Mach I486/I386 Installation Notes

Carnegie Mellon University
Pittsburgh, PA 15213
6 July 1994

1. Introduction

The Mach i486/i386 release is a complete system that is compatible with both the Mach 2.5 monolithic kernel and the Mach 3.0 micro kernel and UX single server combination. On IBM PS/2 machines with the microchannel architecture, only Mach 3.0 is supported. For the duration of this document, any references to IBM PS/2 machines refer specifically to PS/2's with the microchannel architecture. This system has an application environment that is close to BSD 4.3-Tahoe. It has been distributed since 1990. Most of the applications provided in this release will run with either the Mach 2.5 or Mach 3.0 kernel. There are a few programs where different versions must be used depending on which kernel you are using. This system can be used to build the Mach 3.0 micro-kernel and Unix server.

The programs and libraries come from the 4.3BSD-TAHOE release. Additionally, we provide the Mach specific programs, include files, and libraries (libmach and libthreads). The CMUCS library and some of the CMUCS programs (most notably rfs and sup) are also included with this release. Lastly, since the Mach I486/I386 2.5 kernel does not support BSD file system labels, most file system tools are CMU tools derived from 4.3BSD.

This release requires the recipient to have a Mach 2.5 or Mach 3.0 license from CMU and BSD4.3 license from University of California which in turn requires either a UNIX/32V, SYSTEM III or SYSTEM V source license from AT&T. Further, we are only allowed to distribute Sun NFS to sites with Sun NFS educational licenses.

All the releases that are made by CMU are done only by an Internet transfer protocol called **SUP** (Software Update Protocol).

Two additional Mach components can be uploaded with this system: Mach 3.0 and MDOS. Mach 3.0 divides the traditional monolithic kernel into a micro kernel and a server program. The latter provides a "personality" to make the whole behave like a particular operating system. MDOS is a DOS emulation package for use under Mach 3.0. It uses the i486/i386 hardware Virtual 8086 mode to run an emulated DOS as a Mach 3.0 task. To find out about MDOS, read the file mdos.ps on mach.cs.cmu.edu in the /usr/mach/public/doc/unpublished.

The initial ***sup*** will put both a Mach 3.0 kernel/server and a Mach 2.5 kernel on the root of the hard disk. By default, the system will boot Mach 3.0. For more information about Mach 3.0, read the documents mach3_sup, mach3_build.{ps,doc} and mach3_setup.{ps,doc} in the Mach public FTP area, mach.cs.cmu.edu, /usr/mach/public/doc/unpublished.

This document assumes a working knowledge of the installation and maintenance of a 4.3BSD system. If you are unfamiliar with the 4.3BSD system, you should read the system maintainers documentation for 4.3BSD.

2. Hardware Requirements for Mach on i386 Machines

We only support machines with:

- either an i386 or i486 processor
- at least 5 Meg of memory. We typically use 8Meg - 16Meg.
- an AT/ISA compatible bus, or IBM PS/2 microchannel bus. EISA buses can be used but currently none of our drivers use the 32 bit functionality of the EISA bus.
- an approved disk controller: (You should get at least 100Meg of disk. We typically use 300Meg drives.)
    - WD 1007 compatible disk controllers.
    - IDE disks (these are WD 1007 compatible).
    - Adaptec 1542B compatible SCSI controller.
    - Adaptec 1542C SCSI controller is supported by Mach 3.0 only.
    - SCSI controllers that make disks appear to the kernel as IDE disks, such as on IBM PS/2's.
    - ST506 controller [Note: they are awfully slow.]
- displays:
    - simple displays in 25 x 80 character mode.
    - Color SVGA support from XFree86 Release 2.0 which supports the following accelerated chipsets:
        - 8514/A (and true clones)
        - ATI Mach8, Mach32
        - Cirrus CLGD5426, CLGD5428
        - S3 86C911, 86C924, 86C801, 86C805, 86C928
        - Western Digital WD90C31
    - And the following SVGA chipsets:
        - Tseng ET3000, ET4000AX, ET4000/W32
        - Western Digital/Paradise PVGA1
        - Western Digital WD90C00, WD90C10, WD90C11, WD90C30, WD90C31
        - Genoa GVGA
        - Trident TVGA8800CS, TVGA8900B, TVGA8900C, TVGA8900CL, TVGA9000
        - ATI 28800-4, 28800-5, 28800-a
        - NCR 77C22, 77C22E
        - Cirrus Logic CLGD5420, CLGD5422, CLGD5424, CLGD5426, CLGD5428, CLGD6205, CLGD6215, CLGD6225, CLGD6235 Compaq AVGA
        - OAK OTI067, OTI077
- high density 3 1/2" and/or 5 1/4" floppies at 1.44Meg and 1.2Meg, respectively.

We support but do not require:

- Intel 80387 hardware floating point. (recommended)
- The built-in serial line: COMi
- The built-in parallel port: LPTi
- ethernet controller boards (non-IBM PS/2):
    - 3COM's EtherLink I and EtherLink II (sometimes known as 3c501 and 3c503),
    - Western Digital's WD8003E and WD8013E ethernet boards.
    - SMC's WD8003E Elite and WD8013E Elite ethernet boards[1])
    - D-Link's Ethernet Pocket LAN Adapter DE-600
    - Intel's iMX586 or PC586/ENET586 ethernet board (Now obsolete)
- The IBM token ring adapter (IBM PS/2)
- Mice
    - Logitech 3-button mouse
    - Logitech Mouseman mouse

[1]Western Digital sold its WD8003 ethernet product to SMC shortly after reengineering the board. This board is called the WD8003 Elite. A recently sampled board seems to work fine, but there have been problems with some early units. (They would stop transmitting after a while.

  - Microsoft 2-button mouse
  - PS2 mouse
- Wangtek 1/4" standard 3M streamer.

WE DO NOT SUPPORT:
- NFS for commercial licensees of NFS

NOTE: While Mach does not require a network interface for normal operation, our method of installation assumes that you can get the files from CMU to the target machine over the internet.

Mach is being used at CMU on numerous i486, i386 and i386sx clones.

3. The Bootstrap Disks

This distribution is intended to be loaded onto an empty disk; it contains all the binaries and sources that are needed. The Mach boot floppies are a pair of High Density (HD) floppy disks. You can write 3 1/2" or 5 1/4" disks for either a Mach 2.5 or Mach 3.0 kernel; the data contained are the same independent of floppy capacity.

The first disk contains a bootstrap program and a copy of the "Mach Kernel". The second disk contains a file system with Unix binaries. These disks are intended to be used not only during the boot process, but also to recover from problems that might prevent you from being able to boot off the hard disk.

The second disk contains (approximately):

```
PORTFILE

bin:
[               dd              ls              setup
cat             echo            mkdir           sh
chmod           ed              mv              stty
cp              hostname        pwd             sync
date            ln              rm              tar

dev:
MAKEDEV and devices

etc:
badblocks       fdisk           ifconfig        passwd          route
badsect         fsck            init            rc              services
clri            group           mkfs            reboot          termcap
diskutil        halt            mknod           restore         umount
dump            hosts           mount           rfshost         vtoc

mnt:
tmp:

usr/bin:
        rfs     sup
usr/games:
        rain
usr/ucb:
        ftp     telnet
```

The installation process presumes that you are going to boot your target i486/i386 machine from the floppy disks that we have provided and then transfer the files from CMU onto the target machine. In this document, we presume that you are going to access the Mach binaries and sources via the internet. **tar** and **dump**/**restore** are also provided, here, to allow you to copy the sources into some other local machine and then to transfer them to the PC by tape, floppies or lan access. There is some cleverness needed to install the system from an intermediary machine. If you plan to do this, refer to Appendix IV for instructions.

You will be requesting binaries and sources from a database machine at CMU using the **sup** database update program/protocol. (See the **sup** manual page). Your machine will authenticate to the database machine; it must explicitly supply an encryption key (CRYPT) in the protocol negotiations.

The first steps in preparing for a internet distribution are:

- Verify that you have a Mach 2.5 license.
- Verify that your site has internet access.
- Verify that your machine has an acceptable network interface.
- Register your machine name, internet address, and CRYPT with CMU. Send this information to mach@cs.cmu.edu. You should chose your own CRYPT, a character string at least 6 characters long or we will assign a random one.
- Specify what machine you wish to FTP the floppy images to if it is different from the machine you plan to SUP to.

You will need to have a machine running a Unix system that is able to **dd** a disk image on to a floppy of the suitable size. You can either **sup** or **ftp** the floppy images from us. The SUP collection is floppy.image. (see the list of SUP collections in Appendix III). For either **sup** or **ftp** you will need to register the name of the machine you will be using with us before you will have access to the images.

4. Before You SUP

4.1. Low Level Disk Formatting

You might need to perform a "low level format" of your hard disk. This procedure partitions the disk into sectors and then looks for bad areas on the disk. It usually takes several hours. Some manufacturers (HP, Toshiba, ...) ship the disks preformatted. Also, some computer stores will format the disk for you before they ship the system. In our experience, ESDI disks generally need to be formatted, IDE disk should never be formatted, and while SCSI disks can be formatted, this seems unnecessary.

The documentation you got with your disk controller or machine or DOS will tell you how to format your disk. The procedures vary widely and are beyond the scope of this discussion. PLEASE DO NOT CALL US IF YOU ARE HAVING PROBLEMS WITH THIS STEP.

4.2. Manual Pages

You should acquaint yourself with the attached manual pages for the programs: **diskutil**, **fdisk**, **vtoc**, **badblocks** and **sup**.

4.3. Booting to Single User mode

In the guide below, we presume that nothing will go wrong. In our experience, 80% - 90% of the installations proceed smoothly, but there is a common set of pitfalls. ***We annotate problem areas with superscripts; these numbers refer to comments in Appendix I.***

The most common problem seems to be corrupting the floppy disks. To avoid doing this always attempt to sync the disks via **sync, reboot** or **halt** before removing the disk or powering off the machine. You are running Unix on these floppies and it does not take well to having its file system yanked out from underneath it. Sometimes the **fsck** command can repair a damaged floppy, but sometimes you can't get that far.

We will presume for the discussion below that you are performing a "simple" installation. In particular, that you do not want to preserve part of the disk for other systems (like DOS, MDOS, or OS/2), you do not have excessive bad blocks in inopportune places (where the system would put the bad block table) and that you do not have to override the disk geometry parameters that are maintained internal to BIOS. We will discuss all these matters in the section DONT CLOBBER MY DISK.

You use the floppies to bring up Mach in "single user" mode. Insert the "Mach Kernel" disk and reset (or power toggle) the machine. The system should soon print out

boot:[1]

At this prompt, you either type a carriage return (or enter), or wait for a "timeout". Then you will see a number printed out, a pause, a "+", etc. The bootstrap is reading in the kernel.

Shortly, you will be instructed to "insert file system". At this point, you remove the "Mach Kernel" disk and insert the "Mach file system" disk. Once you have switched floppies, you proceed with the boot process by typing a carriage return (or enter) [2]. Next, you will see several lines of printout; the Mach kernel describes the hardware that it finds [3]. Then you will see a lot of activity on the floppy as **fsck** is run. And finally, you will get the "#" prompt, indicating that you are executing at "single user" level and may now type commands. You might want to do an

```
ls -R
```

at this point and play with the system a bit. Note: operating/loading from floppies is rather slow and not indicative of the true system performance.

4.4. Disk Types

The kernel prints out many lines of hardware description. If you have an IDE/ESDI/ST506 disk subsystem, or a SCSI subsystem emulating IDE, you should see:

```
hdc0: port = 1f0, spl = 5, pic = 14.
hd0:  stat = 104, spl = 5, pic = 14. (controller 0, slave 0)
```

and maybe a line for one more slave

```
hd1:  stat = 118, spl = 5, pic = 14. (controller 0, slave 1)
```

And if you have a SCSI disk subsystem, you may see:

```
ahac0:port = 300, spl = 5, pic = 11.
aha0: stat = 104, spl = 5, pic = 11. (controller 0, slave 0)
```

and maybe lines for several additional slaves

```
aha1: stat = 118, spl = 5, pic = 11. (controller 0, slave 1)
```

You might even have both types of disk.

Many SCSI controllers, such as those on PS/2's, make SCSI disks appear to the kernel as IDE disks. In this case, your SCSI disk will appear to the kernel as hd0 rather than aha0. If your SCSI disk is called hd0, you must follow these instructions as if it were an IDE disk. Mach on the PS/2's supports SCSI, only through this IDE interface, when the disk controller makes this possible. On non-PS/2's, the kernel may support SCSI directly.

Several of the installation procedures in the rest of this document need to have the disk device indicated. For all of these instances, we supply a "magic" argument *{**D*I*S*K**}*. If you have one IDE/ESDI disk, you type nothing whenever *{**D*I*S*K**}* is indicated. The default is ***/dev/rhd0c***. If you want to install on the second IDE/ESDI drive, you must manually supply ***/dev/rhd1c*** for *{**D*I*S*K**}*. Finally, if you have a SCSI disk, you use ***/dev/rsd0c*** in place of *{**D*I*S*K**}*. For those with only one SCSI disk and who are generally lucky, you might just "rm" ***/dev/rhd0c*** and link it to ***/dev/rsd0c***. I do this.

4.5. Initializing your Disk

Now you use the Mach procedures on the installation floppy to initialize your hard disk.

`type`

diskutil clobber_my_disk *{**D*I*S*K**}*

This initializes a disk for use under Mach. It will destroy all/any data on the disk. If you do not wish to trash all the data on your disk, read the section "DONT CLOBBER MY DISK". On most disks, this procedure takes about 15 minutes per 100Meg. On an ST506 disk, it is much longer. Note: IDE/SCSI disks do not typically report bad blocks; these are handled in the controller. Note: when this operation completes you do not have file systems; you must run the **setup** program (below).

5. Installation and System Setup

5.1. Supping files

Installing Mach is a simple cook book process. You follow the recipe below to bootstrap over the internet.

1. Using **ed**, edit ***/PORTFILE*** and set the crypt= value to be the CRYPT you told CMU[4]. ***/PORTFILE*** is a **sup** control file. You should remove the line starting with bin.cmu unless you are a local CMU site as you will not have permission to get that collection.

2. If you have a SCSI disk, you must edit the ***/bin/setup*** program used below. Using **ed**, edit ***/bin/setup*** and search for

   `disk=hd0`

   and change it to

   `disk=sd0`

   You are done editing.

3. `type`

   setup

   Called with NO arguments "setup" creates a "root" file system and mounts it under ***/mnt***. It then creates a "usr" file system and mounts it under ***/mnt/RFS/.LOCALROOT/usr***. The pathname ***RFS/.LOCALROOT*** is explained further in the section "Root, Super-root and RFS".

4. `type`

   setup <machine name> <internet address> [<internet gateway>]

   Now you should be on the network. You might try to telnet by IP address somewhere, to verify that your network connection is active. The <internet gateway> is necessary for your machine/site to find a path to CMU; it is not necessary at CMU.

5. `type`

   setup sup

   This invokes **sup** on the control file ***/PORTFILE*** [4,5,6]. Your machine will be requesting from CMU all the binaries for the i386 distribution. On our local ethernet, this takes about 1/2 hour; over the internet, we have seen times in the 2 to 3 hour range. If there are failures, just restart this step by

   `retyping`

   setup sup[7]

   Repeat this until everything completes. The **sup** protocol records what it has successfully transferred from previous attempts and transfers any files still required. When **sup** has successfully completed the ***/mnt/RFS/.LOCALROOT*** and ***/mnt/RFS/.LOCALROOT/usr*** partitions on the hard disk will be populated with all the system binaries and there will be the following three directories ***first***, ***bin.root***, and ***bin.usr*** in ***/mnt/RFS/.LOCALROOT/sup***. Each of these directories contains two **sup** client side control files: ***last*** which contains a list of all the files that were transferred and ***when*** which contains the time at which the **sup** was completed.

6. Before booting from your hard disk,

   cd /mnt/RFS/.LOCALROOT/etc

   and edit the file ***netstart*** (use **ed** as the editor). Fill in your machine name, ip address, broadcast address and gateway in the appropriate places. You also should verify that your machine name and ip address are in the file, ***hosts***.

   fstab is setup for an EDSI/IDE disk. Copy ***fstab.scsi*** on top of ***fstab***, if you are using a SCSI disk.

   The file ***/mnt/mach*** is a link to ***mach.MK.STD+WS*** which is a Mach 3.0 micro-kernel. If you would perfer to boot Mach 2.5, change ***/mnt/mach*** to link to ***mach.25*** .

7. Now it is time to reboot off the hard disk.

   `type`

   `reboot`

   After you get the message "syncing disks ... done", and the screen goes blank and it goes back to a bios message remove the floppy disk. If you remove it too soon, it is likely to be in an inconsistant state that will cause it to fail on subsequent uses. If you fail to remove it all the system will attempt to boot off the floppy again -- rather than the hard disk.

8. Note that the template ***/etc/passwd*** file allows root to login with no password. This is so that you will be able to add yourself as a user and change the root password after you boot off the hard disk. YOU SHOULD NOT LEAVE THE ROOT WITHOUT A PASSWORD AFTER YOU HAVE SET UP YOUR MACHINE.

   The login prompt prints a banner indicating that the system is Mach 2.5. This string is defined in /etc/gettytab, and isn't changed automatically even if you are in fact running Mach 3.0.

9. When you reboot, you must create the file system space allocated for the user data. If your disk is less than 150Meg, the ***d*** partition will fill the disk and you are expected to use this space for system files and your personal files. In this case, you should skip the next

paragraphs.

Otherwise, if you are happy with using the remainder of your disk for a single Mach/Unix file system, type

```
eval 'vtoc -m e {**D*I*S*K**}' 4096 512
```

vtoc -m will type to ***stdout*** the beginning of a **mkfs** line to create your "e" partition. You may now mount the "e" partition. You will probably want ***/etc/fstab*** to always mount it. There is a prototype line in the distributed ***/etc/fstab*** that will mount ***/dev/hd0e*** on ***/usr1***; change the ***xx*** to ***rw***. On Mach 2.5 systems, this line will also arrange for the swap file to be on ***/dev/rhd0e***, which we assume is a lot bigger that ***/dev/rhd0d***. (For SCSI, you will find ***/dev/sd0e*** instead of ***/dev/hd0e***.) You will also note that in the ***/etc/fstab*** file there are two lines for ***/dev/hd0d***. The first line has a trailer that indicates that a swap file should be set up on ***/dev/hd0d***. You need to cancel this line (change ***rw*** to ***xx***) and enable the next line (change ***xx*** to ***rw***). This moves the swap file to ***/dev/hd0e***. These comments on paging apply only to Mach 2.5. For information on paging under Mach 3.0, see the section entitled ***Paging under Mach 3.0***. A 12M paging file was created on the ***d*** partition during the SUP. This is probably not large enough and should be increased and perhaps moved to the ***e*** partition.

If you do not want to organize the rest of the disk as one partition, read the **vtoc** manual page. A very prudent thing to do at this time, would be to run the programs **fdisk**, **vtoc**, and **badblocks**. These will print out the various disk "labels". You want to save the data ON PAPER. If something catastrophic ever happens you will be able to reconstruct what your disk looked like.

10. The floppy devices may be wrong for the type of drive on your machine. They are setup for 5 1/4" floppies via the installation process. You make the devices for the 3 1/2" floppy by

    typing

    ```
    cd /dev
    rm -f *floppy *fd0
    ./MAKEDEV f30
    ```

    or for the 5 1/4" floppy by

    typing

    ```
    cd /dev
    rm -f *floppy *fd0
    ./MAKEDEV f50
    ```

    MAKEDEV f31 or **MAKEDEV** f51 can be used to make devices for a second disk drive. It will be known by the names fd1 and rfd1.

11. Edit all the site dependent BSD files below. You should look at the BSD installation manual for exact details.
    - /etc/hosts
    - /etc/rfsd-hosts - a CMU file that lists all hosts that are allowed rfs access
    - /etc/motd
    - /etc/termcap
    - /etc/namedb
    - /etc/resolv.conf
    - /etc/sendmail.{cf,fc}
    - /etc/passwd
    - /etc/group
    - /etc/printcap - if you have printers
    - /etc/nmbroadcast - contains one line with the local broadcast ip address

12. Once the installation is completed to this point, there are still some more files that you might like to get.

In particular initial ***sup*** does not bring over the whole Mach 3.0 binary release. If you intend to run a Mach 3.0 system, you should ***sup*** the ***bin.mach3*** collection.

In /**sup** on your machine, there is a control file ***SUPFILE*** that will allow you to upgrade your binaries to latest version from CMU. Copy the file to ***MYSUPFILE*** and edit in your CRYPT. You will notice that there are several commented out lines at the bottom of the file that you might want to enable.

- The optional collection ***bin.all***, gets you 4Meg - 6Meg of files that are seldom used, like ***/usr/lib/learn*** and ***/usr/lib/*_p.a***. If you are concerned about disk space, you probably do not want this collection.
- For Color X11R5, there are two choices: ***bin.X11R5*** is the whole 36Meg distribution. The four sup collections, ***bin.X11R5base, bin.X11R5+fonts, bin.X11R5+inc, bin.X11R5+lib***, will also bring over the same stuff. ***bin.X11R5base*** is the "necessary and sufficient" 25Meg subset. ***bin.X11R5+fonts*** are the 100dpi fonts and the huge ***misc*** font files. The +inc and +lib should be obvious.
- ***bin.mach3*** will bring over binaries into ***/usr/mach3*** that you need when running Mach 3.0. This also corresponds to the mach3.release collection with release=i386. There are two lines in SUPFILE corresponding to the bin.mach3 collection. You need to make sure the right line for you has no # as the first character and the other one begins with a #. The # character marks the rest of the line as a comment. IBM PS/2 users should uncomment the line containing the string ***release=ps2***, while other users should use the other line. The ***release=ps2*** option specifies that you want PS/2 versions of the kernel and server.
- mdos will bring over the sources for the MDOS dos emulation package.

After you have edited your sup control file, you run:

```
sup -dov MYSUPFILE
```

It will only take a few minutes to verify you are up to date; if you need to bring over a few changed programs, it will take just slightly longer.

One directory that **sup** brings over, ***/usr/release***, is worth noting. In it there is a file ***NOTES***, which tells you what has changed in the current release; actually is an archive of all the change histories. Also in the directory is ***mach-afs-nfs***, which is a binary of the latest released Mach 2.5 kernel. If you copy it into ***mach*** and reboot, you will be running the latest distributed 2.5 kernel.

The Mach 3.0 kernel and UX server are distributed but not installed by the bin.mach3 collection. This collection has releases for ISA bus kernels and a PS/2 release for kernels that support the microchannel bus. See the comments above regarding the ***release=ps2*** option. The kernel and severs need to be copied from ***usr/mach3/mach_servers*** to the appropriate locations on the root partition before they will be used for booting. See the **Mach3 Setup** document for details.

13. Instructions for Supping the sources can be found in Appendix III.

5.2. Daemons

/etc/rc and ***/etc/rc.local*** will select the daemons you need for:

- NO network
- a network
- a 2.5 kernel
- a 3.0 kernel

To determine whether to start the network daemons, we evaluate the predicate **/bin/inet**. (For example, you might have a laptop that is not connected to the network.) If it returns FALSE, we presume there is NO network. If it returns TRUE, there is. A simple stratagem is to copy **/bin/false** to **/bin/inet** if you DO NOT want network services; copy **/bin/true** if you DO want them.

To determine whether the kernel is 3.0 Mach, we look for and then evaluate the predicate **/usr/mach3/bin/mach3**. If it returns FALSE, we presume the kernel is 2.5 Mach. If it returns TRUE, the kernel is 3.0 Mach. Some of the daemons are different for each system; if you are curious look at ***/etc/rc.local***. NOTE: we assume that you have placed the "3.0 release subtree" in ***/usr/mach3***.

5.3. X11R5 and Mice

5.3.1. X11R5

Color X11R5 currently is not supported on IBM PS/2's. If you're not using a PS/2, you probably want to use X11R5 with new SVGA cards and one megabyte of video memory. The X11R5 we release is XFree86 Release 2.0.

The X11R5 binaries are the ones we use at CMU. Our support organization has chosen to put X11R5 in ***/usr/misc/.X11***, rather than the traditional ***/usr/X11*** or ***/usr/X386***. This means when you sup X11R5 it will be installed in ***/usr/misc/.X11*** and you need to put ***/usr/misc/.X11*** on your path.

Now for the hard part. Read:

- ***/usr/misc/.X11/lib/etc/README***,
- ***/usr/misc/.X11/lib/etc/README.mach***

and whatever else you can find about the X11R5 SVGA server. One of the truly hard/magical parts of the server is coming up with a ***Xconfig*** file to describe your monitor/svga card. The X11R5 documentation describes the process in gory detail, at least twice. In ***/usr/misc/.X11/lib***, you will find some Xconfig's I have played with:

```
XconfigGWa{,3} for the Diamond Speed Star and Crystal Scan 1024NI
XconfigGWb{,3} for the Diamond Speed Star and NEC 4FG
XconfigINa{,3} for the Cardinal 700 with NEC 4FG
Xconfig256k{,3}for generic cards with 256k of memory.
```

The "3" suffix is used if you have a 3 button mouse. Lastly, if you specify a fixed chipset and clock like these configurations do, I believe that any mach kernel will work. BUT you should really be at X142F or MK76/UX36 if you want to have X11R5 svga card "probing" work.

5.3.2. Mice

Mach supports several kinds of external mice, the logitech three button mouse, the microsoft two button mouse, the PS2 mouse and the logitech mousman mouse. We do not support any kind of bus mouse. The way you indicate the mouse type and com port is a bit magical; you set the mouse device minor. The value you set the minor to is computed by the formula:

```
<type> * 8 + <com port>
```

<com port> is 0, or 1 or ... depending on which com port you plug your mouse into. (Some architectures prefer to make com0 the modem line.)

<type> and <minor> are:

Mouse Name	<type value>	<com0 minor>	<com1 minor>
logitech mouse	0	0	1
microsoft mouse	1	8	9
ps2 mouse	2	16	17
no real mouse at all	3	24	25

logitech mouse man (M+ or V type)	**4**	**32**	**33**

We have implemented a neat hack for small notebooks so that they don't have to be saddled with a HUGE mouse. Scroll lock toggles a mode wherein "seldom used 'function keys'" are used to represent mouse actions. F1, F2, F3 are the left, middle, and right mouse button. The first key stroke represents a down click and the second key stroke represents an up click for the mouse button. Then the up down, right, left, arrow keys actually move the mouse. And if you have a standard pad with home, pageup, end, and pagedown filling in the square around the up, down, right, and left arrow keys then the former set do the obvious and move the mouse along the diagonals.

5.4. The Environment

Mach 2.5 I386 has the same kernel features as the Mach 2.5 for all the other supported platforms. The software environment starts with 4.3BSD-TAHOE environment, with one major exception. The Mach 2.5 kernel does not support BSD file system labels; so most file system tools are CMU tools derived from 4.3BSD.

You are also provided with all the Mach programs, include files and libraries and the CMUCS library and some programs (most notably **rfs** and **sup**). However, these files are not in the ***/usr/mach*** tree or ***/usr/cs*** tree as is the case with Mach on other hardware, but are integrated into the BSD tree.

The compile environment/tools, i.e., **cc**, **as**, **ld**, **size**, **nm**, **strip**, **ar**, and **gdb** all come from the GNU products of the Free Software Foundation (FSF). There is one exception, **cpp**; we use a CMU modified 4.3BSD **cpp**. All of the program and library sources have been compiled without using the "traditional" switch and only a few programs needed "writable-strings". You should be aware of what these two features do; see the gcc manual page. You may need them to successfully port your applications.

Finally, one feature about the load/execution function should be noted. Program text starts at 0x10000 (64K). This means that you can not dereference a pointer to 0 or any small integer value. This will cause programs to core dump that were "accidentally" working on other architectures. Only a small number of 4.3BSD-TAHOE programs that had to be fixed to deal with this. (SunOS and Mach on the Sun Architecture machines have a similar restriction.)

5.5. BSD4.3-TAHOE Source Compilation

Since the system sources, ***/usr/src***, are basically 4.3BSD-TAHOE, the BSD standard compilation procedures applies. The typical way to build a program is to go to the appropriate directory and type **make**. You install it by typing **make install**. Note: If the program <foo>.cmucs exists, it was used in preference to the original BSD program <foo>. Also, the compile tools mentioned above, should be taken from ***/usr/src/gnu***; the BSD version has been left for reference purposes.

5.6. Building Kernels

This section tells you how to build Mach 2.5 kernels. Different procedures apply to Mach 3.0. See the document **Building Mach 3.0** (mach3_build.ps).

First, you must decide what kernel to build. The mach.sys sup collection retrieves a rather old Mach

2.5 kernel. This corresponds to the functionality in the official Mach 2.5 release. The mach.mainline (and mach.nfs) collections retrieve essentially what is actively used at CMU today.

To build a new kernel, go to the kernel source tree. Then type

```
make CONFIG=STD+WS-afs-nfs doconf config
```

This will create a directory, STD+WS-afs-nfs. You can go into this directory and type **./make**. This will initiate a standard BSD style kernel build.

5.7. Installing/Booting Kernels

This section applies to Mach 2.5 only. Different procedures apply to Mach 3.0. See the document ***Setup for Mach 3.0*** (mach3_setup.ps).

The kernel that you have built should be placed on the root of the file system. Give it some name other than ***/mach*** [or ***/vmunix***], initially, so you can fall back to the old kernel if necessary. To invoke your kernel

type

```
/etc/reboot
```

The old kernel will shutdown, then the hardware performs self test, and lastly, you are prompted for a kernel to boot. The system types out:

```
boot:
```

If you type nothing, ***mach*** is booted. (If you type carriage return (or enter), ***mach*** is also booted.) You may type a file name of a kernel to boot. If you had put your kernel in the file ***/mach.new***, you would type

```
/RFS/.LOCALROOT/mach.new
```

to boot this kernel. Note: From the section on Root, Super-root and RFS, you should understand why the ***/RFS/.LOCALROOT*** is necessary. Aside: you may also supply flags on the boot line, above.

- ***-a*** causes the kernel to ask for the file system root partition.
- ***-s*** causes the system to boot "single user".
- ***-d*** causes symbols for the debugger to be loaded.

Once you are happy with your kernel, you want to install it as ***/mach***. The old ***/mach*** has a link count of three; the three files ***/mach***, ***/vmunix*** and ***/../../mach*** have to be linked together. You also want to save the old kernel. My preference for doing this is to:

```
cp -p /mach /mach.old
cp -p /mach.new /mach
```

This way after you set up the links once and never worry about them again. Alternatively, you might

```
rm -f /mach.old  /../../mach.old /../../vmunix.old

mv /mach /mach.old
mv /vmunix /vmunix.old
mv /../../mach /../../mach.old

mv /mach.new /mach
ln /mach /vmunix
ln /mach /../..
```

5.8. Root, Super-Root and RFS

Since 1982, CMUCS kernels have had a facility that lets you access files and devices on other machines. We call this the remote file system, RFS. The client side exists in every Mach kernel; the server is a privileged program, **rfsd**, that is run on each machine that will permit access to its local files. We have even ported the server to non-Mach operating systems.

RFS refers to remote files/devices in a location dependent way. The remote file name must not look strange to Mach/Unix programs that parse file names. Thus, the generic name is

```
                /../<remote machine name>/<path on that machine>
for example
                /../wb1/etc/passwd
                /../wb1/dev/rmt0
```

What this implies is that in Mach, the ***root*** file system, "***/***", is not inode 2. The kernel automatically "chroots" (changes the root) to ***/RFS/.LOCALROOT*** as it boots. Thus "***/..***" and "***/../..***" are well defined and not equal to "***/***".

The local machine must have an "RFS link" to the remote machine. To make a link in ***/..***, type:

```
                cd /..

                rfshost <IP ADDRESS> <host name> <host nic names>
For example:
                rfshost 128.2.250.16 wb1.cs.cmu.edu wb1
or even
                rfshost ‘grep wb1.cs.cmu.edu /etc/hosts‘
```

You may create as many entries as you like and as many aliases as you like. A special quoting mechanism is necessary to remove an RFS link. If you are in the directory containing the RFS link, ***link***,

```
                rm .///<link>
```

will remove the RFS link, ***link***.

By default, RFS accesses files on the server using the user id, **rfsd**, and group id, **system**. However, the server must be told that your machine may access its files. The file ***/etc/rfsd-hosts***, on the server, should contain single lines indicating the internet address of machines allowed to contact this machine. (If you change the file, you must kill and restart **rfsd**.)

You can also authenticate yourself to RFS, using the program, **rfs**. **rfs** will prompt you for a user, group, account, password, and "Identify?" Give the default answer for account and "Identify". After running **rfs**, you will have a shell that will allow you to access files on any remote RFS hosts using the specified uid and group. Note: you must create a local magic file for authentication to work. You must type

```
                cd /..
                rfshost 0.0.0.0 CONTROL
```

5.9. Paging under Mach 2.5

The Mach 2.5 kernel pages out to the file named ***pagingfile*** on the disk partition specified in the ***/etc/fstab*** file.

5.10. Paging under Mach 3.0

The file ***/mach_servers/paging_file*** is a symbolic link to a file on a filesystem other than the root, which the kernel uses as paging space. The kernel interprets the destination of the link in a special way. The default value is

```
paging_file -> /dev/hd0d/paging_file
```

The syntax ***/dev/hd0d/paging_file*** refers to the file ***paging_file*** on the root of the filesystem on ***/dev/hd0d***. Since the Mach 3.0 kernel contains no file system code it cannot extend the size of its paging file. The paging file created by the setup script is 12 megaytes. If you have enough disk space you should increase this to between 20 and 40 megabytes.

You may want to move the paging file to your partition on ***/dev/hd0e***. To do this you must follow four steps: create the new file, make the link in ***/mach_servers*** point to the new paging file, reboot your system, and finally remove the original paging file. The following sequence of steps will create a 20 megabyte paging file on ***/dev/hd0e*** and remove the default paging file on ***/dev/hd0d***. The example assumes that ***/dev/hd0e*** is mounted on ***/usr1*** and ***/dev/hd0d*** is mounted on ***/usr***.

```
cd /mach_servers
dd if=/dev/rhd0c of=/usr1/paging_file bs=256k count=80
chmod 600 /usr1/paging_file
rm paging_file
ln -s /dev/hd0e/paging_file
reboot
<when system reboots and you log in again in as root>
rm /usr/paging_file
```

Take care never to remove a paging file that the kernel is currently paging on. In the above example, we removed the link to the paging file; this is safe. It would not have been safe to remove ***/usr/paging_file*** before rebooting.

5.11. Changing Time Zones

As distributed the time zone is set to eastern standard time, i.e. EST5EDT. If you need to change time zones you will need to perform two tasks. First, you will need to re-link ***/etc/zoneinfo/localtime*** as follows:

```
login as root
cd  /etc/zoneinfo
rm  localtime
ln  xxx  localtime   (substitute appropriate timezone file for xxx)
```

Second, you will need to patch time zone variable (i.e. _tz) in the kernel. Prior to doing this you will need to calculate the appropriate patch value in hex of the difference in minutes between your local zone time and GMT (universal time). The value of this constant for eastern standard time, for example is x12c hex (i.e. 300 decimal). This is the value that you will initially see for _tz. To change to Pacific Daylight Time, as an example, _tz should be patched to x1e0 hex (i.e. 480 decimal).

5.12. The Kernel Debugger

Mach 2.5 has a version of "adb" in the kernel. The key sequence "control-alt d" will enter the kernel debugger. As you would expect, ":c" gets you back. Once in the kernel debugger, you are on your own. You might look at the BSD documentation on **adb** and **kdb** for a list of the commands. A laconic description of the debugger may be found in Appendix V. Symbols have to be loaded with the kernel, for **kdb** to be able to use them. If the kernel file is executable (determined by the file's permission bits), the boot program always loads the symbols. The boot line flag, ***-d***, will force symbols to be loaded.

6. Replicating The System

NOTE: If you have one or two local machines, we will allow you to upgrade them directly from CMU. But if you have a larger site, you must take an approach described in this section.

6.1. Cloning

The easiest way to mass produce systems is to set up one machine as a "cloning" machine. Below, we assume you are cloning onto your second ESDI/IDE drive, ***/dev/rhd1c***. If you are using SCSI, you would substitute ***/dev/rsd1c*** wherever ***/dev/rhd1c*** is specified. (And you would use ***/boot.sd*** rather than ***/boot.hd***.)

The recipe follows:

- bring your disk over to cloning machine
- low level format the new disk
- run "clobber_my_disk" on the new disk (**diskutil clobber_my_disk /dev/rhd1c**)
- fill the disk:
    - eval ‘vtoc -m a /dev/rhd1c‘ 4096 512
    - fsck /dev/rhd1a
    - mount /dev/hd1a /mnt
    - mkdir /mnt/RFS
    - mkdir /mnt/RFS/.LOCALROOT
    - mkdir /mnt/RFS/.LOCALROOT/usr
    - eval ‘vtoc -m d /dev/rhd1c ‘ 4096 512
    - fsck /dev/rhd1d
    - mount /dev/hd1d /mnt/RFS/.LOCALROOT/usr
- copy everything you need from the cloning disk to the new disk.
- plug your disk back into your own machine

Clearly, you do not have to mount both file systems at the same time. What is critical is that you perform the **vtoc** commands and that on the ***root*** file system, you create ***RFS/.LOCALROOT*** and change directory there before you begin copying data.

You may use any procedure that works under BSD to copy the cloning disk. You could "**dump|restore**" ***root*** and ***usr*** from the one disk to the new disk. You could "**tar** crf**|tar** xf". You could load via a streamer into the new disk. Follow whichever procedure you are most comfortable with.

These procedures do not install a boot block on the disk. To do this type:

```
dd if=/boot.hd of=/dev/rhd1c bs=b
```

This will copy an image of the boot program, which is stored in ***/boot.hd*** to the boot area on the new disk. It occupies roughly 9 sectors.

6.2. Using Sup

You might want to set up a **sup** repository at your site that is upgraded from CMU on a regular basis. You would then add site dependent information and local improvements to this base. Then local machines would **sup** from this local repository/server. The information in the ***/usr/mach/public/sup*** directory on mach.cs.cmu.edu provides an example of how to set up your local site; refer to the section Sup'ing Sources. We advocate **sup** because it allows you to propagate to each machine at your site only the changes in the repository that have happened since the machine last was upgraded by **sup**. This allows ALL machines to keep current with a minimum overhead. **sup** does all of the bookkeeping.

6.3. Floppies

The Mach 2.5 kernel sources have the tools to produce customized 3 1/2" and 5 1/4" floppies. Two scripts are in the ***standi386at/binaries*** directory. **MKfs** will make the file system floppy, and **MKkernel** will make the kernel floppy. Both take a single argument of "3h" or "5h" which specifies the floppy size as 3 1/2" or 5 1/4", respectively. You must run these commands as "root" and the floppies must have been previously formatted.

You can make Mach 3.0 boot floppies using the scripts **MK3fs** and **MK3kernel**.

More simply, you can just copy the floppy images you got for the original installation.

7. DONT CLOBBER MY DISK

7.1. Minor Variations

Three simple cases of disk setup arise that can still be handled by

diskutil clobber_my_disk {**D*I*S*K**} .

In the first case, you might need/want to override what BIOS believes the disk geometry to be. BIOS allows only 10 bits of cylinder numbers. If you have more, either you can not use them, or you have to lie to BIOS about the number of sectors per track and number of heads so that the "virtual" number of cylinders falls under 1024. If you specify the "-p" option to **diskutil clobber_my_disk**, it will interactively prompt you for the true geometry information.

In the second case, you have bad blocks in what would be the bad block area. **diskutil clobber_my_disk** will abort very early on. You fix this by moving the start of the "Mach (BIOS) partition" forward (see section The Disk Layout below). You type:

`diskutil clobber_my_disk -o <offset in sectors>` {**D*I*S*K**}

The Mach (BIOS) partition will start at <offset in sectors> on the disk. This quantity should be an integral number of cylinders. NOTE: You specify the number in SECTORS.

In the last case, you might have a stupid low level format program which marks all the sectors of a track bad if any sector is bad. **diskutil clobber_my_disk** has a heuristic; it looks for several sector errors before it labels the track as bad. This will cause a lot of needless rattling of the disk (error retries and recalibration), before the track is marked bad by **diskutil clobber_my_disk**. The "-t" option to **diskutil clobber_by_disk** (and **badblocks**) will cause the program to mark the entire track bad if there is one bad sector.

Lastly, you can combine any or all of the options described above.

7.2. Room Enough for DOS or MDOS

If you are starting out with an empty disk and want to leave room for a DOS partition at the front of the disk, there is a very easy way to do this. Use

diskutil clobber_my_disk -o <offset in sectors> {**D*I*S*K**}

as noted in the previous section and specify the number of sectors you want to leave for the DOS partition. (This should be a integral number of cylinders.)

Next, you boot up DOS from a floppy and run DOS's **fdisk** to create a primary DOS partition that uses ONLY the initial cylinders of the disk that you have set aside. Then you run DOS **format** to format the DOS ***c*** drive. Lastly, you populate the ***c*** drive. NOTE: in our experience, using **fdisk** from DOS 3.3 or DOS 5.0, it is very easy to allocate the right number of cylinders. DOS 4.1 **fdisk** is harder to use for this purpose.

MDOS (the Mach DOS emulation server) does not actually require a DOS partition to run. It will read DOS application programs out of the Unix partitions. But is is easier to load software from DOS floppies to a real DOS partition than to a Unix partition. If you think that you might want to run MDOS eventually, you should look at the mdos_info document now and decide if you want to leave room for a DOS partition.

If DOS partitions are already on the disk, the next section will explain what you need to do.

7.3. The Disk Layout

At the start of the disk (sector 0), there is a BIOS boot block that has code and a partition table. (NOTE: BIOS calls this table a partition table. Unix refers to regions of the disk that are separate file systems as partitions. To try to avoid the obvious confusion, we call to the former a "(BIOS) partition", and the latter a partition.) The (BIOS) partition table has four entries; one is designated as "active". The boot process finds the active partition and invokes it. This table makes it possible to divide the disk up into several disjoint areas. Thus Mach and DOS can both coexist on the same disk.

The "Mach (BIOS) partition" begins with a boot program that can occupy up to 29 sectors. This is followed by one sector for a ***vtoc*** (volume table of contents). The ***vtoc*** specifies the override disk geometry and then divides the "Mach (BIOS) partition" into the standard Mach/Unix file system partitions. The ***vtoc*** is followed by four sectors that record bad blocks and a number of sectors that are used as bad block replacements. At the next cylinder boundary, the first Mach partition begins; this is typically the "root".

All the structures described above need to be initialized for Mach to be able to use the disk. The programs, **fdisk**, **vtoc**, and **badblocks**, edit the necessary structures. They will let you edit every field in every structure. We will not describe them in detail; see the manual pages for more information.

We also provide a simpler interface:

diskutil clobber_my_disk {**D*I*S*K**}

This initializes the structures to use the whole disk for Mach. It is equivalent to the following three commands:

```
diskutil finit {**D*I*S*K**}
badblocks -w {**D*I*S*K**}
diskutil fs {**D*I*S*K**}
```

The first step, **diskutil finit**, does a quick and basic initialization of the structures to an empty state. Then, **badblocks** scans the disk for bad spots. Finally, **diskutil fs** puts the basic Mach partitions, "a" - "e" on the disk. (This uses the Mach default layout.)

You may replace **diskutil finit** with **diskutil init**. This DOES NOT initialize the sector 0 disk layout information. You MUST use **fdisk** to pick a "(BIOS) partition" for Mach, before you run this alternate sequence. You should put the sector start address for the "Mach (BIOS) partition" on a cylinder boundary. You follow this method, if the disk already has (BIOS) partitions that you need to preserve.

8. Known Features/Problems/Bugs

8.1. gas

To be compatible with the AT&T assembler, the gas assembler has been generated to accept "#" and "/" as comment characters. This means that, at present, there is no way to indicate division in assembler expressions.

9. Mach Support

Since the Mach project is no longer an active reseach project at CMU, most of the support and futher developement of Mach has to come from the community of users. This community can be reached through two mailing lists that are maintained at CMU and the NetNews BB comp.os.mach.

The two mailing lists to which you may wish to be added are:

machi386@cs.cmu.edu
: covers questions related to running Mach on i386/i486 hardware.

mach3@cs.cmu.edu
: covers topics of more general interest about Mach 3.0.

Subscription to both of these lists is handled by the majordomo program. For information on how to use this interface send mail to majordomo-mach@cs.cmu.edu with the word "help" in the body of the message.

I. Trouble Shooting the Installation

1. One BIOS never gets this far. We would suggest that you get either an AMI or PHOENIX BIOS as a replacement. (We believe that the new boot.fd code of Mar 1, 1991 has solved this problem. If you are still seeing it, please contact us at mach@cs.cmu.edu)

2. Occasionally, people type an extra carriage return which is gobbled down and you are not given an opportunity to put in the file system disk. The kernel will panic shortly and you will have an opportunity to try again. You must use both disks to load Mach.

3. It is very important that one line in the hardware description contains your ethernet hardware address. If you do not see such a line, then the kernel has not recognized your ethernet board. Either the board is unacceptable or the jumper settings are wrong. Refer to Appendix II to get the appropriate jumper settings.

4. Be careful not to delete the `prefix=/usr` at the end of the `bin.usr` line in **PORTFILE**. If you do, **sup** will try to load all the files in the *bin.usr* collection on to the root partition and will run out of space.

5. During the sup of the first collection several commands are executed that can cause trouble. The 16-sector boot program (sd.boot or hd.boot) is written on the the beginning of your hard drive. Only one of these attempts will succeed, depending on what sort of hard disk you have. After that a 12M paging file is created on the d partition of the hard drive.

6. If you get an access error back from **sup**, we might not have put your site/machine into our authentication table yet. More likely is that your installation's name server does not respond when our site queries it, asking it to translate your machine's IP address to its name. Please send a copy of the supfile command and command line that you are using and the transcript of the **sup** session to mach@cs.cmu.edu.

7. **sup** has a very long timeout. It takes minutes for it to realize that a connection has closed. If you are connecting to us from some far away place, you might want to run **sup** or **"setup sup"** in the background. You NEVER want to power cycle the machine; since you are in the process of loading your machine, it is VERY likely that the buffer pool has not been flushed to disk and you will see a lot of inconsistencies when you next **fsck**

8. If the system crashed or you had to reboot before **sup** completed, you must start the setup phase from scratch. Call setup with no arguments, which will make new file systems. Otherwise there may be files on your system which are corrupt, but which **sup** thinks are ok and will not replace.

I.1. A Few More Anomalies

- If you can not low level format your disk, talk to the manufacturer or sales office. This is a routine "BIOS/DOS" procedure. We can not help you.

- If you personally installed a 80387, make sure that there are not switches to change or a setup program, to run to tell the machine about the fpu.

- Unless you are running X, Mach will use a "monochrome 80 x 25" mode of character display. Some vga cards misbehave; you might be better off with a simple monochrome board if you have one. Also, be sure you have initialized your vga board for this mode; some boards require special setup.

- Believe it or not, we have found at least one keyboard that does not work with Mach -- it does work with DOS. What will happen is that after you get the single user prompt on the floppy boot, you will not be able to type.

II. ETHERNET

In the presentation below, you will see that we prefer IRQ 9 for the ethernet interrupt level. On some 8 bit cards, there is no option for IRQ 9; you should use IRQ 2 instead -- which just happens to map to IRQ 9 on a 16 bit AT BUS machine.

II.1. WD8003 & Etherlink II & WD8003 Elite

The first two of these boards use the NS 8390 ethernet chip; they are programmed "basically" the same with a few localized differences to account for the board logic. The new SMC WD8003 Elite uses custom logic and is described later.

II.1.1. EtherLink II

This board has nicely labeled jumpers. There are only two jumpers: one for the memory address and one for the port. You may use either 0xd8000 or 0xdc000 for the memory address. But 0xdc000 is used by the Adaptec 1542B SCSI controller to map its ROM BIOS, so for SCSI configurations you should ONLY use 0xd8000. The thick/thin ether setting and IRQ are made by software. We use the port address to imply a setting of IRQ and thick vs thin ether.

PORT	IRQ	ETHER TRANSCEIVER
0x280	9	thin
0x2A0	9	thick
0x2E0	5	thin
0x300	5	thick

II.1.2. WD8003 & WD8013

We can accept:

port	irq	memory
0x280	9	0xd0000
0x2A0	9	0xd0000
0x2E0	5	0xd0000
0x300	5	0xd0000

The WD8013 is the 16 bit card. It is very nicely laid out and all the jumpers are labeled. Select a set of options consistent with the above table.

For the WD8003, there are five sets of jumpers that must be set. A schematic of the board jumpers is shown later with one acceptable/preferred jumper arrangement. NOTE: with the DELL 320SLT, you must use a setting with IRQ 5.

W1: Sets the Port address. We use 280 which is indicated in the figure below. If you interpret (jumper) shorts as 1 then the map of 16 possible port ranges is given below: (The *'d ports are the ones we support.)

```
port            0x200   0x220   0x240   0x260
pattern         0xF     0x7     0xB     0x3

port            0x280*  0x2A0*  0x2C0   0x2E0*
pattern         0xD     0x5     0x9     0x1

port            0x300*  0x320   0x340   0x360
pattern         0xE     0x6     0xA     0x2
```

port	0x380	0x3A0	0x3C0	0x3E0
pattern	0xC	0x4	0x8	0x0

The jumper labeled 2 in the W1 set, reduces the number of board wait states from 4 to 2.

W2: selects the IRQ level that the board is to use. We use IRQ2 that gets remapped to IRQ9 on ISA bus machines. The possibilties are:

```
jumper 11 -> IRQ2                jumper 9 -> IRQ3
jumper  7 -> IRQ5                jumper 5 -> IRQ5
jumper  3 -> IRQ6                jumper 1 -> IRQ7
```

W3: Is shorted for thin ether, otherwise all the jumpers are open. w4: Is open for thin ether, otherwise shorted. w5: Is only meaningful for thin ether and we wanted it shorted.

Below is a picture of the board and jumpers set up for thin ether.

```
                  W4

                                                                E
                                                                T
        2468a W1                _ W3                            H
         || |                   _                               E
                                _                               R
                                _
                                _                               T
                                _                               A
                                                                P
                                                                S
                                                |  W5

                b97531 W2
                |
```

II.1.3. WD8003 Elite

This is a reworking of the WD8003. It is necessary to read its documentation and use the DOS setup utility that is provided with it. Choose a port/memory address that is compatible with the table below.

port	irq	memory
0x280	9	0xd0000
0x2A0	9	0xd0000
0x2E0	5	0xd0000
0x300	5	0xd0000

II.2. Etherlink I

We can accept:

port	irq
0x300	2

A schematic of the board jumpers is shown below with one acceptible/prefered jumper arrangement.

```
I/O ADDR        MEM
4 5 6 7 8 9     EN

. . . . X X      X
X X X X X X      X
X X X X . .      .


MEM     1 1 1 1 1 1 1 1
ADDR    2 3 4 5 6 7 8 9

        . . X X . X X X
        X X X X X X X X
        X X . . X . . .


INTERRUPT           DMA
2 3 4 5 6 7     1 2 3 1 2 3
|               |     |
```

II.3. PC586

We can accept:

PORT	IRQ	MEMORY
0x0d0000	9	0x0d0000
0x0c0000	5	0x0c0000
0xf00000	12	0xf00000

A schematic of the board jumpers is shown below with one acceptible/prefered jumper arrangement. NOTE: This is not the specification for an iMX586 which is the equivalent board but layed out differently.

Jumper 1: E27-E32 selects > 8 Mhz bus.

The board static ram must be jumpered to address 0xD0000

```
Jumper 2: E28-E33
Jumper 3: E29-E24
Jumper 4: E30-E25
Jumper 5: E31-E26
```

Jumper 6 thru jumper 13 choose one: The board must be configured for interrupt request line 9. This is jumper 10 set to E41-E49.

Jumper 14: E54-E55 selects 0WS mode.

Jumper 15: is left alone.

Jumpers 16 thru Jumpers 21: E1-E18 control whether the board uses cheapnet or standard ethernet. The figure below shows it using cheapernet. Move the jumpers down for standard ethernet.

The figure below visually shows the jumper pins of interest. Those pins marked with X's are the jumpered pins. E56 --- E58 may be missing from older cards.

```
                                        J  J  J  J  J  J
                                        1  1  1  1  2  2
                                        6  7  8  9  0  1

                                     E1 X  X  X  X  X  X E6
                                     E7 X  X  X  X  X  X E12
                                    E13 .  .  .  .  .  . E18
 J15

X E19
X E20
. E21
                        J  J  J  J  J
                        1  2  3  4  5

              E56 X     .  .  X  X  X E22-E26
              E57 X     X  X  X  X  X E27-E31
              E58 .     X  X  .  .  . E32-E36
               .  .  .  .  X  .  .  . E37-E44
               .  .  .  .  X  .  .  . E45-E52

               J  J  J  J  J  J  J  J
               0  0  0  0  1  1  1  1
               6  7  8  9  0  1  2  3

                                        J14

                                E53 .  X  X E55
```

III. Sup Collections

III.1. Binary Collections

Edit the control file below, replacing CRYPT with your crypt. You choose among the optional collections as was explained earlier.

```
bin.root host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
bin.usr host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT prefix=usr
#bin.cmu host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.all host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R4 host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R4small host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R5 host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R5base host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R5+fonts host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R5+inc host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.X11R5+lib host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.mach3 host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
#bin.mach3 host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT release=ps2
#mdos host=x29.mach.cs.cmu.edu hostbase=/i386 base=<BASE> crypt=mdos
#floppy.image host=x29.mach.cs.cmu.edu hostbase=/i386 base=/ crypt=CRYPT
```

There are two control lines above applying to the ***bin.mach3*** collection. You need to make sure the right line for you has no # as the first character and the other one begins with a #. The # character marks the rest of the line as a comment. IBM PS/2 users should uncomment the line containing the string ***release=ps2***, while other users should use the other line. The ***release=ps2*** option specifies that you want PS/2 versions of the kernel and server.

NOTE: the bin.cmu collection is only available for machines at CMU.

III.2. Source Collections

The source kit for Mach includes all of 4.3BSD-TAHOE, plus all the Mach group's sources, plus some of the CMUCS sources, and several of the FSF (gnu) components. These sources will take up about 100Meg of disk space. You will need about another 40Meg, if you want to compile them.

Edit the control file below (which can be found in /sup/SUPFILE), replacing CRYPT with your crypt. For each line in the control file, create the top level directory in which the data will be stored and set the permissions appropriately. (Change to the ***base*** directory (/usr/src) and **mkdir** each ***prefix*** directory.)

type

```
        sup -dov <control file>
```

Repeat this command until **sup** completes successfully. Note: It will probably take several attempts and you probably want to do this at night. Each, the src.<dir> collection below, brings over the BSD/MACH sources necessary to build the programs in the <dir> directory. For example, src.etc is the sources for the ***/etc*** directory and src.usr.lib is obviously the sources for the ***/usr/lib*** directory. src.gnu contains the gnu sources used to build the binary tools and the compiler. Lastly, we have the mach.<dir> collections for kernel sources. mach.sys contains sources for the Mach 2.5 official release. While mach.mainline and mach.nfs contain kernel sources for essentially the most recent monolithic Mach kernel used at CMU.

```
src.MISC host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src crypt=CRYPT
src.bin host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=bin crypt=CRYPT
src.cci host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=cci crypt=CRYPT
src.doc host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=doc crypt=CRYPT
src.etc host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=etc crypt=CRYPT
src.games host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=games crypt=CRYPT
src.include host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=include crypt=CRYPT
src.lib host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=lib crypt=CRYPT
src.man host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=man crypt=CRYPT
src.new host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=new crypt=CRYPT
src.old host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=old crypt=CRYPT
src.tests host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=tests crypt=CRYPT
src.ucb host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=ucb crypt=CRYPT
src.undoc host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=undoc crypt=CRYPT
src.usr.bin host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=usr.bin crypt=CRYPT
src.gnu host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=gnu crypt=CRYPT
src.usr.lib host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=usr.lib crypt=CRYPT
src.nfs host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=nfs crypt=CRYPT
mach.sys host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=sys crypt=CRYPT
mach.mainline host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=sys crypt=CRYPT
mach.nfs host=x29.mach.cs.cmu.edu hostbase=/i386 base=/usr/src prefix=sys crypt=CRYPT
```

NOTE: the src.nfs and mach.nfs collections are only available to sites with Sun NFS educational licenses.

The collections for the Mach 3.0 sources are described in our public FTP directory public/doc/notes/mach3_supinfo. You need to be enabled separately to sup those colllections. Just send mail to mach@cs.cmu.edu with your machine name and IP address to request to be enabled.

If you do not have the space on your machine, but have some other machine (non I386 even) running Mach, you may put the control file there and run the above procedure. Note: you will have to register this new machine at CMU in order to run **sup**.

You can obtain Mach 2.5 I386 sources without using the Mach operating system. You MUST have a Mach 2.5 license and you MUST use the CMU **sup** program to request the sources, though. Everything you need to know about the **sup** program is available for anonymous FTP access on the machine mach.cs.cmu.edu (128.2.209.192), in the directory ***/usr/mach/public/sup***. There is a manual page, the program sources, library sources, and sample "collection" control files. There is also a README file that describes the contents further. If you do not have **sup** and are running 4.3BSD, you should have all the files necessary to build **sup**. If you have problems building **sup** under some other operating system, let us know; we might be able to help.

IV. Installation from an intermediary machine

The setup script that comes on the floppy disks assumes that the target i386 is connected to the net and doing the sups from CMU directly. It assumes that it is running as root, so that the userids and groupids of the files are preserved. This is necessary for a dozen or so set[ug]id files in /bin and /etc. Setup also executes three commands during the sup of the first collection.

If you want to sup to an intermediary machine you need to copy the sup commands from the setup script and execute them on the intermediary being careful to remove the e (execute) flag for the first collection.

```
sup -dov PORTFILE first
sup -dov PORTFILE bin.root
sup -dov PORTFILE bin.usr
```

PORTFILE looks as follows: You need to edit it to include your crypt and to change the base directory to the directory on the intermediary machine where you want the i386 binarires to be placed

```
first     host=x29.mach.cs.cmu.edu hostbase=/i386 base=/mnt/RFS/.LOCALROOT crypt=
bin.root  host=x29.mach.cs.cmu.edu hostbase=/i386 base=/mnt/RFS/.LOCALROOT crypt=
bin.usr   host=x29.mach.cs.cmu.edu hostbase=/i386 base=/mnt/RFS/.LOCALROOT crypt= prefix=usr
```

You should do the sup as root on the intermediary machine to preserve user and group ids.

Next you need to do the first two steps of the setup command on the target i386 machine and then copy the binaries from the intermediary machine to the hard-disk /RFS/.LOCALROOT.

Finally you need to execute the following three commands on the i386

```
execute cd dev; umask 0; tar xvf TARDEV
execute dd if=boot.hd of=/dev/rhd0c
  or
execute dd if=boot.sd of=/dev/rsd0c
execute ln mach ../..; ln mach vmunix
```

before you can boot off the hard disk.

V. KDB commands

Kdb is the kernel debugger for the Mach 2.5 kernel. The following is a brief list of the kdb commands. kdb uses the same command format as adb and expertise in adb is assumed. The Mach 3.0 kernel has a different debugger called ddb. A man page for it comes with the Mach 3.0 kernel sources.

```
To enter the debugger:

                ctrl-alt-d


        The format commands:
                 <addr>/<letters>
and write commands
                <addr>/{w,W} values
function exactly as you would expect from adb.  (And note that / and ?
are treated the same.)  So you can use kdb to disassemble code, look at
memory locations and change memory locations.

        $r      -       will print out the register values
        <rname/var      accesses the value of register_name/variable
        >rname/var      sets the value of register_name/variable

        The neat features are the "step/breakpoint" commands, which are
again analogous to the adb functions:
        :s      -       single step one instruction
        :c,C    -       continue until the next breakpoint
                        (these both takes counts ala adb syntax)
        :p,P    -       print the instructions while single stepping
        :j      -       step until the next call or return and count
                         the instructions
        :J      -       step until we return to this nesting level
                         (show functions entered and instruction
                          counts.)
        :S      -       step over the function call you are stopped at
        :r,R    -       set a temporary break at the return address, so
                         you effectively step out of this function. (Note:
                         make sure you have done the push %ebp, movl %esp,
                         %ebp before you try this.)
```

Breakpoints are set with <addr>:B and <addr>:D and listed with $B, just like adb. You may not provide commands to execute when a breakpoint is hit. <addr>:b #, <addr>:d and $b are very different breakpoints. They use the i386 hardware match facilities. You can set up to 4 breakpoints and you have to specify which one to ":b". Right now they work analogous to the "B" breakpoints. But someday, I will implement access to the hardware break on read and break on write features; all that is supported now is break on execute.

```
        $l      -       lists the state of all the threads
        $L      -       lists the state and stack of all the threads

        $k      -       does a stack trace
        $c      -       does a stack trace
<thread>$K      -       does a stack trace for the give thread
<addres>$C      -       does a stack trace at the given frame.
   <map>$m      -       prints the map at address <map>
<addres>$Pp     -       prints port
```

```
<addres>$PO     -       prints object
<addres>$PM     -       prints map
<addres>$PP     -       prints "page structure"

<function>! arg0 arg1 ...       invokes function with the given arguments

        R!      -       invokes the function to sync the disk and reboot
        Q!      -       invokes the function to reboot

        pb!     -       invokes the playback function.  The screen is small
                         compared to the info you might want to display; pb
                         will play back each line and wait for a space or
                         return before continuing.  Any other character
                         terminates playback.  The buffer is 64k.
```

VI. Manual Pages

Table of Contents