

kernel docs

Table of Contents

- [Introduction](#)
- [Warning - Don't Do It](#)
- [Reference Resources](#)
- [Configuring a Kernel - Sample](#)
- [Categories of Kernel Configurations](#)
 - [Platform or Machine](#)
 - [CPU Directives](#)
 - [Compatibility with Other *Nix.](#)
 - [Debug Directives](#)
 - [File Systems](#)
 - [File System Options](#)
 - [Networking](#)
 - [Operation Related Directives](#)
 - [SCSI Subsystem Options](#)
 - [PCVT Options \(i386 Architecture\) \(pcvt\(4\)\)](#)
 - [System V IPC Options](#)
 - [Devices \(Pseudo-Devices, Boot Config, Bus Controllers, SCSI, USB, Serial & Parallel, Mouse/Mice, IDE FD & HD, Networking, and Multimedia Devices \)](#)
 - [Miscellaneous](#)
- [LKM - Loadable Kernel Modules](#)
- [Author and Copyright](#)

Introduction

These notes are to try and compile kernel optimization options.

Why go to the trouble of documenting something that is already documented well ? I just needed to have a table listing of what options may be there to make it easier for me to remove those 'bits' from the kernel that I do not need in my configuration.

Would you like to build your own custom kernel ? Documentation on that is in the FAQ, config(8), and options(4) if that isn't helping you then hopefully the section at the end of these docs may help you further.

Some of this information is extracted from the FreeBSD documentation.

Warning

As they say, ignore these warnings at your own peril (especially since the warnings are from the developers themselves.)

From the FAQ 5.0 Kernel Configuration:

Under most circumstances you will NOT need to compile your own kernel. The GENERIC kernel will usually be all that you need. In fact, there are several reasons why you do not want to create your own kernel. The main reason is that it is very easy to make changes to the kernel configuration which look logical, but do not work. This is your danger sign. If something does not appear to work properly, please try the GENERIC kernel before sending in a bug report.

5.2 - Kernel Configuration Options

...

Not all kernel options have been tested for compatibility with all other options. Don't put an option in your kernel unless you actually have a reason to do so! The one kernel configuration which gets the most testing is the GENERIC kernel. This is usually a combination of the options in **`/usr/src/sys/arch/<your arch>/conf/GENERIC`** and **`/usr/src/sys/conf/GENERIC`**.

Look closely at these files and you will notice a line like:

`include "../../conf/GENERIC"`

This means that it is referencing yet another configuration file. This file stores non arch-dependent options. So when creating your Kernel Config be sure to look through **`/sys/conf/GENERIC`** and see what you want. There ARE options in there that are NEEDED.

...

It is always helpful to be able to debug problems with the kernel. But many choose not to put these options in their kernel because these options add considerable size to the kernel. They are however extremely helpful in a case where a bug might be present. This will help the developers discover the source of your problems much quicker.

Thanks to Marc Espie for this additional Warning, Reference Information

it's highly recommended to use the default GENERIC kernel as a rule. We try very hard to make it work and be efficient everywhere. Several reasons:

- with a good GENERIC kernel, every user benefits.
- `boot -c + config -e` means most useful tweaks don't need any recompiling.
- problems are MUCH easier to reproduce if starting with a GENERIC kernel.
- GENERIC is much better tested than any other combination. Thousands of people bang on GENERIC...

Imagine you're doing a bug-report. If you are using the GENERIC kernel, it's much more likely that developers will take the time to look at your report. Speaking from experience, bugs in `custom kernels' can be very hard to track down... for instance, as a developer, you don't always have the right combination of hardware to have that kernel running. So you often need to tweak that custom kernel further... so that often, the resulting kernel no longer has the bug. What happens in such a case is that a busy developer will shelve the bug-report until he has time to track it further, and concentrate instead on something reproduceable...

Reference Resources.

[ref: afterboot(8), config(8), options(4), boot_config(8), OpenBSD FAQ: Kernel Configuration]

[ref: CD1:src.tar.gz]

The options for the kernel are pretty well documented in the FAQ and man pages. And the most current information are more likely to be found in those man pages, and source code than anywhere else.

Following Marc Espie's recommendation above, take a serious look at `boot_config(8)` if that will resolve your problems without the need to configure a custom kernel.

Configuring a Kernel - Sample

I needed to make some modifications to my kernel to optimize OpenBSD for using Samba to share files on the network. A simplified process to configuring a kernel is listed below. This step-by-step is covered well in the FAQ, including how to compile the kernel if you don't have space to extract the complete source onto your hard-disk.

After decompressing the OpenBSD source tree onto your hard-disk (`/usr/src`), do the following:

```
# cd /usr/src/sys/arch/$ARCH/conf
```

This moves us into the custom configurations for your hardware architecture. Doing a listing in this directory will give you a sample of machine configurations. On my i386 architecture an (`/usr/src/sys/arch/i386/conf`) `ls` will indicate

```
# ls -al
```

```
CVS          GANDALFLUCIFIER  OPRAH  RAMDISKB  SARUMAN
DISKLESS    GENERIC Makefile.i386  PETRA  RAMDISKC  WALDORF
ELBERETHHERMES  NETTAN   RAMDISK  RAMDISK_CD
```

Make a copy of the Generic kernel configuration. Remember that this is the most tested configuration, so let's not stray too far from 'working' configuration.

```
# cp GENERIC MYCUSTOMKERNEL
```

We can now edit the configuration file **MYCUSTOMKERNEL** to add the directive for increasing NMBCLUSTERS (something I need to do.)

Note that the name you give your file will also be the name given to the Kernel. Although it seems that traditionally kernel names are in ALL CAPITAL LETTERS it seems to work fine with a mix of letter cases.

Edit: `/usr/src/arch/$ARCH/conf/MYCUSTOMKERNEL` to include:

```
option "NMBCLUSTERS=8192"
```

We make the necessary changes to the GENERIC kernel into our proposed custom kernel configuration. After you have saved the above modifications we can now configure the kernel compilation, and make the kernel.

```
# config MYCUSTOMKERNEL
```

Don't forget to run "make depend"

```
# cd ../compile/MYCUSTOMKERNEL
```

```
# make depend && make
```

This uses the "config" program to validate your custom kernel configuration and create directories and files required to perform a compilation on the custom kernel. After the "config" (there should be no errors since we made a very minor modification) we run a make to create our new bsd kernel file "bsd."

To use the kernel copy the kernel to the root directory and make it available during boot time.

```
cp /bsd /bsd.generic
```

```
cp /usr/arch/$ARCH/compile/MYCUSTOMKERNEL/bsd /bsd.nmb
```

We first make a backup of the existing/generic kernel (the docs says that most use a sequence like bsd.1 bsd.2 and I'm only using named kernels above as a demonstration.)

Next I copy the customised kernel into the root partition so boot can find it. In our experimental case I do not make it the default kernel and I name it bsd.nmb to indicate what was changed as opposed to the working kernel bsd.generic.

When you restart your server and get to the boot prompt, we can specify the new kernel we wish to try.

```
Using Drive: 0 Partition: 3
```

```
reading boot....
```

```
probing: pc0 com0 com1 apm mem[639K 95M a20=on]
```

```
disk: fd0 hd0
```

>> OpenBSD/i386 BOOT 1.26

boot > **bsd.nmb** <-- type in the kernel you want to 'boot'

The boot messages will scroll through your screen and you get to the login prompt. After login in you should see a display of your new kernel.

Last login: (date) from (ip-address)

OpenBSD 2.8 (**MYCUSTOMKERNEL**) #1: (date-of-compilation)

Welcome to OpenBSD: The proactively secure Unix-like operating system.

If you find the new kernel behaves as expected, without causing other problems on your system, then you can copy the new kernel over the default boot kernel.

cp **/bsd.nmb** /bsd

Please take the time to read the FAQ and afterboot(8) man pages before you continue.

Categories of Kernel Configuration

The list below is by categories commonly found in documentation:

- Platform or Machine
- CPU Directives
- Compatibility with Other *Nix.
- Debug Directives
- File Systems
- File System Options
- Networking
- Operation Related Directives
- SCSI Subsystem Options
- PCVT Options (i386 Architecture) (pcvt(4))
- System V IPC Options
- Devices
 - Pseudo-Devices
 - Boot Config
 - Bus Controllers
 - SCSI devices
 - USB and Universal Communication Devices
 - Serial & Parallel Ports
 - Mouse/Mice
 - IDE FD & HD Interfaces
 - Networking Devices
 - Multimedia Audio Devices
- Miscellaneous

Platform or Machine Type

Directive	Purpose	Ref:
machine i386	At least one of these directives is required to specify the target system of the kernel.	/usr/src/sys/arch/\$AR
machine atari m68k		CH/conf/[sample-
machine amiga m68k		files]
...		

Not all available architecture directives are listed above, just an example of what to look for when specifying the configuration for your machine. On the i386 class configurations I've played with, the machine option is specified in the Kernel configuration file. On other systems such as the atari or amiga, the machine configuration is specified as part of the "included" files.

Example use on a Macintosh (mac68k/conf/GENERIC)

```
machine mac68k m68k
```

CPU Directives (i386 systems)

Option	Purpose	Ref:
At least one is required		
I386_CPU	Intel i386 Processor	FreeBSD
I486_CPU	Intel i486 Processor (AMD K5)	Handbook
I586_CPU	Intel Pentium Processor (AMD K6, K6-2)	and emails
I686_CPU	Intel Pentium Pro / Pentium II	from readers

GPL_MATH_EMULATE Software Math Co-processor emulation using GNU Library. Should not be necessary on CPUs with internal math co-processor (eg. 486DX and higher)

The i386 Generic Kernel includes all CPUs to support starting up on any of the above CPUs, for optimal code it would be presumably best to use the CPU directive that best fits your configuration.

Warning: Please review the June 2001 misc@openbsd.com mailing list archives which discusses some Kernel problems with omitting too many of the CPU directives. Remember, most of the testing has been with all directives ON, your mileage with taking some of the directives out may be very different.

Compatibility with Other *Nix.

Option	Purpose
COMPAT_SVR4	On those architectures that support it, this enables binary compatibility with AT&T System V.4 UNIX binaries built for the same architecture. This currently includes the sparc and i386. Possibly the most widely known operating system based on this binary architecture is Sun's Solaris 2.x. See compat_svr4(8).
COMPAT_BSDOS	On those architectures that support it, this enables binary compatibility with BSD/OS applications. This option is supported on the i386 architecture. See compat_bsdos(8). Requires option COMPAT_43 also be used for proper operation.

COMPAT_LINUX	On those architectures that support it, this enables binary compatibility with Linux ELF and a.out applications built for the same architecture. This option is supported on the i386 architecture. See <code>compat_linux(8)</code> .
COMPAT_SUNOS	On those architectures that support it, this enables binary compatibility with SunOS 4.x applications built for the same architecture. This option is supported on the sparc and most m68k platforms. See <code>compat_sunos(8)</code> .
COMPAT_ULTRIX	On those architectures that support it, this enables binary compatibility with Ultrix applications built for the same architecture. This option is available on the little-endian MIPS platforms like the pmax and arc. See <code>compat_ultrix(8)</code> .
COMPAT_FREEBSD	On those architectures that support it, this enables binary compatibility with FreeBSD applications built for the same architecture. This option is available on the i386 architecture. See <code>compat_freebsd(8)</code> .
COMPAT_HPUX	On those architectures that support it, this enables binary compatibility with HP/UX applications built for the same architecture. This option is available on some m68k architectures. See <code>compat_hpx(8)</code> .
COMPAT_IBCS2	On those architectures that support it, this enables binary compatibility with iBCS2 applications built for the same architecture. This option is available on the i386 architecture. See <code>compat_ibcs2(8)</code> .
COMPAT_OSF1	On those architectures that support it, this enables binary compatibility with Digital UNIX (formerly OSF/1) applications built for the same architecture. This option is available on the alpha architecture. See <code>compat_osf1(8)</code> .
COMPAT_NOMID	Enable compatibility with a.out executables that lack a machine ID. On the i386, this includes NetBSD 0.8's ZMAGIC format, 386BSD and BSDI's QMAGIC, NMAGIC, and OMAGIC a.out formats. On the hp300 and other m68k architectures this permits certain old 4.3BSD binaries to work, though its use is discouraged now.

COMPAT_43	Use of this option is discouraged. It enables compatibility with 4.3BSD. It adds an old syscall for lseek() as well as ioctls for TIOCGETP and TIOCSETP. The return values for the getpid(2), getgid(2), and getuid(2) system calls are modified as well, to return the parent's PID and UID as well as the current process's. It also enables the deprecated NTTYDISC terminal line discipline. It provides backwards compatibility with the "old" SIOC[GS]IF{ADDR,DSTADDR,BRDADDR,NETMASK} interface ioctls, including binary compatibility for code written before the introduction of the sa_len field in sockaddrs. It also enables support for some older pre BSD 4.4 socket calls.
COMPAT_09	NetBSD 0.9. On those architectures that support it, this enables binary compatibility with NetBSD 0.9 applications built for the same architecture. (needs verification)
COMPAT_10	NetBSD 1.0, On those architectures that support it, this enables binary compatibility with NetBSD 1.0 applications built for the same architecture. (needs verification)
COMPAT_11	NetBSD 1.1, On those architectures that support it, this enables binary compatibility with NetBSD 1.1 applications built for the same architecture. (needs verification)

Example Use on arch/sun3/conf/DISKLESS

```
#option    TCP_COMPAT_42  # TCP bug compatibility with 4.2BSD
option    COMPAT_SUNOS  # can run SunOS 4.1.1 executables
option    COMPAT_43    # and 4.3BSD and ...
option    COMPAT_10    # NetBSD 1.0
option    COMPAT_11    # NetBSD 1.1
option    COMPAT_12    # NetBSD 1.2
```

Debug

Option

Purpose

DDB	Compiles in a kernel debugger for diagnosing kernel problems. See ddb(4) for details. Note: not available on all architectures.
DDB_SAFE_CONSOLE	Allows a break into the kernel debugger during boot. Useful when debugging problems that can cause init(8) to fail.
KGDB	Compiles in a remote kernel debugger stub for diagnosing kernel problems using the "remote target" feature of gdb. See gdb(1) for details. Note: not available on all architectures.
makeoptions DEBUG="-g"	The -g flag causes bsd.gdb to be built in addition to bsd. bsd.gdb is useful for debugging kernel crash dumps with gdb. Note that gdb's -k flag is obsolete and should not be used. Instead, the kernel can be debugged by starting gdb with the kernel name as an argument (no core file) and then use the gdb command "target kcore COREFILE".
DEBUG	Turns on miscellaneous kernel debugging. Since options are turned into preprocessor defines (see above), option DEBUG is equivalent to doing a #define DEBUG throughout the kernel. Much of the kernel has #ifdef DEBUG conditional debugging code. Note that many parts of the kernel (typically device drivers) include their own #ifdef XXX_DEBUG conditionals instead. This option also turns on certain other options, notably option KMEMSTATS, which may decrease system performance.

DIAGNOSTIC	Adds code to the kernel that does internal consistency checks. This code will cause the kernel to panic if corruption of internal data structures is detected.
GPROF	Adds code to the kernel for kernel profiling with kgmon(8).
makeoptions PROF="-pg"	The -pg flag causes the kernel to be compiled with support for profiling. The option GPROF is required for the kernel compile to succeed.
KTRACE	Adds hooks for the system call tracing facility, which allows users to watch the system call invocation behavior of processes. See ktrace(1) for details.

Example Use:

```

mvme88k/conf/M197:
    option          DDB          # Kernel debugger
sun3/conf/SMD_TEST:
    option          PMAP_DEBUG
mvme88k/conf/XT:
    #option        DEBUG        # Add debugging statements
mvme88k/conf/XT:
    #option        SYSCALL_DEBUG # debug all syscalls.

```

As can be seen from the above examples, there are more DEBUG options available for the kernel configuration that may be dependent on the hardware-platform. For more information please browse through the same kernel configurations for your hardware-platform and potentially the #ifdef's in the source codes for your hardware-platform.

File Systems

These options specify which file systems the kernel will support. You must include at least support for the device you boot from. A custom kernel can be smaller if you do not include support for those File Systems you are certain you will not be using.

Option	Purpose
FFS	Berkeley Fast File System (FFS). Most machines need this if they are not running diskless.
EXT2FS	Second Extended File System (EXT2FS). This is the most commonly used file system on the Linux operating system, and is provided here for compatibility. Some specific features of EXT2FS like the "behavior on errors" are not implemented. This file system can't be used with uid_t or gid_t values greater than 65535. Also, the filesystem will not function correctly on architectures with differing byte-orders. That is, a big-endian machine will not be able to read an ext2fs filesystem created on an i386 or other little-endian machine. See mount_ext2fs(8) for details.
MFS	memory file system (MFS). This file system stores files in swappable memory, and produces notable performance improvements when it is used as the file store for /tmp or similar mount points. See mount_mfs(8) for details.
NFSCIENT	client side of the NFS (Network File System) remote file sharing protocol. Although the bulk of the code implementing NFS is kernel based, several user level daemons are needed for it to work. See mount_nfs(8) for details on NFS.
CD9660	ISO 9660 + Rock Ridge file system, which is the standard file system used on many CD-ROMs. It also supports Joliet extensions. See mount_cd9660(8) for details.
MSDOSFS	MS-DOS FAT file system. The kernel also implements the Windows 95 extensions which permit the use of longer, mixedcase file names. See mount_msdos(8) and fsck_msdos(8) for details.

FDESC	A file system which can be mounted on /dev/fd. This filesystem permits access to the per-process file descriptor space via special files in the file system. See mount_fd(8) for details. Note that this facility is redundant, and thus unneeded on most OpenBSD systems, since the fd(4) pseudodevice driver already provides identical functionality. On most systems, instances of fd(4) are mknoded under /dev/fd/ and on /dev/stdin, /dev/stdout, and /dev/stderr.
KERNFS	A special file system (normally mounted on /kern) in which files representing various kernel variables and parameters may be found. See mount_kernfs(8) for details.
NULLFS	A loopback file system. This permits portions of the file hierarchy to be re-mounted in other places. The code really exists to provide an example of a stackable file system layer. See mount_null(8) for details.
PORTAL	Experimental portal filesystem. This permits interesting tricks like opening TCP sockets by opening files in the file system. The portal file system is conventionally mounted on /p and is partially implemented by a special daemon. See mount_portal(8) for details.
PROCFS	A special file system (conventionally mounted on /proc) in which the process space becomes visible in the file system. Among other things, the memory spaces of processes running on the system are visible as files, and signals may be sent to processes by writing to ctl files in the procfs namespace. See mount_procfs(8) for details.
UMAPFS	A loopback file system in which user and group IDs may be remapped this can be useful when mounting alien file systems with different uids and gids than the local system (eg, remote NFS). See mount_umap(8) for details.
UNION	Union file system, which permits directories to be mounted on top of each other in such a way that both file systems remain visible this permits tricks like allowing writing (and the deleting of files) on a read-only file system like a CD-ROM by mounting a local writable file system on top of the read-only file system. This filesystem is still experimental and is known to be somewhat unstable. See mount_union(8) for details.

Example Use on /arch/sparc/conf/SUN4C:

```
option      FFS
option      NFSSERVER    # Sun NFS-compatible filesystem
option      NFSCLIENT   # Sun NFS-compatible filesystem
option      KERNFS      # kernel data-structure filesystem
option      FIFO        # POSIX fifo support (in all filesystems)
option      QUOTA       # fast filesystem with user and group quotas
option      MFS         # memory-based filesystem
option      LOFS        # Loop-back filesystem
option      FDESC       # user file descriptor filesystem
option      NULLFS      # null fs, required by umapfs
option      UMAPFS      # uid/gid remapping filesystem
option      PORTAL      # portal filesystem (still experimental)
option      PROCFS      # /proc
option      CD9660      # ISO 9660 + Rock Ridge file system
option      UNION       # union file system
arc/conf/ARCTIC:option FFS,QUOTA # fast filesystem with user and group quotas
```

MFS.

an example of an /etc/fstab specification for using MFS as a RAM disk for the temporary storage directory /tmp.

```
/dev/wds2b /tmp mfs auto,rw 0 0
```

File System Options

Option	Purpose
FFS_SOFTUPDATES	<p>Enables a scheme that uses partial ordering of buffer cache operations to allow metadata updates in FFS to happen asynchronously, increasing write performance significantly. Normally, the FFS filesystem writes metadata updates synchronously which exacts a performance penalty in favor of filesystem integrity. With soft updates, you gain the performance of asynchronous writes while retaining the safety of synchronous metadata updates.</p> <p>Soft updates must be enabled on a per-filesystem basis. To do this, boot into single user mode and run <code>tunefs -s enable special</code> on each character special device you want to enable soft updates on, then run <code>reboot -n</code>.</p> <p>Processors with a small kernel address space, such as the sun4 and sun4c, do not have enough kernel memory to support soft updates. Attempts to use this option with these CPUs will cause a kernel hang or panic after a short period of use as the kernel will quickly run out of memory. This is not related to the amount of physical memory present in the machine it is a limitation of the CPU architecture itself.</p>
BUFCACHEPERCENT=integer	Percentage of RAM to use as a file system buffer. It defaults to 5.
NFSSERVER	server side of the NFS (Network File System) remote file sharing protocol. Although the bulk of the code implementing NFS is kernel based, several user level daemons are needed for it to work. See <code>mountd(8)</code> and <code> nfsd(8)</code> for details.
QUOTA	Enables kernel support for file system quotas. See <code>quotaon(8)</code> , <code>edquota(8)</code> , <code>repquota(8)</code> , and <code>quota(1)</code> for details. Note that quotas only work on "ffs" file systems, although <code>rpc.rquotad(8)</code> permits them to be accessed over NFS.
FIFO	Adds support for AT&T System V UNIX style FIFOs (i.e., "named pipes"). This option is recommended in almost all cases as many programs use these.

NVNODE=integer This option sets the size of the cache used by the name-to-inode translation routines, (a.k.a. the namei() cache, though called by many other names in the kernel source). By default, this cache has NPROC (set as 20 + 16 * MAXUSERS) * (80 + NPROC / 8) entries. A reasonable way to derive a value of NVNODE, should a large number of namei cache misses be noticed with a tool such as systat(1), is to examine the system's current computed value with sysctl(1), (which calls this parameter "kern.maxvnodes") and to increase this value until either the namei cache hit rate improves or it is determined that the system does not benefit substantially from an increase in the size of the namei cache.

Above definition is in param.c and the kernel option allows us to modify this specification.

EXT2FS_SYSTEM_FLAGS This option changes the behavior of the APPEND and IMMUTABLE flags for a file on an EXT2FS filesystem. Without this option, the superuser or owner of the file can set and clear them. With this option, only the superuser can set them, and they can't be cleared if the securelevel is greater than 0. See also chflags(1).

Example Use:

```
i386/conf/PETRA: option BUFCACHEPERCENT=50 # how much for the buffer cache?
i386/conf/LUCIFER: option BUFCACHEPERCENT=20
```

Networking

Option

GATEWAY

Purpose

Enables IPFORWARDING and (on most ports) increases the size of NMBCLUSTERS. In general, GATEWAY is used to indicate that a system should act as a router, and IPFORWARDING is not invoked directly. (Note that GATEWAY has no impact on protocols other than IP, such as CLNP or XNS.)

IPFORWARDING	Enables IP routing behavior. With this option enabled, the machine will forward IP datagrams between its interfaces that are destined for other machines. Note that even without this option, the kernel will still forward some packets (such as source routed packets) removing GATEWAY and IPFORWARDING is insufficient to stop all routing through a bastion host on a firewall source routing is controlled independently. Note that IP forwarding may be turned on and off independently of the setting of the IPFORWARDING option through the use of the net.inet.ip.forwarding sysctl variable. If net.inet.ip.forwarding is 1, IP forwarding is on. See sysctl(8) and sysctl(3) for details.
MROUTING	Includes support for IP multicast routers. INET should be set along with this. Multicast routing is controlled by the mouted(8) daemon.
INET	Includes support for the TCP/IP protocol stack. This option is currently required. See inet(4) for details
INET6	Includes support for the IPv6 protocol stack. See inet6(4) for details. Unlike INET, INET6 enables multicast routing code as well. This option requires INET at this moment, but it should not.
NS	Include support for the Xerox XNS protocol stack. See ns(4) for details.
ISO,TPIP	Include support for the ubiquitous OSI protocol stack. See iso(4) for details.
EON	Include support for OSI tunneling over IP.
CCITT,LLC,HDLC	Include support for the X.25 protocol stack. The state of this code is currently unknown. It probably contains bugs.

IPX, IPXIP	Include support for Internetwork Packet Exchange protocol commonly in use by Novell NetWare.
NETATALK	Include kernel support for the AppleTalk family of protocols. This suite of supporting code is sometimes called netatalk support.
TCP_COMPAT_42	Use of this option is extremely discouraged, so it should not be enabled. If any other machines on the network require enabling this, it's recommended that they be disconnected from the network. TCP bug compatibility with 4.2BSD. In 4.2BSD, TCP sequence numbers were 32-bit signed values. Modern implementations of TCP use unsigned values. This option clamps the initial sequence number to start in the range 2^{31} rather than the full unsigned range of 2^{32} . Also, under 4.2BSD, keepalive packets must contain at least one byte or else the remote end will not respond.
TCP_SACK	Turns on selective acknowledgements. Additional information about segments already received can be transmitted back to the sender, thus indicating segments that have been lost and allowing for a swifter recovery. Both communication endpoints need to support SACK. The fallback behaviour is NewReno fast recovery phase, which allows one lost segment to be recovered per round trip time. When more than one segment has been dropped per window, the transmission can continue without waiting for a retransmission timeout.
TCP_FACK	Turns on forward acknowledgements allowing a more precise estimate of outstanding data during the fast recovery phase by using SACK information. This option can only be used together with TCP_SACK.

TCP_SIGNATURE	Turns on support for the TCP MD5 Signature option (RFC 2385). This is used by Internet backbone routers to provide per-packet authentication for the TCP packets used to communicate BGP routing information. You will also need a routing daemon that supports this option in order to actually use it.
IPFILTER	This option enables the IP filtering on the packet level using Darren Reed's ip-filter package.
IPFILTER_LOG	This option, in conjunction with option IPFILTER, enables logging of IP packets using ip-filter.
IPFILTER_DEFAULT_BLOCK	This option sets the default policy of ip-filter to block packets that exit the rule-set unmatched. Otherwise they are silently passed. See ipf(1) for details.
PPP_FILTER	This option turns on pcap(3) based filtering for ppp connections. This option is used by pppd(8) which needs to be compiled with PPP_FILTER defined (the current default).
PPP_BSDCOMP	Enables BSD compressor for PPP connections.
PPP_DEFLATE	For use in conjunction with PPP_BSDCOMP; provides an interface to zlib for PPP for deflate compression/decompression.
IPSEC	This option enables IP security protocol support. See ipsec(4) for more details.
ENCDEBUG	This option enables debugging information to be conditionally logged in case IPSEC encounters errors. The option IPSEC is required along with this option. Debug logging can be turned on/off through the use of the net.ipsec.encap.encdebug sysctl variable. If net.ipsec.encap.encdebug is 1, debug logging is on. See sysctl(8) and sysctl(3) for details.

KEY Enables PFKEYv2 (RFC 2367) support. While not IP specific, this option is usually used in conjunction with option IPSEC.

Operation Related Options

Option	Purpose
SWAPPAGER	Turns on paging. (To be specific, this enables the virtual memory module responsible for handling page faults for "anonymous" objects (i.e., BSS pages)). MANDATORY the system cannot actually run without this "option".
DEVPAGER	Support for mmap()ing of devices. (Specifically, this enables the virtual memory module responsible for handling page faults on mapped devices ("cdev" vnodes)). MANDATORY the system cannot actually run without this "option".
NMBCLUSTERS=value	Size of kernel mbuf cluster map, mb_map, in CLBYTES-sized logical pages. Default on most ports is 256 (512 with "option GATEWAY"). See /usr/include/machine/param.h for exact default information. Increase this value if "mb_map full" messages appear.
NKMEMCLUSTERS=value	Machine specific configurations are #defines in arch/[machine]/ Size of kernel malloc area in CLBYTES-sized logical pages. This area is covered by the kernel submap kmem_map. See /usr/include/machine/param.h for the default value, which is port specific. Increase this value if "out of space in kmem_map" panics happen.
NBUF=value	

BUFPAGES=value	These options set the number of pages available for the buffer cache. Their default value is a machine dependent value, often calculated as between 5% and 10% of total available RAM.
NTP	Modify the scheduler code to add hooks necessary for running an NTP daemon. xntpd(8) is available as part of the port collection.
APM_NOPRINT	This option is supported on the i386 architecture. When enabled kernel messages regarding the status of the automatic power management system (APM) are suppressed. APM status can still be obtained using apm(8) and/or apmd(8).

Example of using NMBCLUSTERS

option "NMBCLUSTERS=8192"

Note that the only examples I've seen use quotations, so I follow the trend although there doesn't seem to be any particular reason to use quotations

SCSI Subsystem Options

Option	Purpose
SCSITERSE	Tenser SCSI error messages. This omits the table for decoding ASC/ASCQ info, saving about 8 bytes or so.
SCSIDEBUG	Prints extra debugging info for the SCSI subsystem to the console.

PCVT Options (i386 Architecture)

[ref: pcvt(4)]

Option	Purpose
PCVT_NSCREENS = number	Defines the number of virtual screens. Default: 8
PCVT_SCREENSAVER	Enables the built-in screensaver feature. Default: on

- PCVT_PRETTYSCRNS If enabled, a blinking-star screensaver is used. If disabled, the screen is simply blanked (which might be useful for energysaving monitors). Default: off
- PCVT_CTRL_ALT_DEL If enabled, the key combination invokes a CPU reset. Default: off (To change this, check `sysctl.conf` , set value of `machdep.kbdreset` to 1)
- PCVT_USEKBDSEC Do NOT override a security lock for the keyboard. Default: on
- PCVT_24LINESDEF If enabled, the 25-line modi (VT emulation with 25 lines) defaults to 24 lines only to provide a better compatibility to the original DEV VT220 (TM). Thus it should be possible to use the terminal information for those terminals without further changes. Note that this is a startup option; it is possible to toggle between the 24and 25-lines' display by the `scon(1)` utility. Default: off
- PCVT_META_ESC If enabled, a sequence composed of , followed by the normal key code is emitted if a key is pressed with the key modifier. If disabled, then normal key code with the value 0x80 added is sent. Default: off

The `pcvt` driver provides a virtual screen system with several additional features not available in `pc(4)` standard console device driver. Besides the ability of handling multiple virtual screens, probably the most important is an emulation of a wide range of DEC VT-220 (TM) functionality. See Features for a detailed description.

Note: As of OpenBSD 2.6, `pcvt` does not do character mapping by default in favor of a traditional PC display where 16 colors are available and the standard IBM font is used. See the `-o` flag in `scon(1)` to toggle between this mode and the old mode.

Refer to the file `arch/i386/isa/pcvt/[README.FIRST && pcvt_hdr.h]` in the kernel source tree for detailed documentation.

System V IPC Options

Option	Purpose
SYSVMSG	Includes support for AT&T System V UNIX style message queues. See <code>msgctl(2)</code> , <code>msgget(2)</code> , <code>msgrcv(2)</code> , <code>msgsnd(2)</code> .
SYSVSEM	Includes support for AT&T System V UNIX style semaphores. See <code>semctl(2)</code> , <code>semget(2)</code> , <code>semop(2)</code> .
SYSVSHM	Includes support for AT&T System V UNIX style shared memory. See <code>shmat(2)</code> , <code>shmctl(2)</code> , <code>shmdt(2)</code> , <code>shmget(2)</code> .
SHMPAXPGS=value	Sets the maximum number of AT&T System V UNIX style shared memory pages that are available through the <code>shmget(2)</code> system call. Default value is 1024 on most ports. See <code>/usr/include/machine/vmparam.h</code> for the default.

Pseudo-Devices

[ref: The Complete FreeBSD Book]

Pseudo-device drivers are parts of the kernel that act like device drivers but do not correspond to any actual hardware in the machine.

pseudo-device	Purpose
loop	network loop-back
bpfilter	packet filter
sl	CSLIP
ppp	PPP
tun	network tunnelling over tty
enc	encryption device
strip	Starmode Radio IP Interface
pty	Pseudo-Terminals
tb	Tablet Line Discipline
vnd	Paging to Files
ccd	Concatenated Disk Devices
ksyms	Kernel Symbol Device

pctr
 mtrr memory range attribute control (686 range PCs,
 PPro + & AMD K6-2 + (?)
 sequencer
 raid

Boot Config

Directive	Purpose
option COMCONSOLE	(needs revision-probably in a man page somewhere) according to the FreeBSD documentation "Setting serial port flag 0x20 specifies that you would prefer to have the system console on a serial line. This replaces old COMCONSOLE option."
maxusers	Specify generalised number of users to provide services for. Check man pages for more details on this option.
config bsd root on	Specify boot devices ?

I think the "*config bsd root*" allows you to specify in the kernel where your boot, swap, and kernel dumps are to be placed (specify the drive and partition.) Otherwise you can use "*config bsd swap generic*" and the configuration is determined by the disk layout (disklabel.) Note in the examples below you can specify the root and swap on a remote file system.

Examples from arch/i386/conf:

DISKLESS:	config bsd root on nfs swap on nfs
ELBERETH:	config bsd root on wd0 swap on wd0 and sd0 and sd1 dumps on sd0
GANDALF:	config bsd root on sd0a swap on sd0b and sd1b and sd3b dumps on sd0b
GENERIC:	config bsd swap generic
HERMES:	config bsd root on sd0 swap on sd0 and wd0 dumps on sd0
LUCIFER:	config bsd swap generic
NETTAN:	config bsd swap generic
OPRAH:	config bsd swap generic
PETRA:	config bsd root on sd0a swap on ccd0b dumps on sd1a
RAMDISK:	config bsd root on rd0a swap on rd0b and wd0b and sd0b
RAMDISKB:	config bsd root on rd0a swap on rd0b and wd0b and sd0b
RAMDISKC:	config bsd root on rd0a swap on rd0b and wd0b and sd0b
RAMDISK_CD:	config bsd root on rd0a swap on rd0b and wd0b and sd0b
SARUMAN:	config bsd root on wd0a swap on wd0b dumps on wd0b
WALDORF:	config bsd swap generic

The specifications seems to be equivalent over most platforms, differences may be in the root or swap devices and in a few cases the config is "netbsd" instead ofbsd.

Bus Controllers

Bus Controllers

Format:

BUSn at bus [bus ?][dev ? function ?] [port port_addr iomem io_addr iosiz io_size]

mainbus()

bios() BIOS

apm() Advanced Power Management

isa() ISA Bus. This was the original BUS in the IBM PC and evolved until it was replaced by the PCI BUS. ON IBM PC Compatible equipment continue to use the ISA directove.

eisa() EISA Bus. If you have an EISA motherboard (an upgrade to the ISA motherboard that is not as proliferant as the PCI bus) then you should specify this.

pci PCI Bus. This enables auto-detection of PCI boards and allows pcib to bridge between PCI bus and ISA bus.

pchb PCI-Host bridges

ppb PCI-PCI bridges

pcib PCI-ISA bridges (do nothing)

pcic ISA PCMCIA controllers

pcmcia PCMCIA Support

Example use from arch/i386/conf/PETRA

mainbus() at root	This seems to be the common invocation of mainbus() on all platforms with sample kernel configurations.
bios() at mainbus()	I only see this on i386 configurations, and other platforms I've looked at their sample configurations indicate the use of cpu() at mainbus()
apm() at bios() flags 0x0000	flags 0x0000 to use protocol version 1.0 flags 0x0101 to force protocol version 1.1
	I only see this on i386 configurations, and other platforms I've looked at their sample configurations indicate the use of apm(). This may be related to a platform specific hardware feature, not currently available (or coded) onto other hardware-platforms.
isa() at mainbus()	
isa() at pcib?	
eisa() at mainbus()	
pci* at mainbus() bus ?	
pchb* at pci? dev ? function ?	PCI-Host bridges
ppb* at pci? dev ? function ?	PCI-PCI bridges
pci* at ppb? bus ?	
pci* at pchb? bus ?	
pcib* at pci? dev ? function ?	PCI-ISA bridges (do nothing)
	ISA PCMCIA controllers
pcic0 at isa? port 0x3e0 iomem 0xd0000 iosiz 0x4000	
pcic1 at isa? port 0x3e2 iomem 0xd4000 iosiz 0x4000	
	PCMCIA Support
pcmcia* at pcic? controller ? socket ?	

USB and Universal Communication Devices

USB Devices

Format:

DEVICEn at bus [port port_addr configuration ? interface ?]

uhci	Universal Host Controller (Intel)
ohci	Open Host Controller
usb	USB Support
uhid	USB Generic HID devices
uaudio	USB Audio
ugen	USB Generic driver
puc	PCI "universal" communication device
npx	math coprocessor
isadma	DMA on ISA bus
isapnp	PNP on ISA bus
pc	generic PC console device
vt	
pcppi0 at isa?	
sysbeep0 at pcppi?	

Example use from arch/i386/conf/PETRA

	PCI USB Controllers
uhci* at pci?	Universal Host Controller (Intel)
ohci* at pci?	Open Host Controller
	USB Bus Support
usb* at uhci?	
usb* at ohci?	
	USB Generic HID devices
uhid* at uhub? port ? configuration ? interface ?	
	USB Audio
uaudio* at uhub? port ? configuration ?	
	USB Generic driver
ugen* at uhub? port ? configuration ? interface ?	

puc* at pci? PCI "universal" communication device
 npx0 at isa? port 0xf0 irq 13 math coprocessor
 isadma0 at isa?
 isapnp0 at isa?
 pc0 at isa? port 0x60 irq 1 generic PC console device
 vt0 at isa? port 0x60 irq 1
 pcppi0 at isa?
 sysbeep0 at pcppi?

Serial & Parallel Ports

Format:

DEVICEn at bus port port_addr irq irq_no

com Standard PC serial ports
 ast AST 4-port serial card
 boca BOCA 8-port serial cards
 rtfps RT 4-port serial cards
 com* at pcmcia? function ? PCMCIA modems/serial ports
 cy Cyclom serial card

 lpt standard PC parallel ports

On my system, the serial ports are accessed as /dev/tty00 and /dev/tty01 so this means I can't see it in the below example. The tty devices are referenced in ...i386/conf/files.i386 and created in /dev/MAKEDEV

Example use from arch/i386/conf/PETRA

com0 at isa? port 0x3f8 irq 4 standard PC serial ports MS-DOS COM1
 com1 at isa? port 0x2f8 irq 3 MS-DOS COM2
 com2 at isa? port 0x3e8 irq 5 MS-DOS COM3
 com3 at isa? port 0x2e8 irq 9 (note that this can conflict with some video cards)
 ast0 at isa? port 0x1a0 irq 5 AST 4-port serial card
 #ast1 at isa? port 0x2a0 irq 6 2nd AST 4-port serial card
 com* at ast? slave ?

#boca0 at isa? port 0x100 irq 5 BOCA 8-port serial cards
#com* at boca? slave ?
#rtfps0 at isa? port 0x1230 irq 10 RT 4-port serial cards
#com* at rtfps? slave ?
com* at pcmcia? function ? PCMCIA modems/serial ports
com* at isapnp?
com* at puc? port ?
option for using CD1400 handshaking for incoming data requires a special cable that exchanges the RTS and DTR lines options
CY_HW_RTS
cy0 at isa? iomem 0xd4000 irq 12 ISA cyclom serial card
#cy* at pci? PCI cyclom serial card
lpt0 at isa? port 0x378 irq 7 standard PC parallel ports
lpt1 at isa? port 0x278
lpt2 at isa? port 0x3bc

Mouse/Mice

The following are device settings for mice

Format:

DEVICEn at bus [port port_addr] irq irq_no

Device

lm Logitech bus mouse. Use this device if you have a Logitech (and potentially an ATI Inport) bus mouse board.
mms Microsoft InPort mouse. Use this device if you have a Microsoft Bus Mouse card.
pms PS/2 auxiliary port mouse. Note that this is different from the psm0 device used by FreeBSD (but designating the same thing.)

My PS/2 port mouse is accessed as /dev/pms0 which synchs with how the kernel is configured below. Note that documentation for FreeBSD uses the same port as /dev/psm0 which previously caused problems for me trying to learn OpenBSD with FreeBSD documentation.

Example use from arch/i386/conf/PETRA

lms0 at isa? port 0x23c irq 5 Logitech bus mouse

lms1 at isa? port 0x238 irq 5

mms0 at isa? port 0x23c irq 5 Microsoft InPort mouse

mms1 at isa? port 0x238 irq 5 specifying port address and irq #

pms0 at pckbd? irq 12 PS/2 auxiliary port mouse

SCSI devices

SCSI devices are first configured as the controller, and then mapping the generic scsibus to the controller. After this, we map the scsi device to the scsibus.

Device

sd	SCSI disk drives
st	SCSI tape drives
cd	SCSI CD-ROM drives
ch	SCSI autochangers
ss	SCSI scanners
uk	unknown SCSI

Controllers

bt	BusLogic [57]4X SCSI controllers
aha	Adaptec 154[02] SCSI controllers
ahb	Adaptec 174[024] SCSI controllers
ahc	Adaptec 284x, 274x, aic7770, 2940 SCSI controllers
isp	Qlogic ISP [12]0x0 SCSI/FibreChannel
aic	Adaptec 152[02] SCSI controllers
ncr	NCR 538XX SCSI controllers
adv	AdvanSys 1200A/B and ULTRA SCSI
adw	AdvanSys ULTRA WIDE SCSI

sea	Seagate ST0[12] SCSI controllers
uha	UltraStor [123]4f SCSI controllers
wds	WD7000 and TMC-7000 controllers

Example use from arch/i386/conf/PETRA

bt0 at isa? port 0x330 irq ? drq ?	BusLogic [57]4X SCSI controllers
bt1 at isa? port 0x334 irq ? drq ?	BusLogic [57]4X SCSI controllers
bt2 at isa? port ? irq ?	
scsibus* at bt?	
aha0 at isa? port 0x330 irq ? drq ?	Adaptec 154[02] SCSI controllers
aha1 at isa? port 0x334 irq ? drq ?	Adaptec 154[02] SCSI controllers
aha* at isapnp?	
scsibus* at aha?	
ahb* at eisa? slot ?	Adaptec 174[024] SCSI controllers
scsibus* at ahb?	
ahc0 at isa? port ? irq ?	Adaptec 284x SCSI controllers
ahc* at eisa? slot ?	Adaptec 274x, aic7770 SCSI controllers
ahc* at pci? dev ? function ?	Adaptec 2940 SCSI controllers
scsibus* at ahc?	
isp* at pci? dev ? function ?	Qlogic ISP [12]0x0 SCSI/FibreChannel
scsibus* at isp?	
aic0 at isa? port 0x340 irq 11	Adaptec 152[02] SCSI controllers
aic* at pcmcia? function ?	PCMCIA based aic SCSI controllers
scsibus* at aic?	
ncr* at pci? dev ? function ?	NCR 538XX SCSI controllers
scsibus* at ncr?	

adv* at pci? dev ? function ? scsibus* at adv?	AdvanSys 1200A/B and ULTRA SCSI
adw* at pci? dev ? function ? scsibus* at adw?	AdvanSys ULTRA WIDE SCSI
sea0 at isa? iomem 0xc8000 irq 5 scsibus* at sea?	Seagate ST0[12] SCSI controllers
uha0 at isa? port 0x330 irq ? drq ? uha1 at isa? port 0x334 irq ? drq ? uha* at eisa? slot ? scsibus* at uha?	UltraStor [13]4f SCSI controllers UltraStor [13]4f SCSI controllers UltraStor 24f SCSI controllers
wds0 at isa? port 0x350 irq 15 drq 6 #wds1 at isa? port 0x358 irq 11 drq 5 scsibus* at wds?	WD7000 and TMC-7000 controllers
sd* at scsibus? target ? lun ? st* at scsibus? target ? lun ? cd* at scsibus? target ? lun ? ch* at scsibus? target ? lun ? ss* at scsibus? target ? lun ? uk* at scsibus? target ? lun ?	SCSI disk drives SCSI tape drives SCSI CD-ROM drives SCSI autochangers SCSI scanners unknown SCSI

IDE FD & HD Interfaces

The floppy and hdd interface

format:

DEVICEn at bus port port_addr irq irq_no drq drq_no

Device

fdc	standard PC floppy controllers
pciide	IDE controllers
wdc	

atapiscsi	ATAPI<>SCSI
wt	Archive and Wangtek QIC tape drives
mcd	Mitsumi CD-ROM drives
fd	Floppy drives are designated with fd0a, fd0b
wd	IDE hard drives are designated wd#letter

fdc and wdc are the controllers and you access the specific floppy with either fd0a or fd0b (floppy drive a and b.) You can access the IDE Hard-Disk with wd0a-b on the primary controller or wd1a-b on the secondary controller.

Example use from arch/i386/conf/PETRA

```

fdc0 at isa? port 0x3f0 irq 6 drq 2      standard PC floppy controllers
#fdc1 at isa? port 0x370 irq ? drq ?
fd* at fdc? drive ?

```

IDE controllers

```

pciide* at pci ? dev ? function ? flags
0x0000
wdc0 at isa? port 0x1f0 irq 14 flags
0x00
wdc1 at isa? port 0x170 irq 15 flags
0x00
wdc* at pcmcia? function ?
wdc* at isapnp?

```

IDE hard drives

```

wd* at wdc? channel ? drive ? flags
0x0000
wd* at pciide? channel ? drive ? flags
0x0000

```

ATAPI<>SCSI

```

atapiscsi* at wdc? channel ?
atapiscsi* at pciide? channel ?

```

scsibus* at atapiscsi?

#wt0 at isa? port 0x308 irq 5 drq 1 Archive and Wangtek QIC tape drives

#mcd0 at isa? port 0x300 irq 10 Mitsumi CD-ROM drives

Networking Devices

Supported Ethernet Devices on i386 architecture as extracted from the i386/conf/PETRA sample kernel configuration.

You should be able to comment out the network devices you are not using, or include one which is not in your kernel.

The card support, in my understanding, actually supports the chipset on that card. Where another card uses the same chipset it is likely the driver for the 1st card will work with the 2nd card.

New Network Card not detected ?

If you have a new network card that is not supported by your kernel, one option is to rebuild a kernel with support for all the network cards, find out which chipset it detects the new card as, and then build your custom kernel with the newly discovered card specification.

Of course using loadable kernel modules (lkm) is probably a better solution, but I haven't figured that one out, yet.

Another reason your network card may not be detected is potentially the port address, or the IO memory address, or the IRQ selected by the Kernel configuration. Check your board settings by looking at the board, or with configuration software that may have come with the board and then reconfigure either the board, or the kernel.

format:

deviceN at bus [port port_no] [iomem iomem_no irq irq_no] [drq drq_no]

Device

we	WD/SMC 80x3 ethernet
ec	3COM 3C503 ethernet
ne	NE[12]000 ethernet
eg	3COM 3C505/Etherlink+ ethernet
el	3COM 3C501 ethernet

ef 3COM 3C515 PnP ethernet
ie Intel (?) EtherExpress, StarLAN and 3C507
le (lowercase "elle") IsoLan, NE2100, and DEPCA
ex Intel EtherExpress PRO/10
ep 3COM 3C579 ethernet
fea DEC DEFEA FDDI
lmc Lan Media Corp SSI/T3/HSSI
le PCnet-PCI based ethernet
de DC21X4X-based ethernet
fxp Intel EtherExpress 10/100B ethernet
ne NE2000-compatible ethernet cards
ne PCMCIA based NE2000 ethernet
ep 3COM 3C59x ethernet, and PCMCIA based 3C5xx
ethernet
sm PCMCIA based sm ethernet
xe Xircom ethernet
wi WaveLAN IEEE 802.11
fpa DEC DEFPA FDDI
xl 3c9xx ethernet
rl RealTek 81[23]9 ethernet
tx SMC 83C170 EPIC ethernet
tl Compaq Thunderlan ethernet
vr VIA Rhine ethernet
dc 21143, "tulip" clone ethernet
wb Winbond W89C840F ethernet
ti Alteon Tigon 1Gb ethernet
al ADMtek AL981/AN985 ethernet
sf Adaptec AIC-6915 ethernet
skc SysKonnnect GENesis 984x

Example use from arch/i386/conf/PETRA

we0 at isa? port 0x280 iomem 0xd0000 irq 9	WD/SMC 80x3 ethernet
we1 at isa? port 0x300 iomem 0xcc000 irq 10	
we* at isapnp?	
ec0 at isa? port 0x250 iomem 0xd8000 irq 9	3C503 ethernet
ne0 at isa? port 0x240 irq 9	NE[12]000 ethernet
ne1 at isa? port 0x300 irq 10	NE[12]000 ethernet
ne2 at isa? port 0x280 irq 9	NE[12]000 ethernet
ne* at isapnp?	NE[12]000 PnP ethernet
eg0 at isa? port 0x310 irq 5	3C505/Etherlink+ ethernet
el0 at isa? port 0x300 irq 9	3C501 ethernet
ep0 at isa? port ? irq ?	3C509 ethernet
ep* at isapnp?	3C509 PnP ethernet
ep* at isa? port ? irq ?	3C509 ethernet
ef* at isapnp?	3C515 PnP ethernet
ie0 at isa? port 0x360 iomem 0xd0000 irq 7	StarLAN and 3C507
ie1 at isa? port 0x300 irq 10	EtherExpress
le0 at isa? port 0x360 irq 15 drq 6	IsoLan, NE2100, and DEPCA
ex0 at isa? port 0x320 irq 5	Intel EtherExpress PRO/10
ep0 at eisa? slot ?	
ep* at eisa? slot ?	3C579 ethernet
fea* at eisa? slot ?	DEC DEFEA FDDI
lmc* at pci? dev ? function ?	Lan Media Corp SSI/T3/HSSI
le* at pci? dev ? function ?	PCnet-PCI based ethernet
le* at isapnp?	
de* at pci? dev ? function ?	DC21X4X-based ethernet
fxp* at pci? dev ? function ?	EtherExpress 10/100B ethernet

ne* at pci? dev ? function ?	NE2000-compatible ethernet
ep0 at pci? dev ? function ?	3C59x ethernet
ep* at pci? dev ? function ?	3C59x ethernet
ne* at pcmcia? function ?	PCMCIA based NE2000 ethernet
ep* at pcmcia? function ?	PCMCIA based 3C5xx ethernet
sm* at pcmcia? function ?	PCMCIA based sm ethernet
xe* at pcmcia? function ?	Xircom ethernet
wi* at pcmcia? function ?	WaveLAN IEEE 802.11
fpa* at pci? dev ? function ?	DEC DEFPA FDDI
xl* at pci? dev ? function ?	3c9xx ethernet
rl* at pci? dev ? function ?	RealTek 81[23]9 ethernet
tx* at pci? dev ? function ?	SMC 83C170 EPIC ethernet
tl* at pci? dev ? function ?	Compaq Thunderlan ethernet
vr* at pci? dev ? function ?	VIA Rhine ethernet
dc* at pci? dev ? function ?	21143, "tulip" clone ethernet
wb* at pci? dev ? function ?	Winbond W89C840F ethernet
ti* at pci? dev ? function ?	Alteon Tigon 1Gb ethernet
al* at pci? dev ? function ?	ADMtek AL981/AN985 ethernet
sf* at pci? dev ? function ?	Adaptec AIC-6915 ethernet
skc* at pci? dev ? function ?	SysKonnnect GENesis 984x
sk* at skc?	each port of above

Multimedia Audio Devices

	Media Independent Device (mii) drivers
exphy* at mii? phy ?	3Com internal PHYs
inphy* at mii? phy ?	Intel 82555 PHYs
iophy* at mii? phy ?	Intel 82553 PHYs
icsphy* at mii? phy ?	ICS 1890 PHYs
#lxtphy* at mii? phy ?	Level1 LXT970 PHYs
nsphy* at mii? phy ?	NS and compatible PHYs
#qsphy* at mii? phy ?	Quality Semi QS6612 PHYs
sqphy* at mii? phy ?	Seeq 8x220 PHYs
rlphy* at mii? phy ?	RealTek 8139 internal PHYs
#mtdphy* at mii? phy ?	Myson MTD972 PHYs
dcphy* at mii? phy ?	Digital Clone PHYs
xmphy* at mii? phy ?	XaQti XMAC-II PHYs
brgphy* at mii? phy ?	Broadcom Gigabit PHYs
ukphy* at mii? phy ?	"unknown" PHYs
pss0 at isa? port 0x220 irq 7 drq 6	Personal Sound System
sp0 at pss0 port 0x530 irq 10 drq 0	sound port driver
eap* at pci? dev ? function ?	Ensoniq AudioPCI S5016
eso* at pci? dev ? function ?	ESS Solo-1 PCI AudioDrive
sv* at pci? dev ? function ?	S3 SonicVibes (S3 617)
sb0 at isa? port 0x220 irq 5 drq 1	SoundBlaster
sb* at isapnp? ess* at isapnp?	ESS Tech ES188[78], ES888
wss0 at isa? port 0x530 irq 10 drq 0	Windows Sound System
wss* at isapnp? pas0 at isa? port 0x220 irq 7 drq 1	ProAudio Spectrum
gus0 at isa? port 0x220 irq 7 drq 1 drq2 6	Gravis UltraSound (drq2 is record drq)
ym* at isapnp?	

mpu* at isapnp?

OPL[23] FM syntheziers

#opl0 at isa? port 0x388

use only if not attached to sound card

opl* at eso?

opl* at sb?

opl* at ess?

MIDI support

midi* at pcppi?

MIDI interface to the PC speaker

midi* at sb?

SB MPU401 port

midi* at opl?

OPL FM synth

midi* at ym?

midi* at mpu?

The spkr driver provides a simple tone interface to the built in speaker.

#spkr0 at pcppi?

PC speaker

#Audio Support

audio* at sb?

audio* at gus?

audio* at pas?

audio* at sp?

audio* at ess?

audio* at wss?

audio* at ym?

audio* at eap?

audio* at eso?

audio* at sv?

#audio* at uaudio?

bktr0 at pci? dev ? function ?

BrookTree family video capture and TV tuner board?
(reference to FreeBSD
<http://www.freebsd.org/~fsmp/HomeAuto/Bt848.html>)

Joystick driver. Probe is a little strange; add only if you have one.

#joy0 at isa? port 0x201

joy* at isapnp?

#wdt0 at pci? dev ?
function ?

Ind Computer Source PCI-WDT50x driver

#aeon* at pci? dev ?
function ?

crypto support
Aeon crypto card

Miscellaneous

Option

Purpose

PCIVERBOSE

Makes the boot process more verbose for PCI peripherals (vendor names and other information is printed, etc.).

EISAVERBOSE

Makes the boot process more verbose for EISA peripherals.

PCMCIAVERBOSE

Makes the boot process more verbose for PCMCIA peripherals.

APERTURE

Provide in-kernel support for VGA framebuffer mapping by user-processes (such as an X windows server). This option is supported in the i386 architecture.

XSERVER

Support for X windows in the console driver.

LKM

Enable support for loadable kernel modules. See lkm(4) for details. Note: This option is not yet available on all architectures.

INSECURE

Hardwires the kernel security level at -1. This means that the system always runs in securelevel 0 mode, even when running multiuser. See init(8) for details on the implications of this. The kernel secure level may be manipulated by the superuser by altering the kern.securelevel sysctl variable. (It should be noted that the securelevel may only be lowered by a call from process ID 1, i.e., init(8).) See also sysctl(8) and sysctl(3).

MACHINE_NONCONTIG	This option changes part of the VM/pmap interface, to allow for non-contiguous memory. On some ports it is not an option. These ports typically only use one of the interfaces.
RAM_DISK_HOOKS	This option allows for some machine dependent functions to be called when the ramdisk driver is configured. This can result in automatically loading a ramdisk from floppy on open (among other things).
RAM_DISK_IS_ROOT	Forces the ramdisk to be the root device. This can only be overridden when the kernel is booted in the "ask-for-root" mode.
CCDNBUF=integer	The ccd(4) device driver uses "component buffers" to distribute I/O requests to the components of a concatenated disk. It keeps a freelist of buffer headers in order to reduce use of the kernel memory allocator. CCDNBUF is the number of buffer headers allocated on the freelist for each component buffer. It defaults to 8.
KMEMSTATS	The kernel memory allocator, malloc(9), will keep statistics on its performance if this option is enabled. Unfortunately, this option therefore essentially disables MALLOC() and FREE() forms of the memory allocator, which are used to enhance the performance of certain critical sections of code in the kernel. This option therefore can lead to a significant decrease in the performance of certain code in the kernel if enabled. Examples of such code namei() routine, the ccd(4) driver, the ncr(4) driver, and much of the networking code. Note that this option is silently turned on by the DEBUG option.
BOOT_CONFIG	Adds support for the -c boot option (User Kernel Config). Allows modification of kernel settings (i.e., device parameters) before booting the system.

UVM_SWAP_ENCRYPT Enables kernel support for encrypting pages that are written out to swap storage. Swap encryption prevents sensitive data from remaining on the disk even after the operating system has been shut down. This option should be turned on if cryptographic filesystems are used. The sysctl variable `vm.swapencrypt.enable` controls its behaviour. See `sysctl(8)` and `sysctl(3)` for details.

LKM - Loadable Kernel Modules

[ref: `modload(8)`, `lkm(4)`, `modstat(8)`, `modunload(8)`]

LKM is the answer to that error in your configuration file (or the overzealous, took to many things out, problem.) LKM as its name tells us, allows the administrator to dynamically load kernel modules without rebooting.

Most useful when your designed kernel does not include drivers for a new piece of hardware just attached to your machine. Test it by dynamically loading the kernel module before doing a recompile.

"`modload`" is used to install the module, an object file (.o file). "`modstat`" lets you find out what modules are currently loaded, and `modunload` lets you remove loadable modules the running kernel.