(Soon to be) Frequently Asked Questions about the Utah PA-RISC Code Snapshot

0a. What is this primarily, and its primary purpose?

Essentially, all the PA-RISC and hp700 specific code necessary to get an alternative operating system running on an hp700 workstation, without legal encumbrance (with one small exception). In addition to code written at Utah and by other organizations, HP recently granted permission to freely distribute a number of files essential to this snapshot. Although it is embodied in the form of complete kernel+server source trees and boot images, because the system is unstable it is not intended as a ‘‘release’’ that you load on your machine and run. This snapshot is primarily to assist O.S. developers who wish to target the PA.

0b. But what did it also become, sort of?

It’s a bootable, runnable Mach/Lites system for the hp700 series, with a fairly complete user environment, and complete ELF support. Everything except the bootloader was built using the system itself. For us, it typically stays up for 2-4 hours under compile/edit cycles, before a leak catches up with it. **However, it is not really intended to be run!**

1. When was the snapshot first made available?

December 18, 1994.

2. When will the system be available in a ‘‘release’’ (i.e. stable) form?

By the spring we’d like to have a system which is substantially more complete and stable, but this depends partly on the community for contributions in those areas.

3. What HP machines does the snapshot run on?

Most PA-RISC 1.1 (aka hp700) workstations including the original ‘‘snakes’’ (705, 710, 720, 730, 750), the 7100-based 7x5s, the 7100LC-based 712 (‘‘gecko’’) and newer 715s, and the 7150 based 735/125. Refer to the supported hardware section of the release page for details.

4. Will it run on my older PA 1.0 (aka hp800) workstations?

Not as it stands now. Most of the code still exists in the kernel to handle CIO based 1.0 machines (834/835), and a bunch of drivers are all there: the LAN, HP-IB disk, 6-port MUX, and ‘‘Fireeye’’ 8-bit color display. Other things that will have to be addressed:

- no block TLBs on the 800s (kernel is mapped with one)
- 800s don’t have **rfir** instruction (used in TLB miss code)
- 800s have 2k instead of 4k page size (affects TLB data structures, binary layout, etc.)
- 800s have only 16 FP registers (kernel, setjmp, etc. assume 32)
- 800s have slightly different instruction sets (By default, gcc generates PA 1.1 code)

5. Will it run HP-UX binaries?

Not yet. The HP-UX compatibility code from our BSD port has not been integrated. When we merge this code into Lites, you should be able to run most interesting HP-UX applications such as compilers, the X server, and even VUE.

6. Will it run on top of an HP-UX filesystem?

No. Though it would be nice to plop a kernel and server down on your HP-UX disk and boot that instead, you cannot do it. Besides the fact that it cannot yet execute HP-UX binaries, Lites does not understand the HP-UX filesystem layout (which is almost, but not quite, FFS). Also, we use a different standalone boot program which has a couple of implications: 1) the PA Mach kernel is an ELF object file and the HP-UX boot loader can only handle SOM, 2) the boot programs have different interfaces to the program they load so the HP-UX boot won’t communicate correctly with Mach/Lites and our boot won’t communicate with an HP-UX kernel.

7. Why ELF instead of SOM?

We don’t have source for a complete compiler toolchain for SOM, lacking a linker. Under BSD, we have always just used the HP-UX linker running in compatibility mode. Because ELF is much more widely used than SOM, over the years we’ve invested many man-years developing its PA version. Since it has been our intention to switch to ELF for some time anyway, this seemed like a good opportunity. The big downside is that due to a fundamental mismatch between the generic ELF BFD code and the needs of the PA ELF linker, the linker is about four times as slow as the HP-UX SOM linker. We plan to address linker performance in the future. It is possible there will be a native SOM linker in the future (we have done some of the BFD support), but in any case the HP-UX linker will run when compatibility is integrated, and we plan to continue to recognize SOM format in Lites.

8. Is there an X-server I can use?

Not right now. Again, we have always just used the HP-UX X server in compatibility mode. HP has made available a sample server which has been included in the last couple of X releases, but we have never messed with it. We don’t have the time to support an X server.

9. How can I debug the PA Mach kernel?

PDB, the ‘‘printf’’ debugger :-) Our debug environment is a little lacking in this snapshot. By the time of the first real release, we should have remote kernel debugging re-integrated. This remote debugging, also known as ‘‘KGDB,’’ has a serial line version developed at Utah as well as a newer LAN-based version contributed by Convex. Also from Convex is a variation that allows you to ‘‘attach’’ and debug arbitrary tasks remotely. In addition, we have GDB changes to allow post-mortem debugging of crash dumps which will also appear in the first release. We do not have DDB, the traditional Mach debugger. We made a run at it once but didn’t finish. The partial code is included in the `kernel_unused` tree, and finishing it would be wonderful.

10. How can I debug the PA Lites server?

Ditto. Lites has a feature to allow it to run as a ‘‘second server’’; i.e. it can run as another user task along side a working server. It communicates with the primary server to talk to devices. You need a separate disk and IP address to do this. With this technique, you can debug the server with a standard GDB. However, PA Lites isn’t stable enough to use as a primary server right now and Lites doesn’t support debugging (e.g. ptrace) yet. This needs to be fixed in future releases, by us or others. In addition we need a GDB which talks ‘‘native Mach,’’ making it easier to debug with a flaky server.

11. How can I debug a user process?

Ditto. Lites doesn’t have a working debug interface yet. Future releases will have either ptrace or `/proc` or both.

12. How can I learn more about PA hardware?

The place to start would be the PA-RISC Architecture and Instruction Set Reference Manual. The current version, the third edition, is available online from HP’s web server. The 700 series workstations use a number of commodity parts for which you should be able to get documentation. These include: NCR 53c7x0 SCSI, WD 16C552 serial/parallel interface, Intel 82596 LAN interface, PS/2 keyboard/mouse/floppy interface. AMD Supernet2 FDDI, PSB 2160 and Crystal CS4215 audio codecs. The bus interconnect, DMA, and graphics interfaces are not documented in a generally available form as far as I know.

13. What version of Mach is this?

The Mach kernel source in the snapshot is what we are currently calling ‘‘Mach4.’’ It is the kernel we are evolving for our ARPA-funded Fast and Flexible Mach-based Systems project. It is derived from CMU’s MK83a release. Refer to the Mach 4 page for more details.

14. What exactly is ‘‘Lites’’?

Lites is a user-mode, single-task implementation of BSD Unix which runs on top of a Mach micro-kernel. It is being developed by Johannes Helander at Helsinki University of Technology (HUT) in Finland. Lites is based on the 4.4-lite release from Berkeley, single-server code from CMU, machine dependent ports from various people, and a lot of code and work from Johannes. It is freely available and runs on the i386, pc532, Mips and PA-RISC. See the Lites home page for more details.

15. How can I take advantage of migrating threads?

For a task to become the target of a migrating-thread RPC you need to create ‘‘activations’’ and associate them with a port for which you have a receive right. Threads doing simple SEND+RCV style mach_msg calls on that port will then transparently migrate into your task. This is currently done with an undocumented, kludgey, slightly PA-centric interface to the kernel. It has also not been tested in the context of the MK83 kernel base provided.

16. How can I take advantage of presentation/interface?

Don’t even try. Though the general kernel framework is all there, only special cases work and you have to hand-build the RPC signature (because we haven’t included our IDL compiler, which is still research-only code and will change drastically).

17. Why is Mach on the PA so slow?

Welcome to Mach! As the name implies, a large component of our Fast and Flexible Mach-based Systems project is involved with making a Mach micro-kernel fast without sacrificing the modularity and protection aspects. However, our system is slow even by Mach standards due to lack of tuning in both the kernel and Lites and for one major technical reason. That reason is that the kernel is optimized for migrating RPC and the old, hand-crafted, special-case, send+receive fast IPC path is gone. Unfortunately, since nothing yet uses migrating RPC, everything winds up on the slow, general IPC path.

18. So is any of this useful for doing a fooBSD port?

Absolutely! Most of the code in the kernel `hpdev` and `hpsgc` directories is shared between our Mach and 4.3/4.4 BSD systems already. The kernel pmap module, exception/interrupt handlers and bus configuration code should come across nearly intact. And of course, the native environment is intended to be 4.4bsd, so include files and libraries are exactly what is needed. This is not to say it will be an easy task however...

[UP to Overview] [Code Tour]

Mike Hibler <mike@cs.utah.edu>