Mach4 UK22 Release Notes

March 29, 1996

Contents

Overview

UK22 is a public release of the so-called ‘‘Mach4’’ kernel for the Intel x86 processor family. Mach4 is one component of the Flux project at the University of Utah Computer Science Department. The focus of the x86 branch has been to make Mach more usable on PCs, with particular emphasis on ease of building, compatibility with a number of OS environments and additional hardware support. This release reflects the latest development in those areas.

New in UK22, since UK02p21:

- Greatly expanded support for Linux device drivers. Now nearly all block, scsi and network drivers are supported!
- Linux PCI devices now supported.
- Real serial console.
- New partitioning code, supporting ‘slices’.
- Fast RPC path (‘‘fipc’’)
- Multiboot 0.5 compliant.
- The paging bug that caused machines with 8MB of RAM or less to crash when booting has been fixed.

Tested Configurations

All configurations were built using i386-mach cross-build tools, which can be found at: ftp://flux.cs.utah.edu/flux/binaries/i386.

The kernel was tested with Lites-1.1.u3 running on the following base systems:

FreeBSD 2.1.0-RELEASE and 2.2-current
NetBSD 1.0 and 1.1

The system was also booted on a system running UX42 on a CMU Mach base.

Ftp Information

Mach4 and Mach4-i386

Sources
: ftp://flux.cs.utah.edu/flux/mach4-UK22.tar.gz
  ftp://flux.cs.utah.edu/flux/mach4-i386-UK22.tar.gz
  ftp://flux.cs.utah.edu/flux/patch-UK02p21-UK22.gz
  ftp://flux.cs.utah.edu/flux/patch-i386-UK02p21-UK22.gz

Intel x86: BSD bootable Binaries
: ftp://flux.cs.utah.edu/flux/binaries/i386/mach4/Mach
  ftp://flux.cs.utah.edu/flux/binaries/i386/mach4/Mach+linuxdev

Intel x86: Linux bootable Binaries
: ftp://flux.cs.utah.edu/flux/binaries/i386/mach4/zMach
  A Linux bootable kernel with linuxdev is not available, since it it too large to be booted.

Details

Serial Console:

A real serial console has been implemented that replaces the ‘‘remote console’’ code from UK02p21. This means that all console output will be redirected to a serial line, so that you can log it or connect to the machine through a terminal server, etc.

Limitations:

- Can’t use the physical console at the same time.
- No way to invoke the kernel debugger.
- Console device is not runtime configurable.

To enable the serial console support, you must change a #define in mach4-i386/kernel/bogus/rc.h. RCLINE is the number of the serial device that you want to use, and RCADDR is the port. The options for RCLINE and RCADDR are shown below, as taken from autoconf.c, for the ‘‘com’’ device (these are also in rc.h).

```
options for RCLINE:

RCLINE  RCADDR  info
------  ------  --------------------------
```

```
0       0x3f8   (port 0x3f8/irq 4 DOS COM1)
1       0x2f8   (port 0x2f8/irq 3 DOS COM2)
2       0x3e8   (port 0x3e8/irq 5 DOS COM3)
3       0x2e8   (port 0x2e8/irq 9 DOS COM4)
```

RCBAUD is now #defined in mach4-i386/kernel/i386at/com.c and defaults to 9600 baud.

New partition code:

The new partition code replaces the partition code that was in the IDE and the SCSI drivers. The code adds support for the ‘‘slice’’ abstraction that was introduced in FreeBSD 2.0.5. Slices are just another way to refer to DOS partitions. This makes it much easier to mount msdos, ext2fs or other types of filesystems from Lites, since you don’t have to make a fake entry in your disklabel (or even have a disklabel!) to access them.

Slices map directly to DOS partitions. Slices 1-4 are DOS primary partitions 1-4, slices >5 map to any DOS extended partitions, in the order that they appear on the disk. In addition to this, there is a ‘‘whole disk’’ device (ie. sd0, hd0). To access a BSD partition within a slice, you add the BSD partition letter suffix (ie. To access partition a in slice 1 on your IDE disk, you would use wd0s1a).

To make things even more confusing, there is also a ‘‘compatability slice’’ that maps to the first slice with a BSD label (usually what you want, since BSD’s on the x86 have generally only allowed one BSD partition per disk). If there are no DOS partitions on the disk, then the compatibility slice maps to the whole disk. The compatability slice is there so that old programs that don’t know about slices will still work.

To disable all the partition code’s output if you aren’t using the linux device drivers (since they use their own partitioning code), change PARTITION_DEBUG to 0 in mach4/kernel/scsi/disk_label.c

Some examples:

```
sd0     -> whole disk
sd0s1   -> DOS primary partition 1
sd0s5   -> 1st DOS extended partition
sd0s7   -> 3rd DOS extended partition

sd0s1a  -> BSD partition a, in DOS primary partition 1
sd0a    -> BSD partition a, in first ‘slice’ with disklabel
```

You need to get Lites-1.1.u3 if you want to use slice names/devices with the Lites system.

Fast RPC path:

The fipc_send() and fipc_recv() routines were impelented to provide a fast, direct path from user code directly to the network, without having to go through the standard mach_msg code path. To enable the fipc code in mach4, you must specify `--enable-fipc` on your configure line.

```
kern_return_t fipc_send (fipc_endpoint_t dest, char *buffer, int length);
```

The fipc_endpoint_t struct contains the 6 byte ethernet hardware address (hwaddr) and the unsigned short fipc_port number (port).

```
kern_return_t fipc_recv (unsigned short fipc_port, char *buffer, int *len,
fipc_endpoint_t *sender);
```

fipc_recv() copies up to len bytes into the user buffer, returning the actual message size back in len. The hardware address of the sender is returned in the hwaddr field of the *sender parameter. There is currently no concept of a sending port number.

A receiving thread will pick up the first available message on a port’s queue. If there is a waiting message, the thread will pick that up and return immediately, else it will block until a message arrives. During the time that a thread is blocked on a fipc_port, that port is considered "bound." There is no other reservation mechanism.

There is a receive queue (currently of length 4) for each open fipc_port. There is currently a limit of how many open ports are allowed, in total. This is to guarantee receive buffer space for a full queue on each open port. There are no guarantees of message delivery, however. A successful send does not imply packet receipt. The network card may drop a packet, if the receiving port’s queue is full, or there are no more available ports, and the message is not for an open port (with available queue space) the packet will be dropped.

fipc_send() does no fragmentation and sends messages up to ETHERMTU size, less the size of fipc header structure. Similarly, fipc_recv() does no reassembly.

fipc_send() and fipc_recv() are currently only available when using the native mach NE2000 ethernet driver.

Multiboot 0.5 compliant:

Thanks to Erich Boleyn (erich@uruk.org) for his additions to the Multiboot standard, bringing it to revision 0.5, and for updating Mach4 to it. For more information about multiboot, you should read the Multiboot proposal.

Stephen Clawson <sclawson@cs.utah.edu>
March 29, 1996