's B programming language into the C programming language.

1972 Thompson started rewriting UNIX in C. (And continuous improvement of C.) [Lab02a]

Development timeline

The 1970s, continued

1973

- UNIX completely rewritten in C.
- Thompson added McIlroy's concept of pipes. With this came the UNIX philosophy:

  "Write programs that do one thing and do it well. Write programs to work together. Write programs that handle text streams, because that is a universal interface." [Lab02a]

- Ritchie took initiative to manual pages, McIlroy soon took over and is the mind behind the layout of manual pages. [Lab02a]

Development timeline

The 1970s, continued

1973

- UNIX completely rewritten in C.
- Thompson added McIlroy's concept of pipes. With this came the UNIX philosophy:

  "Write programs that do one thing and do it well. Write programs to work together. Write programs that handle text streams, because that is a universal interface." [Lab02a]

- Ritchie took initiative to manual pages, McIlroy soon took over and is the mind behind the layout of manual pages. [Lab02a]

Development timeline

The 1970s, continued

1973

- UNIX completely rewritten in C.
- Thompson added McIlroy's concept of pipes. With this came the UNIX philosophy:

  "Write programs that do one thing and do it well. Write programs to work together. Write programs that handle text streams, because that is a universal interface." *[Lab02a]*

- Ritchie took initiative to manual pages, McIlroy soon took over and is the mind behind the layout of manual pages. [Lab02a]

Development timeline

The 1970s, continued

1975	Thompson is visiting professor at University of California-Berkeley (UCB). While there he developed version 6 of UNIX.
1978	Professors at Berkeley continued the enhancement of UNIX and distributed their work as Berkeley Software Distribution (BSD). [Lab02a]

Development timeline

The 1970s, continued

1975	Thompson is visiting professor at University of California-Berkeley (UCB). While there he developed version 6 of UNIX.
1978	Professors at Berkeley continued the enhancement of UNIX and distributed their work as Berkeley Software Distribution (BSD). [Lab02a]

Development timeline
The 1980s

1983	• Sockets API added to BSD (4.2BSD), i.e. TCP/IP made easily available. • David Korn develops the Korn Shell scripting language. • Bjarne Stroustrup develops the first version of C++.
1984	AT&T, owner of Bell Labs, started selling UNIX-licenses to companies. [Lab02a]

Development timeline
The 1980s

1983
- Sockets API added to BSD (4.2BSD), i.e. TCP/IP made easily available.
- David Korn develops the Korn Shell scripting language.
- Bjarne Stroustrup develops the first version of C++.

1984 AT&T, owner of Bell Labs, started selling UNIX-licenses to companies. [Lab02a]

Development timeline
Present

To this day, UNIX-like operating systems operate “most large Internet servers, businesses and universities, and a major part of academic and industrial research in operating systems is based on UNIX” [Lab02a].

Evolution of UNIX

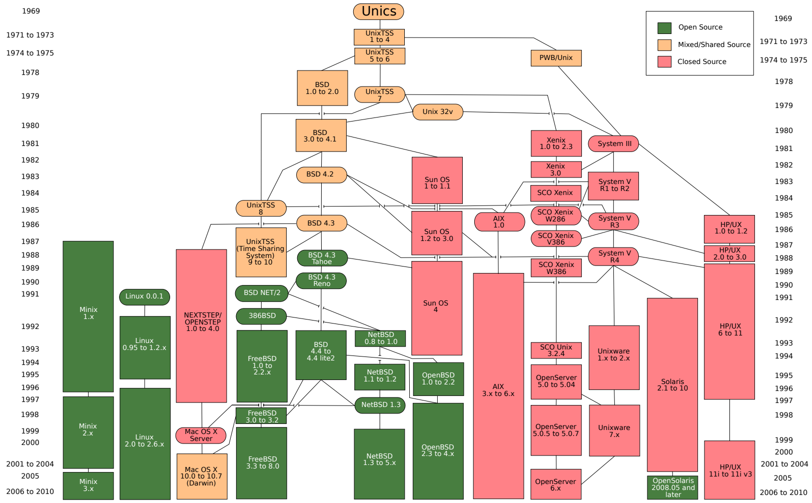

Image: `https://en.wikipedia.org/wiki/File:Unix_history-simple.svg`.
For details see `http://www.levenez.com/unix/`.

Overview

Architectural overview

Layered approach:

1. hardware,
2. monolithic kernel (drivers, system calls),
3. shell,
4. tools and application programs, and
5. users. [?, See]Figure 2.11, page 60]Silberschatz2005osc

Features:

- time-sharing,
- portability [Lab02b], and
- *everything is a file.*

Architectural overview

Layered approach:

1. hardware,
2. monolithic kernel (drivers, system calls),
3. shell,
4. tools and application programs, and
5. users. [?, See]Figure 2.11, page 60]Silberschatz2005osc

Features:

- time-sharing,
- portability [Lab02b], and
- *everything is a file*.

Architectural overview

Different types of kernels

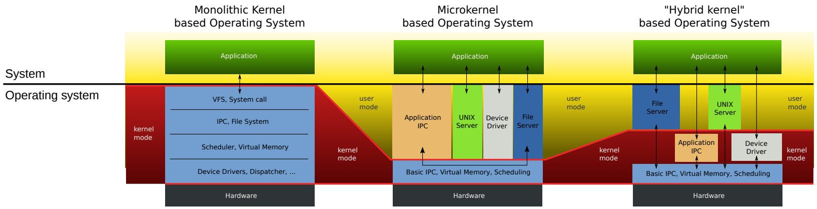
Monolithic Kernel based Operating System
Microkernel based Operating System
"Hybrid kernel" based Operating System
System
Operating system
Application
VFS, System call
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
Hardware
kernel mode
user mode
Application IPC
UNIX Server
Device Driver
File Server
Basic IPC, Virtual Memory, Scheduling

Image: `https://en.wikipedia.org/wiki/File:OS-structure2.svg`.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Architectural overview

Kernels of different UNIX-like and UNIX-based systems

OpenBSD	uses a monolithic kernel and the layered structure of the classical UNIX.
FreeBSD	uses a monolithic kernel but has added kernel modules.
Linux	uses a monolithic kernel with loadable kernel modules.
MacOS X	uses a hybrid between monolithic kernel and microkernel.

During your advanced study assignment you will go more in-depth on this subject.

Architectural overview

Kernels of different UNIX-like and UNIX-based systems

OpenBSD	uses a monolithic kernel and the layered structure of the classical UNIX.
FreeBSD	uses a monolithic kernel but has added kernel modules.
Linux	uses a monolithic kernel with loadable kernel modules.
MacOS X	uses a hybrid between monolithic kernel and microkernel.

During your advanced study assignment you will go more in-depth on this subject.

Architectural overview

Kernels of different UNIX-like and UNIX-based systems

OpenBSD	uses a monolithic kernel and the layered structure of the classical UNIX.
FreeBSD	uses a monolithic kernel but has added kernel modules.
Linux	uses a monolithic kernel with loadable kernel modules.
MacOS X	uses a hybrid between monolithic kernel and microkernel.

During your advanced study assignment you will go more in-depth on this subject.

Architectural overview

Kernels of different UNIX-like and UNIX-based systems

OpenBSD	uses a monolithic kernel and the layered structure of the classical UNIX.
FreeBSD	uses a monolithic kernel but has added kernel modules.
Linux	uses a monolithic kernel with loadable kernel modules.
MacOS X	uses a hybrid between monolithic kernel and microkernel.

During your advanced study assignment you will go more in-depth on this subject.

Architectural overview

Kernels of different UNIX-like and UNIX-based systems

OpenBSD	uses a monolithic kernel and the layered structure of the classical UNIX.
FreeBSD	uses a monolithic kernel but has added kernel modules.
Linux	uses a monolithic kernel with loadable kernel modules.
MacOS X	uses a hybrid between monolithic kernel and microkernel.

During your advanced study assignment you will go more in-depth on this subject.

UNIX File System Structure

The UNIX design is to have everything represented as a file in the file system.

/	The root directory.
/bin	Fundamental user utilities.
/dev	Device files (refers to actual physical devices).
/etc	Configuration files.
/home	User home directories.
/sbin	System programs and administration utilities fundamental to the system.
/usr	The majority of user utilities.
/var	Data files used by system programs, e.g. logs.

For further details see hier(7) [bsd, lin].

UNIX File System Structure

The UNIX design is to have everything represented as a file in the file system.

/	The root directory.
/bin	Fundamental user utilities.
/dev	Device files (refers to actual physical devices).
/etc	Configuration files.
/home	User home directories.
/sbin	System programs and administration utilities fundamental to the system.
/usr	The majority of user utilities.
/var	Data files used by system programs, e.g. logs.

For further details see hier(7) [bsd, lin].

UNIX File System Structure, continued

/usr/bin	Common utilities, programming tools, and applications.
/usr/include	Standard C include files.
/usr/X11R6	Files required for the X window system.
/usr/lib	System libraries used by programs in /usr/bin.
/usr/src	The source code for the system.

For further details see man-page hier(7) [bsd, lin].

Overview

UNIX Shell

- The simplistic and modular design of UNIX makes many different shells available, e.g.
  - Korn Shell (ksh),
  - Bourne Shell (sh),
  - Bourne Again Shell (bash), and
  - the X window system (X11R6).
- The shell interprets commands from the user and executes them.
  - The UNIX design of the shell is to implement all commands as separate programs – *which does one thing and does it well.*
  - This way commands can easily be added or removed.
  - The programs are located in /bin, /sbin, /usr/bin, etc.
  - Standard shells are located in /bin.

Mittuniversitetet
MID SWEDEN UNIVERSITY

UNIX Shell

- The simplistic and modular design of UNIX makes many different shells available, e.g.
  - Korn Shell (ksh),
  - Bourne Shell (sh),
  - Bourne Again Shell (bash), and
  - the X window system (X11R6).
- The shell interprets commands from the user and executes them.
  - The UNIX design of the shell is to implement all commands as separate programs – *which does one thing and does it well*.
  - This way commands can easily be added or removed.
  - The programs are located in /bin, /sbin, /usr/bin, etc.
  - Standard shells are located in /bin.

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

$>$	redirects stdout to a named file.
$<$	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

>	redirects stdout to a named file.
<	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

>	redirects stdout to a named file.
<	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

$>$	redirects stdout to a named file.
$<$	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

$>$	redirects stdout to a named file.
$<$	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

$>$	redirects stdout to a named file.
$<$	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Bourne Shell

- Environment variables accessible from shell and programs.
- Looks for simple commands in directories named in PATH environment variable.
- The two special and always open files: stdin and stdout.

stdin	input from e.g. keyboard connected to terminal.
stdout	output from process to e.g. display.

- Redirections:

>	redirects stdout to a named file.
<	reads contents of file as stdin.

- The pipe: redirects stdout of one process to stdin of another.

Shell Scripting

- Just shell commands, redirections, pipes etc. written to a file.
- Thanks to the UNIX design, there is no difference reading from stdin with stdin being the keyboard and stdin being redirected from a file.
- The file is hence read by the shell process and executed.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Shell Scripting

- Just shell commands, redirections, pipes etc. written to a file.
- Thanks to the UNIX design, there is no difference reading from stdin with stdin being the keyboard and stdin being redirected from a file.
- The file is hence read by the shell process and executed.

Shell Scripting

- Just shell commands, redirections, pipes etc. written to a file.
- Thanks to the UNIX design, there is no difference reading from stdin with stdin being the keyboard and stdin being redirected from a file.
- The file is hence read by the shell process and executed.

References I

- *OpenBSD Reference Manual.*
  OpenBSD 5.0.
- Bell Labs.
  The creation of the UNIX operating system, 2002.
- Bell Labs.
  An overview of the UNIX operating system, 2002.
- *Linux Programmer's Manual.*