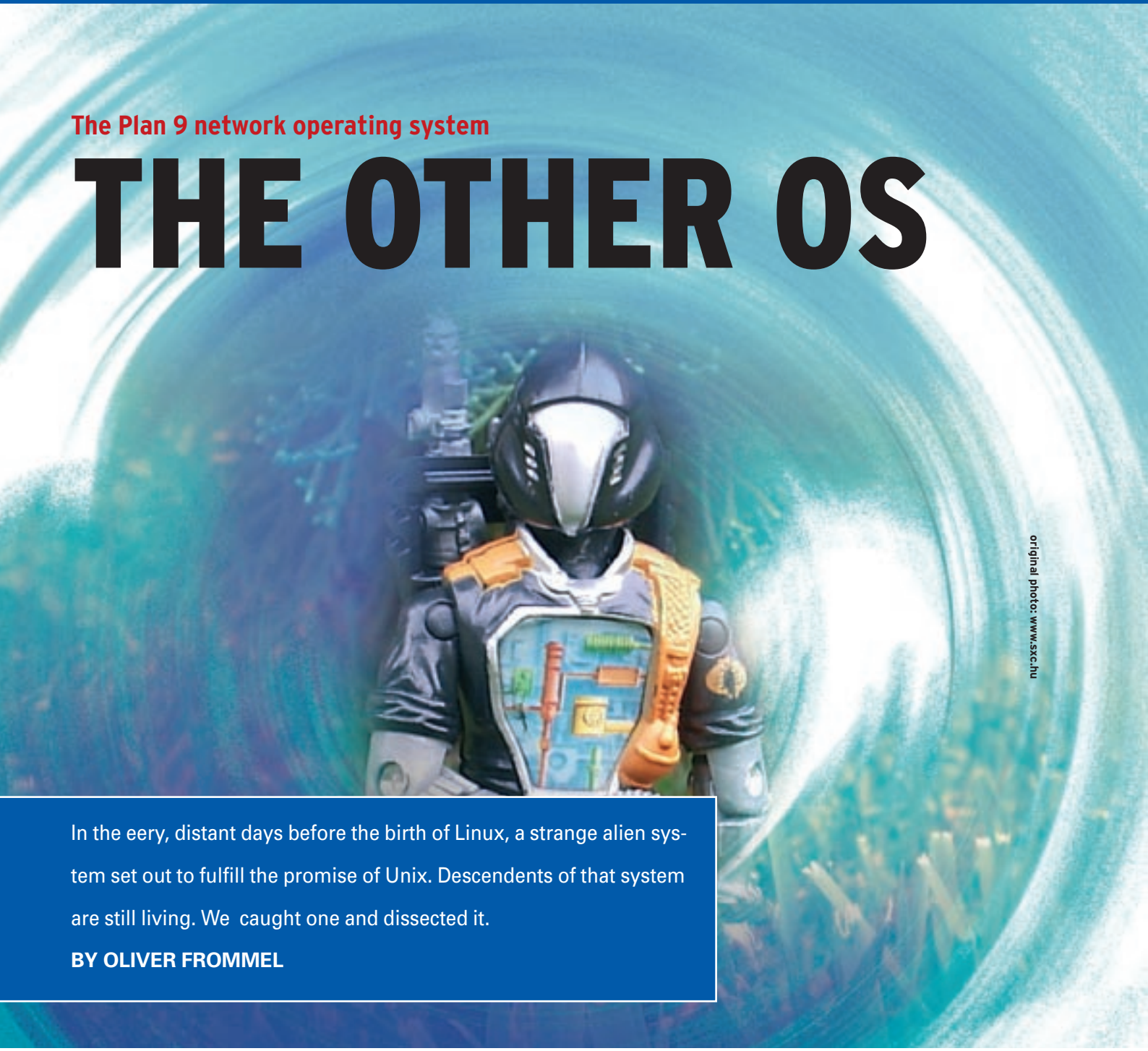
The Plan 9 network operating system

THE OTHER OS



original photo: www.sxc.hu

In the eery, distant days before the birth of Linux, a strange alien system set out to fulfill the promise of Unix. Descendents of that system are still living. We caught one and dissected it.

BY OLIVER FROMMEL

Linux has its roots in the famous AT&T Bell Labs, home of the original Unix system. The Bell Labs programmers have been busy ever since, and one of the fruits of their labors is the distributed operating system Plan 9 [1].

Plan 9 began in the late 1980s as a new system designed to address some problems with Unix that the Bell engineers considered "too deep to fix." This new operating system did indeed come with some innovations that had an influence on later systems. But until recently, Plan 9 was under a proprietary license that never really caught on with users.

Bell Labs and a small community of programmers have continued to perfect and develop Plan 9. The license was liberalized over the years, and in 2003, Plan 9 was finally released under a recognized open source license. A commercial offshoot, known as Inferno, is also available under a free license [2].

We had a look at the current "4th Edition" release of Plan 9 to see if this legendary alternative is still alluring.

Getting Started

The distribution takes the form of a 65Mbyte packed ISO image, which is also suitable for use as a live CD. If you have difficulty booting from the CD, you can check the website for a boot floppy generator to match your hardware. The install is spartan, but well-organized and more or less foolproof (Figure 1). The installer checks the current status after each step. An image for graphics adapters with Vesa mode support is also available.

If you prefer not to experiment with a full installation right now, you might like to try your luck with a virtual machine. See the "Virtual Plan 9" box for details. If you are interested, you can also try out a userspace-only install including Plan 9's most important tools [3].

More Consistent than Unix

The underlying concept for Plan 9 is that it is a distributed operating system, not

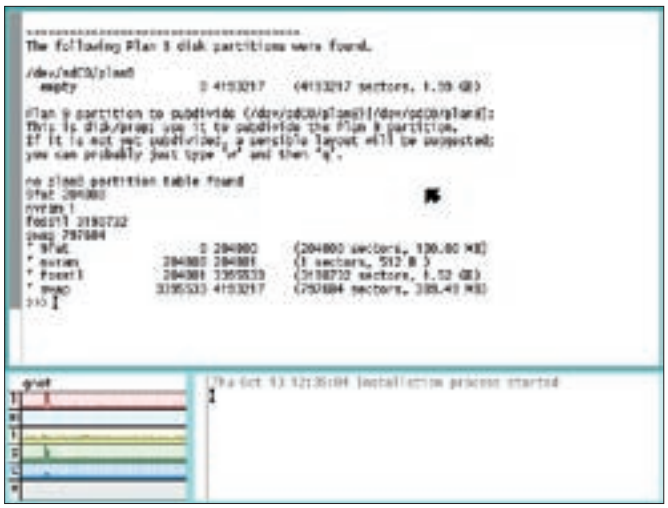


Figure 1: The so-called graphical installation displays the system load and a log window below the text-based menu. The tool suggests the four required partitions.

like Unix, where network functionality is added by mechanisms such as remote login and networking filesystems. In Plan 9, networking is built into the foundations of the operating system. For example, all resources can theoretically be distributed transparently across a Plan 9 network. The system hides the fact that resources are non-local from the user. Plan 9 not only supports file servers, but also authentication servers and CPU servers. You can use Plan 9 to implement grid systems such as 9grid.

Plan 9 is based on the Unix paradigm that "Everything is a file." Due to the huge collection of add-ons, Unix was increasingly forced to accept ad-hoc changes that contravene this original principle. One example of this is the ghastly socket interface, which uses read and write functions that are different from normal files. Plan 9 puts an end to this, providing file-based system interfaces, for example, */net/tcp* and */net/udp*, as network interfaces.

Most system services follow the server principle and interact via file-based interfaces. Plan 9 does not have the usual FTP program. Instead, the *ftps* server mounts the directory in */n/ftp* on the server. The *9660fs* server is responsible for mounting CD ROMs. Plan 9 was one of the first operating systems to implement the Proc filesystem to support file-based process control. Basic network functionality is provided by the Plan 9 file protocol, 9P, which comprises about 30 protocol messages. A Linux implementation [4] of 9P was added to the kernel mainstream in version 2.6.14, thus removing any obstacles to Linux and Plan 9 interaction.

Namespaces

Plan 9 takes this file-based design a step further with the concept of namespaces. Traditionally, Unix manages all its resources in a single namespace, in which */dev/tty1* always represents the same terminal. In contrast to this, Plan 9 applications have their own namespaces, so that */dev/window* will always point to an application's own window.

Union mount is another important feature. The union mount feature helps simplify the management of resources such as files and directories by mounting multiple directories in a single directory. This makes it possible to mount directories with executables on the file server in the local */bin* directory, which removes the need for typical *PATH* variable tricks. Linux just recently adopted this design with the overlay file system, Union-FS.

Graphics with Rio

In contrast to X11 on Unix, Plan 9 directly integrates a GUI desktop, and this vastly simplifies programming models. The underlying model is based on Niklaus Wirth's Oberon [5], which in turn draws on the Xerox Cedar system. Plan 9's desktop provides a simple programming model with a file interface and elementary operations. For example, a simple *cat /dev/screen > output* creates a screenshot of an application. In contrast to this, the Rio implementation is quite sophisticated: it comprises concurrent processes and threads that call each other reciprocally.

Rio can be seen as a departure from the legacy, line-oriented terminal model that harks back to the days when printers really did echo user input. Linux still has residual traces of this line-oriented heritage in the form of the many terminal window applications, from Xterm to the KDE console. This makes it possible to enter input at any position in a window: users simply select a command and use the mouse to execute. As an advanced windowing system, Rio makes heavy use of any mouse buttons you have available.

The Acme program is an unusual mix of shell, editor, and windowing system. Acme follows the principle we just outlined but adds elementary menus to terminal windows and redirects output to new subwindows (Figure 2). An X11 implementation of Acme, dubbed Wily, is based on the 9lib library [6].

Despite its advanced age, Rio can also handle multi-language programs, as it relies on Unicode character sets. Plan 9 uses UTF-8 encoding, which was actu-

The Name

Plan 9 derives its name from the title of a cult trash film "Plan 9 from Outer Space," by the genial Hollywood dilettante, Ed Wood. According to the FAQ [8], the name upholds the Bell Labs' tradition of "selecting names that make marketeers wince."

In the film, which is often considered Wood's greatest contribution to cinema, a pair of aliens launch a plan to resurrect dead people and create an army of zombies that will attack the capitals of the Earth. The purpose of this gruesome mission is to stop humans from completing their work on the Solaranite, an exotic weapon that will "explode the particles of sunlight" and destroy the universe.

It is interesting to consider whether Bell engineers chose the name because they felt they were resurrecting the mortal remains of Unix to attack the world. The tantalizing similarity of "Solaranite" with "Solaris," another enemy weapon that derives its power from a different Sun, is perhaps stretching the metaphor, although the first versions of Solaris appeared when Plan 9 was in early development.

Other Plan 9 names also contain references to pop culture, such as the Acme editor, named for the company in the Warner Brothers Roadrunner cartoon. The old windowing system was known as 8 1/2, based on the name of a Fellini movie.

ally invented by the Plan 9 programming team led by Rob Pike.

Security

Plan 9 does not have a root superuser, and thus it does not have the kind of SUID programs that have caused no end of security problems on Unix. Following a similar approach to Kerberos, a distributed Plan 9 system does not transmit passwords over the wire but uses encrypted tickets instead. One way to create a user account, is to enable the Fossil fileserver by entering *con /srv/fscons*, and then give the *uname user user* command. New users can initialize their environments the first time they log in by entering */sys/lib/newuser*.

Of course, Plan 9 has developer tools for the C programming language. The compiler for the x86 architecture is titled *8c*, the linker is *8l*, and the build program is *mk*. To compile a new kernel with these tools, the sysop changes the directory to */sys/src/9/pc* and edits the corresponding configuration file, which can be named differently depending on the environment: in the simplest of cases, this is *pc*; *pcauth* is an authentication server, *pcf* a Fossil file server, and so on. After configuring the kernel, the admin can compile and install by giving the *mk CONF=pcf install* command. The kernel has to reside in the 9fat boot partition to boot; the partition is enabled by entering the *9fat:* command. This is also where the boot configuration file, *plan9.ini*, resides.

Not All that Glitters

The manpages provide documentation for tools and approaches; they are available online and in the Plan 9 Wiki. Unfortunately, a lot of changes were introduced with the 4th Edition, and this makes some of the information obsolete; so make sure you check the version number.

Despite the fantastic design, Plan 9 is not perfect. For example, Rio 9term windows often fail to scroll properly and will not show you the text passages you need. Acme is less buggy. The whole system is based on a GUI desktop, and many functions are simply not available at the text-based console: one example is the ability to quit processes by pressing the [Del] key. Another issue is the fairly small user community, Comp.os.Plan9, which continues to work hard on developing Plan 9, fights a constant battle against the lack of resources.

The fact that Plan 9 has not achieved widespread distribution thus far can certainly be attributed to the previous licensing policy, which prevented unhindered distribution. Today, Plan 9 is far less interesting for developers of free software, as Linux gives them an operating system that works well, despite dragging a lot of legacy Unix ballast around with it [7]. ■

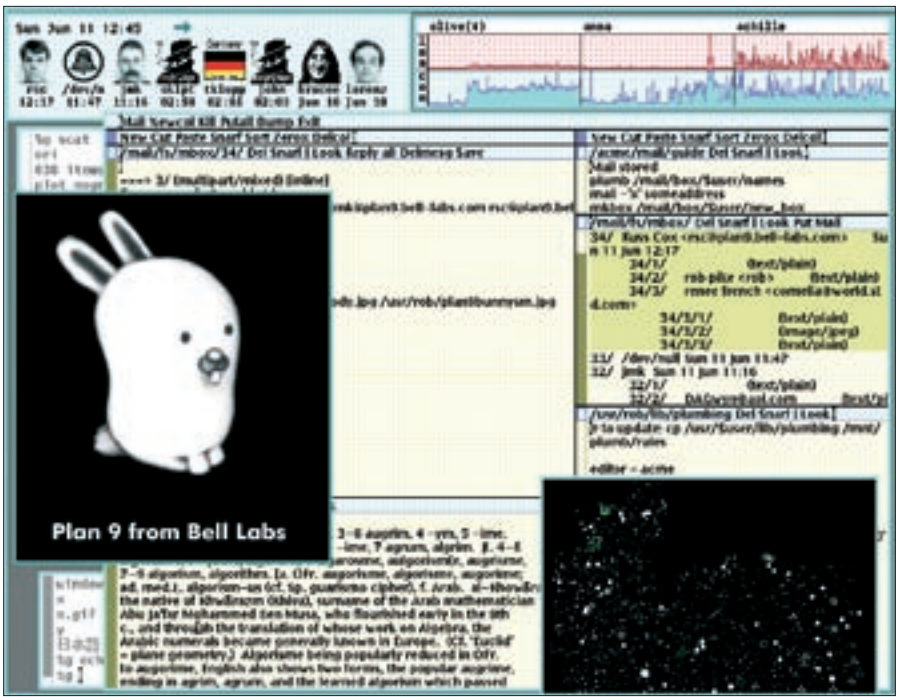


Figure 2: A Plan 9 system in action. A multiply-tiled Acme editor window is hiding behind the mascot, Glenda, as is evidenced by the menu items: New, Cut, Paste, and so on.

Virtual Plan 9

If you don't want to sacrifice a primary partition to Plan 9 at this stage, you can use various emulators for your experiments. The easiest way is to use VMware; the Plan 9 site even has a ready-to-run image for you. This removes the need to install; just download the image, plug it into the virtual machine, and boot. However, the image will only run on VMware 4. Plan 9 will not run at all on VMware 5, according to the developers.

Instead, the developers recommend the free PC emulator Qemu, which is almost (but not quite) as fast as VMware with the non-free kernel accelerator module, Kqemu. Qemu can boot directly using the Plan 9 ISO image download. First create a disk image with the *qemu-img* command, then boot from the ISO:

```
qemu-img create plan9.img 2G
qemu -cdrom plan9.iso -boot d ➥
-hda plan9.img
```

After you complete the install, which could take a few hours, *qemu -boot c plan9.img* will boot the new system.

Alternatively, there is a DVD image at [9] by Xenoppix, a Knoppix variant which integrates the Xen Virtual Machine Monitor and can boot NetBSD and Plan 9.

INFO

[1] Plan 9: *http://www.cs.bell-labs.com/plan9dist*

[2] Inferno by Vita Nuova: *http://www.vitanuova.com*

[3] Plan 9 from Userspace: *http://swtch.com/plan9port*

[4] V9FS for Linux: *http://v9fs.sourceforge.net*

[5] Oberon: *http://www.oberon.ethz.ch*

[6] Acme clone Wily: *http://sf.net/projects/wily*

[7] Eric Raymond on Plan 9: *http://www.faqs.org/docs/artu/plan9.html*

[8] Plan 9 FAQ: *http://www.cs.bell-labs.com/wiki/plan9/FAQ*

[9] Xenoppix: *http://unit.aist.go.jp/itri/knoppix/xen/index-en.html*