A Tour of the PA-RISC Code

XXX -- This page is still under construction

Contents

The Kernel Bootstrap

The PA boot program comes directly from our 4.3bsd base and consists of two directories: `stand` and `parisc`. The former contains the ‘‘machine independent’’ code for opening files, resolving pathnames, etc. and comes from the standard Mach release (via 4.4-lite and NetBSD).

The PA specific directory contains the remaining components necessary to build the bootstrap loader: the boot program proper and additional standalone library routines and device drivers. It also includes the `mkboot` program which is used to create a LIF format file from the SOM or ELF binary. This file is suitable for installation into `/usr/mdec` where it can be installed with the ‘‘-B’’ option of disklabel, and interpreted by the boot ROM.

The boot loader does supports the full:

```
devicetype(adaptor, controller, unit, partition)pathname
```

syntax, though you are likely to run into problems trying to change anything but the ‘‘unit’’, ‘‘partition’’ or ‘‘pathname’’ fields. It is also not very flexible about passing arguments to the kernel and does not support passing arguments to the server at all.

The bootstrap has not yet been integrated into the kernel build environment: it still uses a simple BSD-style makefile.

Mach Kernel

All of the PA specific kernel code is in `mach4/mach4-parisc`. This directory includes not only the kernel proper but the associated packages: `mig` (the Mach interface generator), `libmach` (kernel interface library), `libthreads` (the ‘‘C-threads’’ library), and `include` which contains header files for all of the above.

There is currently no machine dependent code for mig.

PA specific code in libmach all pertains to the ‘‘standalone’’ version; i.e. the version linked in with standalone Mach applications such as the Lites server. Most of these files, in the `standalone` sub-directory, are the so-called ‘‘millicode’’ routines which provide support for the compiler; e.g. indirect function calls, integer multiplication and division. There are also assembly versions of basic functions such as bzero, bcopy and setjmp/longjmp support.

C-threads support includes primitives for context-switching (co-routine calls), spin locks, and thread startup. Changes and fixes were also necessary to the machine independent part of cthreads to handle ‘‘upward’’ (toward higher addresses) growing stacks and reverse polarity locks which require explicit initialization (a zero-valued lock is taken on the PA). These were deemed potentially machine independent features that should be supported in a general fashion.

For the include tree, `machine` contains machine dependent ‘‘specializations’’ of standard include files. Here are the definition of necessary basic Mach types such as spin locks and thread state, special exception code values, the system call trap template, and architecture specific info such as page size and address ranges. `sa` contains a few miscellaneous files for standalone programs.

The kernel proper has a number of sub-directories. `aux` contains auxiliary programs for building the kernel itself, currently just the ‘‘genassym’’ program. `bogus` has all the device/pseudo-device header files normally created by the BSD **config** program. This directory should go away when a more dynamic configuration scheme is adopted. `boot_ufs` contains the old in-kernel bootstrap and default pager code from Mach3. This is all machine independent code, but is used only by the PA and hence wound up here. `hpdev` includes all the device driver code that is common between our BSD ports to the hp300/400 (68k) and hp700/800 (PA) platforms. There is an indirect console driver, a serial-port driver, the device independent code for the raw (grf) and terminal emulator (ite) interfaces to framebuffers, and a driver for the keyboard/mouse (hil) interface. All of these drivers are shared between our Mach and BSD kernels with the `MACH_KERNEL` ifdef providing the distinction.

More on the console interface.

Explain that HIL shared queues, and mmap’ed GRF don’t work.

ITE doesn’t support color or soft keys.

Lites Server and Emulator

`parisc/parisc`, `parisc/server`, includes

emulator trampoline, sysent tables, syscall stubs

Other Code

libc, libm

[overview] [FAQ]

Mike Hibler <mike@cs.utah.edu>