

Plan 9 from Bell Labs

Unix, Only Moreso

Adam Frick

2019-04-04

History

Unics: Designed in a distant era

- ▶ Ken Thompson and friends started to simplify the original Multics system in 1969
- ▶ The known Unix ecosystem developed as needs grew
 - ▶ *Original inclusions: an editor (`ed`), PDP-7 assembler, and shell*
 - ▶ *Word processing (`roff`), Thompson shell, PDP-11 port (1971)*
 - ▶ *Rewritten in C (Version 4, 1973)*
 - ▶ *Bourne Shell (`sh`) (Version 7, 1977)*
- ▶ Unix released to educational markets in 1975 and commercially in 1982

Unics: Designed in a distant era

- ▶ It is the 1980's.
- ▶ Unix entrenched in a text- and keyboard-centric user interface
- ▶ Networking and graphics incohesive to original Unix design
- ▶ Computers were becoming cheaper and more sophisticated
- ▶ Ken Thompson and friends were ready to start fresh
- ▶ What will change this time around?

Design of Plan 9

9P: a universal communication protocol

- ▶ Client-server model (server responds to client's requests)
- ▶ Used for local and remote services
- ▶ Byte-oriented, little-endian
- ▶ 9P file servers for communication of programs and hardware

The file system: a universal communication interface

- ▶ Files are exposed by hardware to invoke behavior
- ▶ Files are exposed by programs to send/receive messages
- ▶ I/O achieved with regular reads and writes of text streams
- ▶ Sharing files means sharing data and resources

8 1/2: an unsophisticated windowing system

- ▶ A stacking window manager - everything is a window
- ▶ Windows are unaware of surrounding environment by design of the OS
- ▶ Files for graphics, tty, etc. are served for programs to interact with
- ▶ Mouse alone is used to interact with UI

The mouse: for everything other than typing

- ▶ Closely integrated with the window system
- ▶ Each button has a general purpose
 - ▶ *Left button: selecting text*
 - ▶ *Middle button: operations within a window*
 - ▶ *Right button: operations with windows*
- ▶ Buttons can be overridden with a graphical program in a conventional way

Private name spaces: processes have a unique view of the file system

- ▶ The window system creates a new name space for each terminal
- ▶ Files are created and brought into familiar places
- ▶ For the console I/O file, `/dev/cons`
 - ▶ *It is privately "mounted" (via `bind`) to `/dev/cons`*
 - ▶ *Consequently, programs always expect to read/write text to `/dev/cons`*

Feature orthogonality: measure twice, awk -F once

- ▶ Plan 9 takes the Unix philosophy without compromise
- ▶ Common actions need not be rewritten in every core utility:
 - ▶ *Walking through a file tree (`--recursive` is omitted)*
 - ▶ *Formatting a field-separated list into N columns (use `mc`)*
- ▶ For the sake of size and simplicity, at the expense of convenience

Plumbing: Associate file and functionality

- ▶ The plumber is a file server to communicate data between pertinent programs
- ▶ How processes use it
 - ▶ *Writing to: process has data that needs to be handled*
 - ▶ *Reading from: process(es) read file for data it conventionally handles*
- ▶ If no program open is eligible to handle this data, a rule file matches the file with a program to run

Consequence of encapsulation and ubiquity

- ▶ Components of a system can be shared by exposing file systems!
- ▶ The user is to use a diskless terminal, while connecting to:
 - ▶ *A CPU server which has many processors on tap*
 - ▶ *A file server which has plentiful working and static storage*
- ▶ This distributed system was the original vision of Plan 9

U-nixed

GNU

- ▶ Unruly

`ioctl` syscall

- ▶ File I/O a sufficient abstraction

Superuser

- ▶ 'host owner' the closest parallel, only has resource control

`{hard,sym}links`

- ▶ `bind` + private name spaces work similarly

Usage

One who computes

- ▶ 'I want to '{'write text','view documents',
'process text'}' in Plan 9!'
▶ *Sorry, you can't brace expand in Plan 9*
- ▶ 'I want to '^('write text' 'view documents'
'process text')!^' in Plan 9!'
▶ *Yes, good.*

Userspace: The Final Frontier

- ▶ Everything you could possibly use a computer for:
 - ▶ *write text* (*sam*, *acme*)
 - ▶ *view documents* (*page*)
 - ▶ *process text* (*sed*, *awk*, *regexp*)
 - ▶ *format text* (*troff*)
 - ▶ *browse the web* (*mothra*)
 - ▶ *compile and link C code* (e.g. *8c*, *8l*)
 - ▶ *shell* (*rc*)

The Shell - rc

- ▶ Simplified syntax for sake of writability
- ▶ Canonical Bourne Shell features retained, like file redirection and piping
- ▶ Not too many surprises here

Sam

- ▶ Similar to `ed` in spirit
- ▶ Two windows:
 - ▶ *Text editing window: WYSIWYG*
 - ▶ *Command language window: `ed`-like*
- ▶ Operates on a selection of text, called a 'dot'
 - ▶ *`i/foo` for inserting 'foo' before dot*
 - ▶ *`c/foo` for changing contents of dot to 'foo'*
 - ▶ *`a/foo` for appending 'foo' after dot*

Acme

- ▶ Culminates Plan 9's design choices into a pleasant interface
- ▶ Consists of columns which contain vertically-stacked windows
- ▶ The mouse facilitates UI interaction and manipulating text
- ▶ Exposes a file system for the layout and contents of windows
- ▶ Borrows the command language from `sam`
- ▶ Like Emacs in spirit, where users may spend most of their time in Acme

Plan 9 C Compiler

- ▶ Plan 9 has its own flavor style of C, similar to C99
- ▶ Compiler and linker distinct programs for each architecture
- ▶ The process may be automated with `mk`
- ▶ Trivial cross-compilation - just set `objtype` to the target
- ▶ An interpreter exists for all targets (for MIPS: `vi`)

Plan 9 C Compiler

```
void
main(int argc, char *argv[])
{
    print("hello, world\n");
    exits("");
}
```

► To run in the shell (386 architecture):

```
% 8c foo.c
% 8l foo.8
% 8.out
hello, world
%
```

Text Processing

- ▶ `sed`, `awk`, and a regular expression language are all there
- ▶ Some Unix commands deemed superfluous as these accomplish the same thing
 - ▶ *Example: head is omitted as `sed 10q` works too*
 - ▶ *Want any of them back? Make an `rc` script with the same name in `$home/bin/rc/`, which is bound to `/bin`*
- ▶ Have fun!

:(

Where in the World is Plan 9 Users Group?

- ▶ Plan 9 didn't catch on
- ▶ Plan 9 wasn't better enough than Unix
- ▶ Plan 9 had a commercial license and cost money for most of its lifespan
- ▶ Plan 9 needed a kick-start to fix issues with documentation and sparse drivers, but never got one
- ▶ Plan 9 died in 2015

Dead, but not forgotten

- ▶ Forks of direct successors or inspirations to Plan 9 exist
- ▶ 9front: Plan 9 but with more drivers and programs
- ▶ Harvey OS: Plan 9 rewritten to build in GCC or Clang
- ▶ Akaros: An research operating system oriented for HPC/parallel applications
- ▶ `/proc`, a virtual file system exposing functionality with the kernel, was borrowed by the Linux kernel