Make room for the FSF kernel

HURD THE GNUS?

RICHARD SMEDLEY

The GNU Hurd is the GNU project's replacement for the Unix kernel. Most of us are getting by quite nicely with the Linux kernel in this role, but the Hurd will offer several advance features when it reaches release status. Come and discover more.

The release, as we near publication, of Debian's new *G1* binary CDs of GNU/Hurd has prompted an expansion of this column's coverage of the Hurd. We hope to present to you our full report on installing and running this release in the next issue, meanwhile we investigate what makes the Hurd so special.

How's Trix?

Three years after the founding of the Free Software Foundation (FSF), by Richard Stallman, the first *GNUsletter* identifies *TRIX – as the GNU kernel. In 1987, whilst changes continued to be made to TRIX, Stallman and the FSF began negotiating with Professor Rashid of Carnegie-Mellon University (CMU) about working with them on the development of the Mach kernel. In the February 1988 GNUsletter* Stallman spoke about taking Mach and putting the Berkeley Sprite filesystem on top of it, after first removing the *Berkeley Unix* – specific code. At the end of 1988 the choice is still not settled between Mach, which was not yet Freely licenced, and TRIX. Sprite was even discussed as a full kernel solution.

Some work starts on developing the Hurd in 1990 but more work goes into trying to free Mach: "We are still interested in a multi-process kernel running on top of Mach. The CMU lawyers are currently deciding if they can release Mach with distribution conditions that will enable us to distribute it. If they decide to do so, then we will probably start work. CMU has available under the same terms as Mach a single-server partial Unix emulator named Poe; it is rather slow and provides minimal functionality. We would probably begin by extending Poe to provide full functionality. Later we hope to have a modular emulator divided into multiple processes." – *GNUsletter*, January 1991.

Eventually Mach 3.0 is released under a GNU GPL-compatible licence and the Hurd is officially announced as a group of servers running on top of the Mach microkernel.

It was into this background of a slow start for the GNU kernel that Linus Torvalds announced a kernel of his own. Ten years on we see the Linux

A Unix Tradition.

The Free Software world is littered – some would say plagued – with bad puns and self-recursive acronyms. `Hurd' stands for `Hird of Unix-Replacing Daemons'. And, then, `Hird' stands for `Hurd of Interfaces Representing Depth'. So we have the first software to be named by a pair of mutually recursive acronyms.

Marcus Brinkmann working on the HURD

kernel scaling up for enterprise, taking over the embedded sector and slowly marching onto the desktop – particularly where the advantages of thin client solutions are appreciated. GNU/Linux is the success story of the FSF's project to produce a free Unix-like OS.

Debian and the Hurd

During those ten years work on Hurd has occasionally been rather slow. Although the Hurd was booting by 1994, and later that year ran emacs and gcc, progress slowed after the binary release of 1996, which worked with NetBSD boot floppies. Piecemeal development left the Hurd broken. It was in this state when in 1998 Marcus Brinkmann started the Debian GNU/ Hurd port.

A keen Free software advocate, Brinkmann was also inspired by the freedoms which a microkernel can give when running a multi-server system such as the Hurd (see *Free, Freer, Freest*, below). Brinkmann's lead in bringing the powerful packaging tools of Debian (such as *dpkg*) and the porting to GNU/Hurd of almost half of the packages currently available for Debian GNU/Linux has helped to revitalise the kernel project. Making GNU/Hurd a Debian port has brought in many more developers and users to contribute bug reports. Debian binary CDs and a not-too-difficult installation procedure have resulted in a more accessible system for those merely curious today, who may be the committed users of tomorrow.

A New Strategy of OS Design

OS development in the period up to the mid 1990's left many with the impression that microkernels were "better in theory but worse in practice." We will counter-balance that false conclusion with a rehearsal of the differences between microkernels and monolithic kernels and then look at what makes the Hurd (on Mach) so different from other microkernel OS implementations.

A monolithic kernel is (relatively) easy to get off the ground, but gradually becomes harder to maintain. All of the services that an OS must perform for programs to share hardware, and for users to share a computer, are implemented in the kernel. The kernel grows as more code is added for device drivers, network protocols, process management, authentication, file systems, POSIX compatible interfaces and more.

As all parts of the kernel can access all of the kernel's data structures, this is a temptation for some coders to make short cuts, instead of programming clean interfaces. In the real world this leads to a faster kernel at the expense of clarity and comprehensibility in the code. A change to one small part of the kernel can break an apparently unrelated part.

Of course the monolithic kernel has a very rich set of features and has no need for message passing – components communicate with each other transparently. A system of dynamically loaded modules – as now used in Linux and AtheOS (see *Linux Magazine* issue 13) – improves the system further.

Some features have not made it into Linux as an attempt is made to contain the size and complexity of the kernel. Nevertheless some code bloat is inevitable as core functionality grows with inevitable creeping featurs. A microkernel avoids this by only implementing the *infrastructure*

Why not Linux?

You may find that GNU/Linux is the answer to most of your requirements. However the GNU organisation puts forward a powerful case for the Hurd:
It's Free Software – all the benefits and protections of the GPL.
It's compatible – a modern, Unix-like kernel using the GNU C library which closely follows the standards of ANSI/ISO, BSD, POSIX, Single Unix, SVID, and X/Open.
It's built to survive – object-oriented structure.
It's scalable – The Hurd implementation is *aggressively* multithreaded so that it runs efficiently on both single processors and symmetric multiprocessors. The Hurd interfaces are designed to allow transparent network clusters (*collectives*), although this feature has not yet been implemented.
It's extensible – every part of the system is designed to be modified and extended.
It's stable – It is possible to develop and test new Hurd kernel components without rebooting the machine (not even accidentally). Running your own kernel components doesn't interfere with other users, and so no special system privileges are required. The mechanism for kernel extensions is secure by design: it is impossible to impose your changes upon other users unless they authorize them or you are the system administrator.

Free, Freer, Freest.

Linux and BSD give you freedoms to modify, run and copy the OS and applications you need.
Hurd gives additional freedoms – particularly to the advance user.
Kernel code is no longer something that cannot be touched. Any user, without root privileges, can extend their kernel with their own services, and share their code with other users. Hurd takes ideas of Free Software and of unix modularity to its logical extreme, ultimately benefiting developer and user alike.

necessary for other tasks to provide the features required of a modern OS. This boils down to resource management (paging policy and scheduling) and message passing, all other services can be run from user space - though basic hardware device support may be needed to bootstrap the system.

To recap briefly the advantages of a microkernel. What platform specific code there is, is often limited to the microkernel making porting easy. The modularity of the design has the advantage of making it easier to integrate new features and upgrades into the OS. Most processes can run in user space – if one comes crashing down it will not take the rest of the OS with it, as would be the case if it were in kernel space.

A better service

Other designs implemented upon the Mach microkernel have single servers and thus many of the disadvantages of a monolithic kernel. The Hurd is a multi server system. Each server runs as a Mach Task and provides *one* of the services of the OS. They communicate through Mach message passing. Bugs in a server will not affect the rest of the OS – so overall stability is improved. Indeed if a file system server for a mounted partition crashes it need not take down the whole system. The partition is unmounted and the server can be started again with debugging information (using gdb). Testing new servers means no reboot – nor is there a need to take down any existing servers.

Amongst multi-server systems the Hurd is unique for allowing *users* to replace or add to virtually all of the system dynamically. Authentication servers establish the identity of programs which need to trust each other and the process server establishes control over system components by the superuser. No other server has any special status.

Partition number blues

Linux users are familiar with the hd[a-d][1-n] system of referring to disks and partitions. Those making the move from LILO to GRUB find a new system of (hdN,n) with all the disks in order, first the IDE devices, then the SCSI; disks and partitions are zero indexed. For example (hd0,3) refers to the fourth partition on the first drive. (hd1,0) refers to the first partition on the second drive – if there is only one IDE drive then (hd1,0) is on the first SCSI drive, if there are at least two IDE drives then (hd1,0) is the second IDE drive, regardless of whether it is a master or slave disk on the primary or secondary controller (Thus Linux might see, for example, hdc1 for the same disk and partition). This is the way that the BIOS sees the disks.

It is important to become comfortable with the different systems to avoid mistakes during installation – particularly as Hurd uses a slightly different system, based upon the BSD *slice approach to partitions. In the example of hdc1 (linux) or (hd1,0) (GRUB) above, Hurd calls it as hd1s1 if it is an IDE drive, however SCSI drives are numbered according to their SCSI id. Each system exists for a good reason and anyone wanting to cross-install Hurd from Linux and boot with GRUB will for now, unfortunately, just have to get used to it.*

A good Mach

Hurd is a set of services which runs upon the Mach microkernel – originally upon Mach 3 – with device drivers from Linux 2.0.x
From this was developed GNU Mach, which is what new users will probably try first. However for extra flexibility most developers work with OSKit Mach. Utah's *OSKit for OS construction* with Mach sitting on top and handling virtual memory management and messaging.

Make it clean

The very modularity of the system – the way it is split up into individual components – calls for clear and consistent interfaces to be written at the start. Responsibilities are clear and the object oriented (OO) nature of the design means that code is easier to maintain and develop and new services easier to write.

Inter-process communication in Mach is based on the ports concept. A port is a message queue which can be used as a one-way communication channel. You also need a port right. This can be a send right, receive right, or send-once right. With the appropriate port right, you are allowed to send messages to the server, receive messages from it, or send just one single message.

Mach ports are analogous to the ports of socket-based communication. It does not matter if the communicating threads are executing on the same processor, on separate processors in a multiprocessing environment, or on different computers on a network. Mach scales up well for distributed computing.

A fine translation

A translator is a Hurd server which provides the basic filesystem interface. It sits between the contents of a file and the user accessing this file. Translators are just another user process – so can be run by any user. The only privileges needed are access rights for the underlying inode to which the translator is attached. The information about translators is stored in the inode – many translators don't even require an actual file.

A translator can be provided for the node /ftp, enabling transparent ftp and such commands as

```
less /ftp/ftp.uu.net/inet/rfc/rfc1097
```

The flexibility implied by translators can take a while to sink in. Mount points, symbolic links and device files are all translators. A running translator is an *Active* translator, however powerful handling of devices and files comes from Passive translators. A passive translator is one that has not yet started. As soon as the passive translator is accessed, it is automatically read out of the inode

Richard Stallman

Info

GNU/Hurd:	*http://www.gnu.org/software/hurd/*
Grub:	*ftp://alpha.gnu.org/gnu/grub*
Snapshot of Hurd source code:	*ftp://alpha.gnu.org/gnu/cvs/hurd.tgz*
Debian GNU/Hurd:	*http://www.debian.org/ports/hurd*
	http://lists.debian.org/ports.html
Kernel cousin archives:	*http://kt.zork.net/debian-hurd/back-issues.html*
Easy install:	*http://www.pick.ucam.org/~mcv21/hurd.html*
A UK Hurd page:	*http://www.hurd.uklinux.net.html*

and an active translator is started on top of it using the command line that was stored in the inode. If the active translator is lost a new one will be started next time the inode is read (the device is accessed). As it sits on the inode it survives reboots – there is no need to maintain a configuration file for example mount points.

For a more comprehensive look at translators and the other Hurd servers I recommend the documentation on the GNU Hurd Web site.

Follow the Hurd

Potential kernel hackers may find that there is much less to learn before starting coding, due to the modular construction and cleanly designed interfaces. As described above, services can be dynamically loaded and debugged on a live system without impinging upon other functionality of the OS.

In the *info box* you will find details of mailing lists and other resources. Do not feel that you have to be able to hack C to help. Anyone can get involved maintaining web sites, submitting bug reports or compiling their favourite applications to run on Hurd.

The ability to dynamically replace the various system servers may reinforce the impression of a geek OS, nevertheless Debian's packaging ensures that many users will get to experiment with features which will never be available under even the best monolithic kernels (read Linux). If you would like to try it out, look out for the report on our experiences with the new Debian GNU/Hurd CDs in future articles, or download it now from debian – there is plenty of good documentation on the site and at *http://hurd.gnu.org/*