Plan 9

Required reading: Plan 9 from Bell Labs

Background

Had moved away from the ``one computing system'' model of Multics and Unix.

Many computers (`workstations'), self-maintained, not a coherent whole.

Pike and Thompson had been batting around ideas about a system glued together by a single protocol as early as 1984. Various small experiments involving individual pieces (file server, OS, computer) tried throughout 1980s.

Ordered the hardware for the ``real thing'' in beginning of 1989, built up WORM file server, kernel, throughout that year.

Some time in early fall 1989, Pike and Thompson were trying to figure out a way to fit the window system in. On way home from dinner, both independently realized that needed to be able to mount a user-space file descriptor, not just a network address.

Around Thanksgiving 1989, spent a few days rethinking the whole thing, added bind, new mount, flush, and spent a weekend making everything work again. The protocol at that point was essentially identical to the 9P in the paper.

In May 1990, tried to use system as self-hosting. File server kept breaking, had to keep rewriting window system. Dozen or so users by then, mostly using terminal windows to connect to Unix.

Paper written and submitted to UKUUG in July 1990.

Because it was an entirely new system, could take the time to fix problems as they arose, *in the right place*.

Design Principles

Three design principles:

1. Everything is a file.
2. There is a standard protocol for accessing files.
3. Private, malleable name spaces (bind, mount).

Everything is a file.

Everything is a file (more everything than Unix: networks, graphics).

```
% ls -l /net
% lp /dev/screen
% cat /mnt/wsys/1/text
```

Standard protocol for accessing files

9P is the only protocol the kernel knows: other protocols (NFS, disk file systems, etc.) are provided by user-level translators.

Only one protocol, so easy to write filters and other converters. *Iostats* puts itself between the kernel and a command.

```
% iostats -xvdfdf /bin/ls
```

Private, malleable name spaces

Each process has its own private name space that it can customize at will. (Full disclosure: can arrange groups of processes to run in a shared name space. Otherwise how do you implement *mount* and *bind*?)

Iostats remounts the root of the name space with its own filter service.

The window system mounts a file system that it serves on `/mnt/wsys`.

The network is actually a kernel device (no 9P involved) but it still serves a file interface that other programs use to access the network. Easy to move out to user space (or replace) if necessary: *import* network from another machine.

Implications

Everything is a file + can share files => can share everything.

Per-process name spaces help move toward ``each process has its own private machine."

One protocol: easy to build custom filters to add functionality (e.g., reestablishing broken network connections).

File representation for networks, graphics, etc.

Unix sockets are file descriptors, but you can't use the usual file operations on them. Also far too much detail that the user doesn't care about.

In Plan 9:

```
dial("tcp!plan9.bell-labs.com!http");
```

(Protocol-independent!)

Dial more or less does:
write to /net/cs: tcp!plan9.bell-labs.com!http read back: /net/tcp/clone 204.178.31.2!80 write to /net/tcp/clone: connect 204.178.31.2!80 read connection number: 4 open /net/tcp/4/data

Details don't really matter. Two important points: protocol-independent, and ordinary file operations (open, read, write).

Networks can be shared just like any other files.

Similar story for graphics, other resources.

Conventions

Per-process name spaces mean that even full path names are ambiguous (`/bin/cat` means different things on different machines, or even for different users).

Convention binds everything together. On a 386, `bind /386/bin /bin`.

In Plan 9, always know where the resource *should* be (e.g., `/net`, `/dev`, `/proc`, etc.), but not which one is there.

Can break conventions: on a 386, `bind /alpha/bin /bin`, just won't have usable binaries in `/bin` anymore.

Object-oriented in the sense of having objects (files) that all present the same interface and can be substituted for one another to arrange the system in different ways.

Very little ``type-checking'': `bind /net /proc; ps`. Great benefit (generality) but must be careful (no safety nets).

Other Contributions

Portability

Plan 9 still is the most portable operating system. Not much machine-dependent code, no fancy features tied to one machine's MMU, multiprocessor from the start (1989).

Many other systems are still struggling with converting to SMPs.

Has run on MIPS, Motorola 68000, Nextstation, Sparc, x86, PowerPC, Alpha, others.

All the world is not an x86.

Alef

New programming language: convenient, but difficult to maintain. Retired when author (Winterbottom) stopped working on Plan 9.

Good ideas transferred to C library plus conventions.

All the world is not C.

UTF-8

Thompson invented UTF-8. Pike and Thompson converted Plan 9 to use it over the first weekend of

September 1992, in time for X/Open to choose it as the Unicode standard byte format at a meeting the next week.

UTF-8 is now the standard character encoding for Unicode on all systems and interoperating between systems.

Simple, easy to modify base for experiments

Whole system source code is available, simple, easy to understand and change. There's a reason it only took a couple days to convert to UTF-8.

```
  49343  file server kernel

 181611  main kernel
  78521    ipaq port (small kernel)
  20027      TCP/IP stack
  15365      ipaq-specific code
  43129      portable code

1326778  total lines of source code
```

Dump file system

Snapshot idea might well have been ``in the air'' at the time. (`OldFiles` in AFS appears to be independently derived, use of WORM media was common research topic.)

Generalized Fork

Picked up by other systems: FreeBSD, Linux.

Authentication

No global super-user. Newer, more Plan 9-like authentication described in later paper.

New Compilers

Much faster than gcc, simpler.

8s to build acme for Linux using gcc; 1s to build acme for Plan 9 using 8c (but running on Linux)

IL Protocol

Now retired. For better or worse, TCP has all the installed base. IL didn't work very well on asymmetric or high-latency links (e.g., cable modems).

Idea propagation

Many ideas have propagated out to varying degrees.

Linux even has bind and user-level file servers now (FUSE), but still not per-process name spaces.