

Frequently Asked Questions for FreeBSD 12.X and 13.X

Abstract

This is the Frequently Asked Questions (FAQ) for FreeBSD versions 13.X and 12.X. Every effort has been made to make this FAQ as informative as possible; if you have any suggestions as to how it may be improved, send them to the [FreeBSD documentation project mailing list](#).

The latest version of this document is always available from the [FreeBSD website](#). It may also be downloaded as one large [HTML](#) file with HTTP or as a variety of other formats from the [FreeBSD FTP server](#).

Table of Contents

1. Introduction	3
2. Documentation and Support	8
3. Installation	11
4. Hardware Compatibility	14
5. Troubleshooting	18
6. User Applications	24
7. Kernel Configuration	26
8. Disks, File Systems, and Boot Loaders	28
9. ZFS	36
10. System Administration	38
11. The X Window System and Virtual Consoles	46
12. Networking	53
13. Security	58
14. Serial Communications	61
15. Miscellaneous Questions	63
16. The FreeBSD Funnies	67
17. Advanced Topics	71
18. Acknowledgments	75

Chapter 1. Introduction

1.1. What is FreeBSD?

FreeBSD is a modern operating system for desktops, laptops, servers, and embedded systems with support for a large number of [platforms](#).

It is based on U.C. Berkeley's "4.4BSD-Lite" release, with some "4.4BSD-Lite2" enhancements. It is also based indirectly on William Jolitz's port of U.C. Berkeley's "Net/2" to the i386™, known as "386BSD", though very little of the 386BSD code remains.

FreeBSD is used by companies, Internet Service Providers, researchers, computer professionals, students and home users all over the world in their work, education and recreation.

For more detailed information on FreeBSD, refer to the [FreeBSD Handbook](#).

1.2. What is the goal of the FreeBSD Project?

The goal of the FreeBSD Project is to provide a stable and fast general purpose operating system that may be used for any purpose without strings attached.

1.3. Does the FreeBSD license have any restrictions?

Yes. Those restrictions do not control how the code is used, but how to treat the FreeBSD Project itself. The license itself is available at [license](#) and can be summarized like this:

- Do not claim that you wrote this.
- Do not sue us if it breaks.
- Do not remove or modify the license.

Many of us have a significant investment in the project and would certainly not mind a little financial compensation now and then, but we definitely do not insist on it. We believe that our first and foremost "mission" is to provide code to any and all comers, and for whatever purpose, so that the code gets the widest possible use and provides the widest possible benefit. This, we believe, is one of the most fundamental goals of Free Software and one that we enthusiastically support.

Code in our source tree which falls under the [GNU General Public License \(GPL\)](#) or [GNU Library General Public License \(LGPL\)](#) comes with slightly more strings attached, though at least on the side of enforced access rather than the usual opposite. Due to the additional complexities that can evolve in the commercial use of GPL software, we do, however, endeavor to replace such software with submissions under the more relaxed [FreeBSD license](#) whenever possible.

1.4. Can FreeBSD replace my current operating system?

For most people, yes. But this question is not quite that cut-and-dried.

Most people do not actually use an operating system. They use applications. The applications are what really use the operating system. FreeBSD is designed to provide a robust and full-featured environment for applications. It supports a wide variety of web browsers, office suites, email readers, graphics programs, programming environments, network servers, and much more. Most of these applications can be managed through the [Ports Collection](#).

If an application is only available on one operating system, that operating system cannot just be replaced. Chances are, there is a very similar application on FreeBSD, however. As a solid office or Internet server or a reliable workstation, FreeBSD will almost certainly do everything you need. Many computer users across the world, including both novices and experienced UNIX® administrators, use FreeBSD as their only desktop operating system.

Users migrating to FreeBSD from another UNIX®-like environment will find FreeBSD to be similar. Windows® and Mac OS® users may be interested in instead using [GhostBSD](#), [MidnightBSD](#) or [NomadBSD](#) three FreeBSD-based desktop distributions. Non-UNIX® users should expect to invest some additional time learning the UNIX® way of doing things. This FAQ and the [FreeBSD Handbook](#) are excellent places to start.

1.5. Why is it called FreeBSD?

- It may be used free of charge, even by commercial users.
- Full source for the operating system is freely available, and the minimum possible restrictions have been placed upon its use, distribution and incorporation into other work (commercial or non-commercial).
- Anyone who has an improvement or bug fix is free to submit their code and have it added to the source tree (subject to one or two obvious provisions).

It is worth pointing out that the word "free" is being used in two ways here: one meaning "at no cost" and the other meaning "do whatever you like". Apart from one or two things you *cannot* do with the FreeBSD code, for example pretending you wrote it, you can really do whatever you like with it.

1.6. What are the differences between FreeBSD and NetBSD, OpenBSD, and other open source BSD operating systems?

James Howard wrote a good explanation of the history and differences between the various projects, called [The BSD Family Tree](#) which goes a fair way to answering this question. Some of the information is out of date, but the history portion in particular remains accurate.

Most of the BSDs share patches and code, even today. All of the BSDs have common ancestry.

The design goals of FreeBSD are described in [What is the goal of the FreeBSD Project?](#), above. The design goals of the other most popular BSDs may be summarized as follows:

- OpenBSD aims for operating system security above all else. The OpenBSD team wrote [ssh\(1\)](#) and [pf\(4\)](#), which have both been ported to FreeBSD.
- NetBSD aims to be easily ported to other hardware platforms.
- DragonFly BSD is a fork of FreeBSD 4.8 that has since developed many interesting features of its own, including the HAMMER file system and support for user-mode "vkernels".

1.7. What is the latest version of FreeBSD?

At any point in the development of FreeBSD, there can be multiple parallel branches. 13.X releases are made from the 13-STABLE branch, and 12.X releases are made from the 12-STABLE branch.

Up until the release of 12.0, the 12.X series was the one known as *-STABLE*. However, as of 14.X, the 12.X branch will be designated for an "extended support" status and receive only fixes for major problems, such as security-related fixes.

Releases are made [every few months](#). While many people stay more up-to-date with the FreeBSD sources (see the questions on [FreeBSD-CURRENT](#) and [FreeBSD-STABLE](#)) than that, doing so is more of a commitment, as the sources are a moving target.

More information on FreeBSD releases can be found on the [Release Engineering page](#) and in [release\(7\)](#).

1.8. What is *FreeBSD-CURRENT*?

[FreeBSD-CURRENT](#) is the development version of the operating system, which will in due course become the new FreeBSD-STABLE branch. As such, it is really only of interest to developers working on the system and die-hard hobbyists. See the [relevant section](#) in the [Handbook](#) for details on running *-CURRENT*.

Users not familiar with FreeBSD should not use FreeBSD-CURRENT. This branch sometimes evolves quite quickly and due to mistakes can be un-buildable at times. People that use FreeBSD-CURRENT are expected to be able to analyze, debug, and report problems.

1.9. What is the FreeBSD-STABLE concept?

FreeBSD-STABLE is the development branch from which major releases are made. Changes go into this branch at a slower pace and with the general assumption that they have first been tested in FreeBSD-CURRENT. However, at any given time, the sources for FreeBSD-STABLE may or may not be suitable for general use, as it may uncover bugs and corner cases that were not yet found in FreeBSD-CURRENT. Users who do not have the resources to perform testing should instead run the most recent release of FreeBSD. *FreeBSD-CURRENT*, on the other hand, has been one unbroken line since 2.0 was released.

For more detailed information on branches see "[FreeBSD Release Engineering: Creating the Release](#)"

[Branch](#)", the status of the branches and the upcoming release schedule can be found on the [Release Engineering Information](#) page.

Version [12.3](#) is the latest release from the 12-STABLE branch; it was released on December 7, 2021. Version [13.1](#) is the latest release from the 13-STABLE branch; it was released on May 16, 2022.

1.10. When are FreeBSD releases made?

The Release Engineering Team <re@FreeBSD.org> releases a new major version of FreeBSD about every 18 months and a new minor version about every 8 months, on average. Release dates are announced well in advance, so that the people working on the system know when their projects need to be finished and tested. A testing period precedes each release, to ensure that the addition of new features does not compromise the stability of the release. Many users regard this caution as one of the best things about FreeBSD, even though waiting for all the latest goodies to reach *-STABLE* can be a little frustrating.

More information on the release engineering process (including a schedule of upcoming releases) can be found on the [release engineering](#) pages on the FreeBSD Web site.

For people who need or want a little more excitement, binary snapshots are made weekly as discussed above.

1.11. When are FreeBSD snapshots made?

FreeBSD [snapshot](#) releases are made based on the current state of the *-CURRENT* and *-STABLE* branches. The goals behind each snapshot release are:

- To test the latest version of the installation software.
- To give people who would like to run *-CURRENT* or *-STABLE* but who do not have the time or bandwidth to follow it on a day-to-day basis an easy way of bootstrapping it onto their systems.
- To preserve a fixed reference point for the code in question, just in case we break something really badly later. (Although Git normally prevents anything horrible like this happening.)
- To ensure that all new features and fixes in need of testing have the greatest possible number of potential testers.

No claims are made that any *-CURRENT* snapshot can be considered "production quality" for any purpose. If a stable and fully tested system is needed, stick to full releases.

Snapshot releases are directly available from [snapshot](#).

Official snapshots are generated on a regular basis for all actively developed branches.

1.12. Who is responsible for FreeBSD?

The key decisions concerning the FreeBSD project, such as the overall direction of the project and who is allowed to add code to the source tree, are made by a [core team](#) of 9 people. There is a much larger team of more than 350 [committers](#) who are authorized to make changes directly to the

FreeBSD source tree.

However, most non-trivial changes are discussed in advance in the [mailing lists](#), and there are no restrictions on who may take part in the discussion.

1.13. Where can I get FreeBSD?

Every supported release of FreeBSD is available from the [FreeBSD release locator page](#):

- For the latest 13-STABLE release, 13.2-RELEASE, follow the link for [the appropriate architecture and installation mode for 13.2-RELEASE](#).
- For the latest 12-STABLE release, 12.4-RELEASE, follow the link for [the appropriate architecture and installation mode for 12.4-RELEASE](#).
- [Snapshot](#) releases are made monthly for the [-CURRENT](#) and [-STABLE](#) branches, these being of service purely to bleeding-edge testers and developers.

Information about obtaining FreeBSD on CD, DVD, and other media can be found in [the Handbook](#).

1.14. How do I access the Problem Report database?

The Problem Report database of all user change requests may be queried by using our web-based PR [query](#) interface.

The [web-based problem report submission interface](#) can be used to submit problem reports through a web browser.

Before submitting a problem report, read [Writing FreeBSD Problem Reports](#), an article on how to write good problem reports.

Chapter 2. Documentation and Support

2.1. What good books are there about FreeBSD?

The project produces a wide range of documentation, available online from this link: <https://www.FreeBSD.org/docs/>.

2.2. Is the documentation available in other formats, such as PDF?

Yes. [The documentation is also available in several formats](#) on the FreeBSD download site.

Documentation directories are categorized according to:

- The document's name, such as `faq`, or `handbook`.
- The document's language, based on the locale names found under `/usr/share/locale` on a FreeBSD system, with the encodings removed as all documentation now uses UTF-8. The current languages are:

Name	Meaning
<code>en</code>	English
<code>bn-bd</code>	Bengali or Bangla (Bangladesh)
<code>da</code>	Danish
<code>de</code>	German
<code>el</code>	Greek
<code>es</code>	Spanish
<code>fr</code>	French
<code>hu</code>	Hungarian
<code>it</code>	Italian
<code>ja</code>	Japanese
<code>ko</code>	Korean
<code>mn</code>	Mongolian
<code>nl</code>	Dutch
<code>pl</code>	Polish
<code>pt-br</code>	Portuguese (Brazil)
<code>ru</code>	Russian
<code>tr</code>	Turkish
<code>zh-cn</code>	Simplified Chinese (China)

Name	Meaning
zh-tw	Traditional Chinese (Taiwan)



Some documents may not be available in all languages.

- The document's format. We produce the documentation in a number of different output formats. Each format has its own advantages and disadvantages. Some formats are better suited for online reading, while others are meant to be aesthetically pleasing when printed on paper. Having the documentation available in any of these formats ensures that our readers will be able to read the parts they are interested in, either on their monitor, or on paper after printing the documents. The currently available formats are:

Format	Meaning
html	Depending on the document: one large HTML file containing the entire document or a collection of small linked HTML files, both together with images, stylesheets, and JavaScript
pdf	Adobe's Portable Document Format

- The compression and packaging scheme.
 - a. Where the format is `html`, the files are bundled up using `tar(1)`. The resulting `.tar` is then compressed using `gzip(1)`.
 - b. The PDF format generates one file. For example, `explaining-bsd_en.pdf`, `faq_en.pdf`, and so on.

After choosing the format, download the files, uncompress them if necessary, then copy the appropriate documents into place.

For example, the HTML version of the FAQ can be found in `doc/en/books/faq/faq_en.tar.gz`. To download and uncompress that file, type:

```
% fetch https://download.freebsd.org/doc/en/books/faq/faq_en.tar.gz
% tar xvf faq_en.tar.gz
```

If the file is compressed, `tar` will automatically detect the appropriate format and decompress it correctly, resulting in a collection of files. The main one is called `index.html`, which will contain the whole document, starting with the table of contents linking to the other parts of the document.

2.3. Where do I find info on the FreeBSD mailing lists? What FreeBSD news groups are available?

Refer to the [Handbook entry on mailing-lists](#) and the [Handbook entry on newsgroups](#).

2.4. Are there FreeBSD IRC (Internet Relay Chat) channels?

Yes, most major IRC networks host a FreeBSD chat channel and the FreeBSD wiki holds an up to date [list of IRC channels](#).

Each of these channels are distinct and are not connected to each other. Since their chat styles differ, try each to find one suited to your chat style.

2.5. Are there any web based forums to discuss FreeBSD?

The official FreeBSD forums are located at <https://forums.FreeBSD.org/>.

2.6. Where can I get commercial FreeBSD training and support?

[iXsystems, Inc.](#), parent company of the [FreeBSD Mall](#), provides commercial FreeBSD and TrueOS software [support](#), in addition to FreeBSD development and tuning solutions.

BSD Certification Group, Inc. provides system administration certifications for DragonFly BSD, FreeBSD, NetBSD, and OpenBSD. Refer to [their site](#) for more information.

Any other organizations providing training and support should contact the Project to be listed here.

Chapter 3. Installation

3.1. Which platform should I download? I have a 64 bit capable Intel® CPU, but I only see amd64.

amd64 is the term FreeBSD uses for 64-bit compatible x86 architectures (also known as "x86-64" or "x64"). Most modern computers should use amd64. Older hardware should use i386. When installing on a non-x86-compatible architecture, select the platform which best matches the hardware.

3.2. Which file do I download to get FreeBSD?

On the [Getting FreeBSD](#) page, select `[iso]` next to the architecture that matches the hardware.

Any of the following can be used:

file	description
disc1.iso	Contains enough to install FreeBSD and a minimal set of packages.
dvd1.iso	Similar to disc1.iso but with additional packages.
memstick.img	A bootable image sufficient for writing to a USB stick.
bootonly.iso	A minimal image that requires network access during installation to completely install FreeBSD.

Full instructions on this procedure and a little bit more about installation issues in general can be found in the [Handbook entry on installing FreeBSD](#).

3.3. What do I do if the install image does not boot?

This can be caused by not downloading the image in *binary* mode when using FTP.

Some FTP clients default their transfer mode to *ascii* and attempt to change any end-of-line characters received to match the conventions used by the client's system. This will almost invariably corrupt the boot image. Check the SHA-256 checksum of the downloaded boot image: if it is not *exactly* that on the server, then the download process is suspect.

When using a command line FTP client, type *binary* at the FTP command prompt after getting connected to the server and before starting the download of the image.

3.4. Where are the instructions for installing FreeBSD?

Installation instructions can be found at [Handbook entry on installing FreeBSD](#).

3.5. How can I make my own custom release or install disk?

Customized FreeBSD installation media can be created by building a custom release. Follow the instructions in the [Release Engineering](#) article.

3.6. Can Windows® co-exist with FreeBSD? (x86-specific)

If Windows® is installed first, then yes. FreeBSD's boot manager will then manage to boot Windows® and FreeBSD. If Windows® is installed afterwards, it will overwrite the boot manager. If that happens, see the next section.

3.7. Another operating system destroyed my Boot Manager. How do I get it back? (x86-specific)

This depends upon the boot manager. The FreeBSD boot selection menu can be reinstalled using [boot0cfg\(8\)](#). For example, to restore the boot menu onto the disk *ada0*:

```
# boot0cfg -B ada0
```

The non-interactive MBR bootloader can be installed using [gpart\(8\)](#):

```
# gpart bootcode -b /boot/mbr ada0
```

For more complex situations, including GPT disks, see [gpart\(8\)](#).

3.8. Do I need to install the source?

In general, no. There is nothing in the base system which requires the presence of the source to operate. Some ports, like [sysutils/lsof](#), will not build unless the source is installed. In particular, if the port builds a kernel module or directly operates on kernel structures, the source must be installed.

3.9. Do I need to build a kernel?

Usually not. The supplied **GENERIC** kernel contains the drivers an ordinary computer will need. [freebsd-update\(8\)](#), the FreeBSD binary upgrade tool, cannot upgrade custom kernels, another reason to stick with the **GENERIC** kernel when possible. For computers with very limited RAM, such as embedded systems, it may be worthwhile to build a smaller custom kernel containing just the required drivers.

3.10. Should I use DES, Blowfish, or MD5 passwords and how do I specify which form my users receive?

FreeBSD uses *SHA512* by default. DES passwords are still available for backwards compatibility with operating systems that still use the less secure password format. FreeBSD also supports the Blowfish and MD5 password formats. Which password format to use for new passwords is controlled by the `passwd_format` login capability in `/etc/login.conf`, which takes values of `des`, `blf` (if these are available) or `md5`. See the [login.conf\(5\)](#) manual page for more information about login capabilities.

3.11. What are the limits for FFS file systems?

For FFS file systems, the largest file system is practically limited by the amount of memory required to `fsck(8)` the file system. `fsck(8)` requires one bit per fragment, which with the default fragment size of 4 KB equates to 32 MB of memory per TB of disk. This does mean that on architectures which limit userland processes to 2 GB (e.g., i386™), the maximum `fsck(8)`'able filesystem is ~60 TB.

If there was not a `fsck(8)` memory limit the maximum filesystem size would be 2^{64} (blocks) * 32 KB \Rightarrow 16 Exa * 32 KB \Rightarrow 512 ZettaBytes.

The maximum size of a single FFS file is approximately 2 PB with the default block size of 32 KB. Each 32 KB block can point to 4096 blocks. With triple indirect blocks, the calculation is $32\text{ KB} * 12 + 32\text{ KB} * 4096 + 32\text{ KB} * 4096^2 + 32\text{ KB} * 4096^3$. Increasing the block size to 64 KB will increase the max file size by a factor of 16.

3.12. Why do I get an error message, readin failed after compiling and booting a new kernel?

The world and kernel are out of sync. This is not supported. Be sure to use `make buildworld` and `make buildkernel` to update the kernel.

Boot the system by specifying the kernel directly at the second stage, pressing any key when the | shows up before loader is started.

3.13. Is there a tool to perform post-installation configuration tasks?

Yes. `bsdconfig` provides a nice interface to configure FreeBSD post-installation.

Chapter 4. Hardware Compatibility

4.1. General

4.1.1. I want to get a piece of hardware for my FreeBSD system. Which model/brand/type is best?

This is discussed continually on the FreeBSD mailing lists but is to be expected since hardware changes so quickly. Read through the Hardware Notes for FreeBSD [12.3](#) or [13.1](#) and search the [mailing list archives](#) before asking about the latest and greatest hardware. Chances are a discussion about that type of hardware took place just last week.

Before purchasing a laptop, check the archives for [FreeBSD general questions mailing list](#), or possibly a specific mailing list for a particular hardware type.

4.1.2. What are the limits for memory?

FreeBSD as an operating system generally supports as much physical memory (RAM) as the platform it is running on does. Keep in mind that different platforms have different limits for memory; for example i386™ without PAE supports at most 4 GB of memory (and usually less than that because of PCI address space) and i386™ with PAE supports at most 64 GB memory. As of FreeBSD 10, AMD64 platforms support up to 4 TB of physical memory.

4.1.3. Why does FreeBSD report less than 4 GB memory when installed on an i386™ machine?

The total address space on i386™ machines is 32-bit, meaning that at most 4 GB of memory is addressable (can be accessed). Furthermore, some addresses in this range are reserved by hardware for different purposes, for example for using and controlling PCI devices, for accessing video memory, and so on. Therefore, the total amount of memory usable by the operating system for its kernel and applications is limited to significantly less than 4 GB. Usually, 3.2 GB to 3.7 GB is the maximum usable physical memory in this configuration.

To access more than 3.2 GB to 3.7 GB of installed memory (meaning up to 4 GB but also more than 4 GB), a special tweak called PAE must be used. PAE stands for Physical Address Extension and is a way for 32-bit x86 CPUs to address more than 4 GB of memory. It remaps the memory that would otherwise be overlaid by address reservations for hardware devices above the 4 GB range and uses it as additional physical memory (see [pae\(4\)](#)). Using PAE has some drawbacks; this mode of memory access is a little bit slower than the normal (without PAE) mode and loadable modules (see [kld\(4\)](#)) are not supported. This means all drivers must be compiled into the kernel.

The most common way to enable PAE is to build a new kernel with the special ready-provided kernel configuration file called PAE, which is already configured to build a safe kernel. Note that some entries in this kernel configuration file are too conservative and some drivers marked as unready to be used with PAE are actually usable. A rule of thumb is that if the driver is usable on 64-bit architectures (like AMD64), it is also usable with PAE. When creating a custom kernel configuration file, PAE can be enabled by adding the following line:

options

PAE

PAE is not much used nowadays because most new x86 hardware also supports running in 64-bit mode, known as AMD64 or Intel® 64. It has a much larger address space and does not need such tweaks. FreeBSD supports AMD64 and it is recommended that this version of FreeBSD be used instead of the i386™ version if 4 GB or more memory is required.

4.2. Architectures and Processors

4.2.1. Does FreeBSD support architectures other than the x86?

Yes. FreeBSD divides support into multiple tiers. Tier 1 architectures, such as i386 or amd64; are fully supported. Tiers 2 and 3 are supported on a best-effort basis. A full explanation of the tier system is available in the [Committer's Guide](#).

A complete list of supported architectures can be found on the [platforms page](#).

4.2.2. Does FreeBSD support Symmetric Multiprocessing (SMP)?

FreeBSD supports symmetric multi-processor (SMP) on all non-embedded platforms (e.g. i386, amd64, etc.). SMP is also supported in arm and MIPS kernels, although some CPUs may not support this. FreeBSD's SMP implementation uses fine-grained locking, and performance scales nearly linearly with number of CPUs.

[smp\(4\)](#) has more details.

4.2.3. What is microcode? How do I install Intel® CPU microcode updates?

Microcode is a method of programmatically implementing hardware level instructions. This allows for CPU bugs to be fixed without replacing the on board chip.

Install [sysutils/devcpu-data](#), then add:

```
microcode_update_enable="YES"
```

to `/etc/rc.conf`

4.3. Peripherals

4.3.1. What kind of peripherals does FreeBSD support?

See a list of hardware known to work and any applicable restrictions in the Hardware Notes for FreeBSD [12.3](#) or [13.1](#).

4.4. Keyboards and Mice

4.4.1. Is it possible to use a mouse outside the X Window system?

The default console driver, [vt\(4\)](#), provides the ability to use a mouse pointer in text consoles to cut & paste text. Run the mouse daemon, [moused\(8\)](#), and turn on the mouse pointer in the virtual console:

```
# moused -p /dev/xxxx -t yyyy
# vidcontrol -m on
```

Where `xxxx` is the mouse device name and `yyyy` is a protocol type for the mouse. The mouse daemon can automatically determine the protocol type of most mice, except old serial mice. Specify the `auto` protocol to invoke automatic detection. If automatic detection does not work, see the [moused\(8\)](#) manual page for a list of supported protocol types.

For a PS/2 mouse, add `moused_enable="YES"` to `/etc/rc.conf` to start the mouse daemon at boot time. Additionally, to use the mouse daemon on all virtual terminals instead of just the console, add `allscreens_flags="-m on"` to `/etc/rc.conf`.

When the mouse daemon is running, access to the mouse must be coordinated between the mouse daemon and other programs such as X Windows. Refer to the FAQ [Why does my mouse not work with X?](#) for more details on this issue.

4.4.2. How do I cut and paste text with a mouse in the text console?

It is not possible to remove data using the mouse. However, it is possible to copy and paste. Once the mouse daemon is running as described in the [previous question](#), hold down button 1 (left button) and move the mouse to select a region of text. Then, press button 2 (middle button) to paste it at the text cursor. Pressing button 3 (right button) will "extend" the selected region of text.

If the mouse does not have a middle button, it is possible to emulate one or remap buttons using mouse daemon options. See the [moused\(8\)](#) manual page for details.

4.4.3. My mouse has a fancy wheel and buttons. Can I use them in FreeBSD?

The answer is, unfortunately, "It depends". These mice with additional features require specialized driver in most cases. Unless the mouse device driver or the user program has specific support for the mouse, it will act just like a standard two, or three button mouse.

For the possible usage of wheels in the X Window environment, refer to [that section](#).

4.4.4. How do I use my delete key in sh and csh?

For the Bourne Shell, add the following lines to `~/.shrc`. See [sh\(1\)](#) and [editrc\(5\)](#).

```
bind ^[[3~ ed-delete-next-char # for xterm
```

For the C Shell, add the following lines to `~/.cshrc`. See [csh\(1\)](#).

```
bindkey ^[[3~ delete-char # for xterm
```

4.5. Other Hardware

4.5.1. Workarounds for no sound from my [pcm\(4\)](#) sound card?

Some sound cards set their output volume to 0 at every boot. On FreeBSD 13 and earlier, run the following command every time the machine boots:

```
# mixer pcm 100 vol 100 cd 100
```

Use the following command on FreeBSD 14 and later:

```
# mixer pcm.volume=100 vol.volume=100 cd.volume=100
```

4.5.2. Does FreeBSD support power management on my laptop?

FreeBSD supports the ACPI features found in modern hardware. Further information can be found in [acpi\(4\)](#).

Chapter 5. Troubleshooting

5.1. Why is FreeBSD finding the wrong amount of memory on i386™ hardware?

The most likely reason is the difference between physical memory addresses and virtual addresses.

The convention for most PC hardware is to use the memory area between 3.5 GB and 4 GB for a special purpose (usually for PCI). This address space is used to access PCI hardware. As a result real, physical memory cannot be accessed by that address space.

What happens to the memory that should appear in that location is hardware dependent. Unfortunately, some hardware does nothing and the ability to use that last 500 MB of RAM is entirely lost.

Luckily, most hardware remaps the memory to a higher location so that it can still be used. However, this can cause some confusion when watching the boot messages.

On a 32-bit version of FreeBSD, the memory appears lost, since it will be remapped above 4 GB, which a 32-bit kernel is unable to access. In this case, the solution is to build a PAE enabled kernel. See the entry on memory limits for more information.

On a 64-bit version of FreeBSD, or when running a PAE-enabled kernel, FreeBSD will correctly detect and remap the memory so it is usable. During boot, however, it may seem as if FreeBSD is detecting more memory than the system really has, due to the described remapping. This is normal and the available memory will be corrected as the boot process completes.

5.2. Why do my programs occasionally die with Signal 11 errors?

Signal 11 errors are caused when a process has attempted to access memory which the operating system has not granted it access to. If something like this is happening at seemingly random intervals, start investigating the cause.

These problems can usually be attributed to either:

1. If the problem is occurring only in a specific custom application, it is probably a bug in the code.
2. If it is a problem with part of the base FreeBSD system, it may also be buggy code, but more often than not these problems are found and fixed long before us general FAQ readers get to use these bits of code (that is what -CURRENT is for).

It is probably not a FreeBSD bug if the problem occurs compiling a program, but the activity that the compiler is carrying out changes each time.

For example, if `make buildworld` fails while trying to compile `ls.c` into `ls.o` and, when run again, it fails in the same place, this is a broken build. Try updating source and try again. If the compile fails elsewhere, it is almost certainly due to hardware.

In the first case, use a debugger such as [gdb\(1\)](#) to find the point in the program which is attempting to access a bogus address and fix it.

In the second case, verify which piece of hardware is at fault.

Common causes of this include:

1. The hard disks might be overheating: Check that the fans are still working, as the disk and other hardware might be overheating.
2. The processor running is overheating: This might be because the processor has been overclocked, or the fan on the processor might have died. In either case, ensure that the hardware is running at what it is specified to run at, at least while trying to solve this problem. If it is not, clock it back to the default settings.)

Regarding overclocking, it is far cheaper to have a slow system than a fried system that needs replacing! Also the community is not sympathetic to problems on overclocked systems.

3. Dodgy memory: if multiple memory SIMMS/DIMMS are installed, pull them all out and try running the machine with each SIMM or DIMM individually to narrow the problem down to either the problematic DIMM/SIMM or perhaps even a combination.
4. Over-optimistic motherboard settings: the BIOS settings, and some motherboard jumpers, provide options to set various timings. The defaults are often sufficient, but sometimes setting the wait states on RAM too low, or setting the "RAM Speed: Turbo" option will cause strange behavior. A possible idea is to set to BIOS defaults, after noting the current settings first.
5. Unclean or insufficient power to the motherboard. Remove any unused I/O boards, hard disks, or CD-ROMs, or disconnect the power cable from them, to see if the power supply can manage a smaller load. Or try another power supply, preferably one with a little more power. For instance, if the current power supply is rated at 250 Watts, try one rated at 300 Watts.

Read the section on [Signal 11](#) for a further explanation and a discussion on how memory testing software or hardware can still pass faulty memory. There is an extensive FAQ on this at [the SIG11 problem FAQ](#).

Finally, if none of this has helped, it is possibly a bug in FreeBSD. Follow [these instructions](#) to send a problem report.

5.3. My system crashes with either Fatal trap 12: page fault in kernel mode, or panic:, and spits out a bunch of information. What should I do?

The FreeBSD developers are interested in these errors, but need more information than just the error message. Copy the full crash message. Then consult the FAQ section on [kernel panics](#), build a debugging kernel, and get a backtrace. This might sound difficult, but does not require any programming skills. Just follow the instructions.

5.4. What is the meaning of the error `maxproc limit exceeded by uid %i`, please see `tuning(7)` and `login.conf(5)`?

The FreeBSD kernel will only allow a certain number of processes to exist at one time. The number is based on the `kern.maxusers` `sysctl(8)` variable. `kern.maxusers` also affects various other in-kernel limits, such as network buffers. If the machine is heavily loaded, increase `kern.maxusers`. This will increase these other system limits in addition to the maximum number of processes.

To adjust the `kern.maxusers` value, see the [File/Process Limits](#) section of the Handbook. While that section refers to open files, the same limits apply to processes.

If the machine is lightly loaded but running a very large number of processes, adjust the `kern.maxproc` tunable by defining it in `/boot/loader.conf`. The tunable will not get adjusted until the system is rebooted. For more information about tuning tunables, see `loader.conf(5)`. If these processes are being run by a single user, adjust `kern.maxprocperuid` to be one less than the new `kern.maxproc` value. It must be at least one less because one system program, `init(8)`, must always be running.

5.5. Why do full screen applications on remote machines misbehave?

The remote machine may be setting the terminal type to something other than `xterm` which is required by the FreeBSD console. Alternatively the kernel may have the wrong values for the width and height of the terminal.

Check the value of the `TERM` environment variable is `xterm`. If the remote machine does not support that try `vt100`.

Run `stty -a` to check what the kernel thinks the terminal dimensions are. If they are incorrect, they can be changed by running `stty rows RR cols CC`.

Alternatively, if the client machine has `x11/xterm` installed, then running `resize` will query the terminal for the correct dimensions and set them.

5.6. Why does it take so long to connect to my computer via ssh or telnet?

The symptom: there is a long delay between the time the TCP connection is established and the time when the client software asks for a password (or, in `telnet(1)`'s case, when a login prompt appears).

The problem: more likely than not, the delay is caused by the server software trying to resolve the client's IP address into a hostname. Many servers, including the Telnet and SSH servers that come with FreeBSD, do this to store the hostname in a log file for future reference by the administrator.

The remedy: if the problem occurs whenever connecting the client computer to any server, the

problem is with the client. If the problem only occurs when someone connects to the server computer, the problem is with the server.

If the problem is with the client, the only remedy is to fix the DNS so the server can resolve it. If this is on a local network, consider it a server problem and keep reading. If this is on the Internet, contact your ISP.

If the problem is with the server on a local network, configure the server to resolve address-to-hostname queries for the local address range. See [hosts\(5\)](#) and [named\(8\)](#) for more information. If this is on the Internet, the problem may be that the local server's resolver is not functioning correctly. To check, try to look up another host such as www.yahoo.com. If it does not work, that is the problem.

Following a fresh install of FreeBSD, it is also possible that domain and name server information is missing from `/etc/resolv.conf`. This will often cause a delay in SSH, as the option `UseDNS` is set to `yes` by default in `/etc/ssh/sshd_config`. If this is causing the problem, either fill in the missing information in `/etc/resolv.conf` or set `UseDNS` to `no` in `sshd_config` as a temporary workaround.

5.7. Why does `file: table is full` show up repeatedly in `dmesg(8)`?

This error message indicates that the number of available file descriptors have been exhausted on the system. Refer to the [kern.maxfiles](#) section of the [Tuning Kernel Limits](#) section of the Handbook for a discussion and solution.

5.8. Why does the clock on my computer keep incorrect time?

The computer has two or more clocks, and FreeBSD has chosen to use the wrong one.

Run `dmesg(8)`, and check for lines that contain `Timecounter`. The one with the highest quality value that FreeBSD chose.

```
# dmesg | grep Timecounter
Timecounter "i8254" frequency 1193182 Hz quality 0
Timecounter "ACPI-fast" frequency 3579545 Hz quality 1000
Timecounter "TSC" frequency 2998570050 Hz quality 800
Timecounters tick every 1.000 msec
```

Confirm this by checking the `kern.timecounter.hardware` [sysctl\(3\)](#).

```
# sysctl kern.timecounter.hardware
kern.timecounter.hardware: ACPI-fast
```

It may be a broken ACPI timer. The simplest solution is to disable the ACPI timer in

/boot/loader.conf:

```
debug.acpi.disabled="timer"
```

Or the BIOS may modify the TSC clock-perhaps to change the speed of the processor when running from batteries, or going into a power saving mode, but FreeBSD is unaware of these adjustments, and appears to gain or lose time.

In this example, the `i8254` clock is also available, and can be selected by writing its name to the `kern.timecounter.hardware sysctl(3)`.

```
# sysctl kern.timecounter.hardware=i8254
kern.timecounter.hardware: TSC -> i8254
```

The computer should now start keeping more accurate time.

To have this change automatically run at boot time, add the following line to `/etc/sysctl.conf`:

```
kern.timecounter.hardware=i8254
```

5.9. What does the error `swap_pager: indefinite wait buffer: mean?`

This means that a process is trying to page memory from disk, and the page attempt has hung trying to access the disk for more than 20 seconds. It might be caused by bad blocks on the disk drive, disk wiring, cables, or any other disk I/O-related hardware. If the drive itself is bad, disk errors will appear in `/var/log/messages` and in the output of `dmesg`. Otherwise, check the cables and connections.

5.10. What is a lock order reversal?

The FreeBSD kernel uses a number of resource locks to arbitrate contention for certain resources. When multiple kernel threads try to obtain multiple resource locks, there's always the potential for a deadlock, where two threads have each obtained one of the locks and blocks forever waiting for the other thread to release one of the other locks. This sort of locking problem can be avoided if all threads obtain the locks in the same order.

A run-time lock diagnostic system called `witness(4)`, enabled in FreeBSD-CURRENT and disabled by default for stable branches and releases, detects the potential for deadlocks due to locking errors, including errors caused by obtaining multiple resource locks with a different order from different parts of the kernel. The `witness(4)` framework tries to detect this problem as it happens, and reports it by printing a message to the system console about a `lock order reversal` (often referred to also as LOR).

It is possible to get false positives, as `witness(4)` is conservative. A true positive report *does not*

mean that a system is dead-locked; instead it should be understood as a warning that a deadlock could have happened here.



Problematic LORs tend to get fixed quickly, so check the [FreeBSD-CURRENT mailing list](#) before posting to it.

5.11. What does 'Called ... with the following non-sleepable locks held' mean?

This means that a function that may sleep was called while a mutex (or other unsleepable) lock was held.

The reason this is an error is because mutexes are not intended to be held for long periods of time; they are supposed to only be held to maintain short periods of synchronization. This programming contract allows device drivers to use mutexes to synchronize with the rest of the kernel during interrupts. Interrupts (under FreeBSD) may not sleep. Hence it is imperative that no subsystem in the kernel block for an extended period while holding a mutex.

To catch such errors, assertions may be added to the kernel that interact with the [witness\(4\)](#) subsystem to emit a warning or fatal error (depending on the system configuration) when a potentially blocking call is made while holding a mutex.

In summary, such warnings are non-fatal, however with unfortunate timing they could cause undesirable effects ranging from a minor blip in the system's responsiveness to a complete system lockup.

For additional information about locking in FreeBSD see [locking\(9\)](#).

5.12. Why does buildworld/installworld die with the message touch: not found?

This error does not mean that the [touch\(1\)](#) utility is missing. The error is instead probably due to the dates of the files being set sometime in the future. If the CMOS clock is set to local time, run `adjkerntz -i` to adjust the kernel clock when booting into single-user mode.

Chapter 6. User Applications

6.1. Where are all the user applications?

Refer to [the ports page](#) for info on software packages ported to FreeBSD.

Most ports should work on all supported versions of FreeBSD. Those that do not are specifically marked as such. Each time a FreeBSD release is made, a snapshot of the ports tree at the time of release is also included in the `ports/` directory.

FreeBSD supports compressed binary packages to easily install and uninstall ports. Use [pkg\(7\)](#) to control the installation of packages.

6.2. How do I download the Ports tree? Should I be using Git?

See [Installing the Ports Collection](#)

6.3. Why can I not build this port on my 12.X -, or 13.X -STABLE machine?

If the installed FreeBSD version lags significantly behind *-CURRENT* or *-STABLE*, update the Ports Collection using the instructions in [Using the Ports Collection](#). If the system is up-to-date, someone might have committed a change to the port which works for *-CURRENT* but which broke the port for *-STABLE*. [Submit](#) a bug report, since the Ports Collection is supposed to work for both the *-CURRENT* and *-STABLE* branches.

6.4. I just tried to build INDEX using make index, and it failed. Why?

First, make sure that the Ports Collection is up-to-date. Errors that affect building INDEX from an up-to-date copy of the Ports Collection are high-visibility and are thus almost always fixed immediately.

There are rare cases where INDEX will not build due to odd cases involving `OPTIONS_SET` being set in `make.conf`. If you suspect that this is the case, try to make INDEX with those variables turned off before reporting it to [FreeBSD ports mailing list](#).

6.5. I updated the sources, now how do I update my installed ports?

FreeBSD does not include a port upgrading tool, but it does have some tools to make the upgrade process somewhat easier. Additional tools are available to simplify port handling and are described in the [Upgrading Ports](#) section in the FreeBSD Handbook.

6.6. Do I need to recompile every port each time I perform a major version update?

Yes! While a recent system will run with software compiled under an older release, things will randomly crash and fail to work once other ports are installed or updated.

When the system is upgraded, various shared libraries, loadable modules, and other parts of the system will be replaced with newer versions. Applications linked against the older versions may fail to start or, in other cases, fail to function properly.

For more information, see [the section on upgrades](#) in the FreeBSD Handbook.

6.7. Do I need to recompile every port each time I perform a minor version update?

In general, no. FreeBSD developers do their utmost to guarantee binary compatibility across all releases with the same major version number. Any exceptions will be documented in the Release Notes, and advice given there should be followed.

6.8. Why is `/bin/sh` so minimal? Why does FreeBSD not use `bash` or another shell?

Many people need to write shell scripts which will be portable across many systems. That is why POSIX® specifies the shell and utility commands in great detail. Most scripts are written in Bourne shell (`sh(1)`), and because several important programming interfaces ([make\(1\)](#), [system\(3\)](#), [popen\(3\)](#), and analogues in higher-level scripting languages like Perl and Tcl) are specified to use the Bourne shell to interpret commands. As the Bourne shell is so often and widely used, it is important for it to be quick to start, be deterministic in its behavior, and have a small memory footprint.

The existing implementation is our best effort at meeting as many of these requirements simultaneously as we can. To keep `/bin/sh` small, we have not provided many of the convenience features that other shells have. That is why other more featureful shells like `bash`, `scsh`, `tcsh(1)`, and `zsh` are available. Compare the memory utilization of these shells by looking at the "VSZ" and "RSS" columns in a `ps -u` listing.

Chapter 7. Kernel Configuration

7.1. I would like to customize my kernel. Is it difficult?

Not at all! Check out the [kernel config section of the Handbook](#).



The new kernel will be installed to the `/boot/kernel` directory along with its modules, while the old kernel and its modules will be moved to the `/boot/kernel.old` directory. If a mistake is made in the configuration, simply boot the previous version of the kernel.

7.2. Why is my kernel so big?

GENERIC kernels shipped with FreeBSD are compiled in *debug mode*. Kernels built in debug mode contain debug data in separate files that are used for debugging. FreeBSD releases prior to 11.0 store these debug files in the same directory as the kernel itself, `/boot/kernel/`. In FreeBSD 11.0 and later the debug files are stored in `/usr/lib/debug/boot/kernel/`. Note that there will be little or no performance loss from running a debug kernel, and it is useful to keep one around in case of a system panic.

When running low on disk space, there are different options to reduce the size of `/boot/kernel/` and `/usr/lib/debug/`.

To not install the symbol files, make sure the following line exists in `/etc/src.conf`:

```
WITHOUT_KERNEL_SYMBOLS=yes
```

For more information see [src.conf\(5\)](#).

If you want to avoid building debug files altogether, make sure that both of the following are true:

- This line does not exist in the kernel configuration file:

```
makeoptions DEBUG=-g
```

- Do not run [config\(8\)](#) with `-g`.

Either of the above settings will cause the kernel to be built in debug mode.

To build and install only the specified modules, list them in `/etc/make.conf`:

```
MODULES_OVERRIDE= accf_http ipfw
```

Replace `accf_httpd ipfw` with a list of needed modules. Only the listed modules will be built. This reduces the size of the kernel directory and decreases the amount of time needed to build the

kernel. For more information, read `/usr/share/examples/etc/make.conf`.

Unneeded devices can be removed from the kernel to further reduce the size. See [I would like to customize my kernel. Is it difficult?](#) for more information.

To put any of these options into effect, follow the instructions to [build and install](#) the new kernel.

For reference, the FreeBSD 11 amd64 kernel (`/boot/kernel/kernel`) is approximately 25 MB.

7.3. Why does every kernel I try to build fail to compile, even GENERIC?

There are a number of possible causes for this problem:

- The source tree is different from the one used to build the currently running system. When attempting an upgrade, read `/usr/src/UPDATING`, paying particular attention to the "COMMON ITEMS" section at the end.
- The `make buildkernel` did not complete successfully. The `make buildkernel` target relies on files generated by the `make buildworld` target to complete its job correctly.
- Even when building `FreeBSD-STABLE`, it is possible that the source tree was fetched at a time when it was either being modified or it was broken. Only releases are guaranteed to be buildable, although `FreeBSD-STABLE` builds fine the majority of the time. Try re-fetching the source tree and see if the problem goes away. Try using a different mirror in case the previous one is having problems.

7.4. Which scheduler is in use on a running system?

The name of the scheduler currently being used is directly available as the value of the `kern.sched.name` sysctl:

```
% sysctl kern.sched.name
kern.sched.name: ULE
```

7.5. What is kern.sched.quantum?

`kern.sched.quantum` is the maximum number of ticks a process can run without being preempted in the 4BSD scheduler.

Chapter 8. Disks, File Systems, and Boot Loaders

8.1. How can I add my new hard disk to my FreeBSD system?

See the [Adding Disks](#) section in the FreeBSD Handbook.

8.2. How do I move my system over to my huge new disk?

The best way is to reinstall the operating system on the new disk, then move the user data over. This is highly recommended when tracking *-STABLE* for more than one release or when updating a release instead of installing a new one. Install `booteasy` on both disks with [boot0cfg\(8\)](#) and dual boot until you are happy with the new configuration. Skip the next paragraph to find out how to move the data after doing this.

Alternatively, partition and label the new disk with either [sade\(8\)](#) or [gpart\(8\)](#). If the disks are MBR-formatted, `booteasy` can be installed on both disks with [boot0cfg\(8\)](#) so that the computer can dual boot to the old or new system after the copying is done.

Once the new disk set up, the data cannot just be copied. Instead, use tools that understand device files and system flags, such as [dump\(8\)](#). Although it is recommended to move the data while in single-user mode, it is not required.

When the disks are formatted with UFS, never use anything but [dump\(8\)](#) and [restore\(8\)](#) to move the root file system. These commands should also be used when moving a single partition to another empty partition. The sequence of steps to use `dump` to move the data from one UFS partitions to a new partition is:

1. `newfs` the new partition.
2. `mount` it on a temporary mount point.
3. `cd` to that directory.
4. `dump` the old partition, piping output to the new one.

For example, to move `/dev/ada1s1a` with `/mnt` as the temporary mount point, type:

```
# newfs /dev/ada1s1a
# mount /dev/ada1s1a /mnt
# cd /mnt
# dump 0af - / | restore rf -
```

Rearranging partitions with `dump` takes a bit more work. To merge a partition like `/var` into its parent, create the new partition large enough for both, move the parent partition as described above, then move the child partition into the empty directory that the first move created:

```
# newfs /dev/ada1s1a
# mount /dev/ada1s1a /mnt
# cd /mnt
# dump 0af - / | restore rf -
# cd var
# dump 0af - /var | restore rf -
```

To split a directory from its parent, say putting `/var` on its own partition when it was not before, create both partitions, then mount the child partition on the appropriate directory in the temporary mount point, then move the old single partition:

```
# newfs /dev/ada1s1a
# newfs /dev/ada1s1d
# mount /dev/ada1s1a /mnt
# mkdir /mnt/var
# mount /dev/ada1s1d /mnt/var
# cd /mnt
# dump 0af - / | restore rf -
```

The `cpio(1)` and `pax(1)` utilities are also available for moving user data. These are known to lose file flag information, so use them with caution.

8.3. Which partitions can safely use Soft Updates? I have heard that Soft Updates on / can cause problems. What about Journaled Soft Updates?

Short answer: Soft Updates can usually be safely used on all partitions.

Long answer: Soft Updates has two characteristics that may be undesirable on certain partitions. First, a Soft Updates partition has a small chance of losing data during a system crash. The partition will not be corrupted as the data will simply be lost. Second, Soft Updates can cause temporary space shortages.

When using Soft Updates, the kernel can take up to thirty seconds to write changes to the physical disk. When a large file is deleted the file still resides on disk until the kernel actually performs the deletion. This can cause a very simple race condition. Suppose one large file is deleted and another large file is immediately created. The first large file is not yet actually removed from the physical disk, so the disk might not have enough room for the second large file. This will produce an error that the partition does not have enough space, even though a large chunk of space has just been released. A few seconds later, the file creation works as expected.

If a system should crash after the kernel accepts a chunk of data for writing to disk, but before that

data is actually written out, data could be lost. This risk is extremely small, but generally manageable.

These issues affect all partitions using Soft Updates. So, what does this mean for the root partition?

Vital information on the root partition changes very rarely. If the system crashed during the thirty-second window after such a change is made, it is possible that data could be lost. This risk is negligible for most applications, but be aware that it exists. If the system cannot tolerate this much risk, do not use Soft Updates on the root file system!

/ is traditionally one of the smallest partitions. If /tmp is on /, there may be intermittent space problems. Symlinking /tmp to /var/tmp will solve this problem.

Finally, [dump\(8\)](#) does not work in live mode (-L) on a filesystem, with Journaled Soft Updates (SU+J).

8.4. Can I mount other foreign file systems under FreeBSD?

FreeBSD supports a variety of other file systems.

UFS

UFS CD-ROMs can be mounted directly on FreeBSD. Mounting disk partitions from Digital UNIX and other systems that support UFS may be more complex, depending on the details of the disk partitioning for the operating system in question.

ext2/ext3

FreeBSD supports [ext2fs](#), [ext3fs](#), and [ext4fs](#) partitions. See [ext2fs\(5\)](#) for more information.

NTFS

FUSE based NTFS support is available as a port ([sysutils/fusefs-ntfs](#)). For more information, see [ntfs-3g\(8\)](#).

FAT

FreeBSD includes a read-write FAT driver. For more information, see [mount_msdosfs\(8\)](#).

ZFS

FreeBSD includes a port of Sun™'s ZFS driver. The current recommendation is to use it only on amd64 platforms with sufficient memory. For more information, see [zfs\(8\)](#).

FreeBSD includes the Network File System NFS and the FreeBSD Ports Collection provides several FUSE applications to support many other file systems.

8.5. How do I mount a secondary DOS partition?

The secondary DOS partitions are found after *all* the primary partitions. For example, if **E** is the second DOS partition on the second SCSI drive, there will be a device file for "slice 5" in /dev. To mount it:

```
# mount -t msdosfs /dev/da1s5 /dos/e
```

8.6. Is there a cryptographic file system for FreeBSD?

Yes, [gbde\(8\)](#) and [geli\(8\)](#). See the [Encrypting Disk Partitions](#) section of the FreeBSD Handbook.

8.7. How do I boot FreeBSD and Linux® using GRUB?

To boot FreeBSD using GRUB, add the following to either `/boot/grub/menu.lst` or `/boot/grub/grub.conf`, depending upon which is used by the Linux® distribution.

```
title FreeBSD 9.1
  root (hd0,a)
  kernel /boot/loader
```

Where `hd0,a` points to the root partition on the first disk. To specify the slice number, use something like this (`hd0,2,a`). By default, if the slice number is omitted, GRUB searches the first slice which has the `a` partition.

8.8. How do I boot FreeBSD and Linux® using BootEasy?

Install LILO at the start of the Linux® boot partition instead of in the Master Boot Record. Then boot LILO from BootEasy.

This is recommended when running Windows® and Linux® as it makes it simpler to get Linux® booting again if Windows® is reinstalled.

8.9. How do I change the boot prompt from ??? to something more meaningful?

This cannot be accomplished with the standard boot manager without rewriting it. There are a number of other boot managers in the `sysutils` category of the Ports Collection.

8.10. How do I use a new removable drive?

If the drive already has a file system on it, use a command like this:

```
# mount -t msdosfs /dev/da0s1 /mnt
```

If the drive will only be used with FreeBSD systems, partition it with UFS or ZFS. This will provide long filename support, improvement in performance, and stability. If the drive will be used by

other operating systems, a more portable choice, such as `msdosfs`, is better.

```
# dd if=/dev/zero of=/dev/da0 count=2
# gpart create -s GPT /dev/da0
# gpart add -t freebsd-ufs /dev/da0
```

Finally, create a new file system:

```
# newfs /dev/da0p1
```

and mount it:

```
# mount /dev/da0s1 /mnt
```

It is a good idea to add a line to `/etc/fstab` (see [fstab\(5\)](#)) so you can just type `mount /mnt` in the future:

```
/dev/da0p1/mnt ufs rw,noauto 0 0
```

8.11. Why do I get Incorrect super block when mounting a CD?

The type of device to mount must be specified. This is described in the Handbook section on [Using Data CDs](#).

8.12. Why do I get Device not configured when mounting a CD?

This generally means that there is no CD in the drive, or the drive is not visible on the bus. Refer to the [Using Data CDs](#) section of the Handbook for a detailed discussion of this issue.

8.13. Why do all non-English characters in filenames show up as ? on my CDs when mounted in FreeBSD?

The CD probably uses the "Joliet" extension for storing information about files and directories. This is discussed in the Handbook section on [Using Data CD-ROMs](#).

8.14. A CD burned under FreeBSD cannot be read under any other operating system. Why?

This means a raw file was burned to the CD, rather than creating an ISO 9660 file system. Take a look at the Handbook section on [Using Data CDs](#).

8.15. How can I create an image of a data CD?

This is discussed in the Handbook section on [Writing Data to an ISO File System](#). For more on working with CD-ROMs, see the [Creating CDs Section](#) in the Storage chapter in the Handbook.

8.16. Why can I not mount an audio CD?

Trying to mount an audio CD will produce an error like `cd9660: /dev/cd0: Invalid argument`. This is because `mount` only works on file systems. Audio CDs do not have file systems; they just have data. Instead, use a program that reads audio CDs, such as the [audio/xmcd](#) package or `port`.

8.17. How do I mount a multi-session CD?

By default, `mount(8)` will attempt to mount the last data track (session) of a CD. To load an earlier session, use the `-s` command line argument. Refer to `mount_cd9660(8)` for specific examples.

8.18. How do I let ordinary users mount CD-ROMs, DVDs, USB drives, and other removable media?

As `root` set the `sysctl` variable `vfs.usermount` to 1.

```
# sysctl vfs.usermount=1
```

To make this persist across reboots, add the line `vfs.usermount=1` to `/etc/sysctl.conf` so that it is reset at system boot time.

Users can only mount devices they have read permissions to. To allow users to mount a device permissions must be set in `/etc/devfs.conf`.

For example, to allow users to mount the first USB drive add:

```
# Allow all users to mount a USB drive.
    own      /dev/da0    root:operator
    perm     /dev/da0    0666
```

All users can now mount devices they could read onto a directory that they own:

```
% mkdir ~/my-mount-point
% mount -t msdosfs /dev/da0 ~/my-mount-point
```

Unmounting the device is simple:

```
% umount ~/my-mount-point
```

Enabling `vfs.usermount`, however, has negative security implications. A better way to access MS-DOS® formatted media is to use the [emulators/mtools](#) package in the Ports Collection.



The device name used in the previous examples must be changed according to the configuration.

8.19. The `du` and `df` commands show different amounts of disk space available. What is going on?

This is due to how these commands actually work. `du` goes through the directory tree, measures how large each file is, and presents the totals. `df` just asks the file system how much space it has left. They seem to be the same thing, but a file without a directory entry will affect `df` but not `du`.

When a program is using a file, and the file is deleted, the file is not really removed from the file system until the program stops using it. The file is immediately deleted from the directory listing, however. As an example, consider a file large enough to affect the output of `du` and `df`. A file being viewed with `more` can be deleted without causing an error. The entry is removed from the directory so no other program or user can access it. However, `du` shows that it is gone as it has walked the directory tree and the file is not listed. `df` shows that it is still there, as the file system knows that `more` is still using that space. Once the `more` session ends, `du` and `df` will agree.

This situation is common on web servers. Many people set up a FreeBSD web server and forget to rotate the log files. The access log fills up `/var`. The new administrator deletes the file, but the system still complains that the partition is full. Stopping and restarting the web server program would free the file, allowing the system to release the disk space. To prevent this from happening, set up [newsyslog\(8\)](#).

Note that Soft Updates can delay the freeing of disk space and it can take up to 30 seconds for the change to be visible.

8.20. How can I add more swap space?

This section [of the Handbook](#) describes how to do this.

8.21. Why does FreeBSD see my disk as smaller than the manufacturer says it is?

Disk manufacturers calculate gigabytes as a billion bytes each, whereas FreeBSD calculates them as 1,073,741,824 bytes each. This explains why, for example, FreeBSD's boot messages will report a disk that supposedly has 80 GB as holding 76,319 MB.

Also note that FreeBSD will (by default) [reserve](#) 8% of the disk space.

8.22. How is it possible for a partition to be more than 100% full?

A portion of each UFS partition (8%, by default) is reserved for use by the operating system and the `root` user. `df(1)` does not count that space when calculating the `Capacity` column, so it can exceed 100%. Notice that the `Blocks` column is always greater than the sum of the `Used` and `Avail` columns, usually by a factor of 8%.

For more details, look up `-m` in `tunefs(8)`.

Chapter 9. ZFS

9.1. What is the minimum amount of RAM one should have to run ZFS?

A minimum of 4GB of RAM is required for comfortable usage, but individual workloads can vary widely.

9.2. What is the ZIL and when does it get used?

The ZIL (ZFS intent log) is a write log used to implement posix write commitment semantics across crashes. Normally writes are bundled up into transaction groups and written to disk when filled ("Transaction Group Commit"). However syscalls like `fsync(2)` require a commitment that the data is written to stable storage before returning. The ZIL is needed for writes that have been acknowledged as written but which are not yet on disk as part of a transaction. The transaction groups are timestamped. In the event of a crash the last valid timestamp is found and missing data is merged in from the ZIL.

9.3. Do I need a SSD for ZIL?

By default, ZFS stores the ZIL in the pool with all the data. If an application has a heavy write load, storing the ZIL in a separate device that has very fast synchronous, sequential write performance can improve overall system performance. For other workloads, a SSD is unlikely to make much of an improvement.

9.4. What is the L2ARC?

The L2ARC is a read cache stored on a fast device such as an SSD. This cache is not persistent across reboots. Note that RAM is used as the first layer of cache and the L2ARC is only needed if there is insufficient RAM.

L2ARC needs space in the ARC to index it. So, perversely, a working set that fits perfectly in the ARC will not fit perfectly any more if a L2ARC is used because part of the ARC is holding the L2ARC index, pushing part of the working set into the L2ARC which is slower than RAM.

9.5. Is enabling deduplication advisable?

Generally speaking, no.

Deduplication takes up a significant amount of RAM and may slow down read and write disk access times. Unless one is storing data that is very heavily duplicated, such as virtual machine images or user backups, it is possible that deduplication will do more harm than good. Another consideration is the inability to revert deduplication status. If data is written when deduplication is enabled, disabling dedup will not cause those blocks which were deduplicated to be replicated until they are next modified.

Deduplication can also lead to some unexpected situations. In particular, deleting files may become much slower.

9.6. I cannot delete or create files on my ZFS pool. How can I fix this?

This could happen because the pool is 100% full. ZFS requires space on the disk to write transaction metadata. To restore the pool to a usable state, truncate the file to delete:

```
% truncate -s 0 unimportant-file
```

File truncation works because a new transaction is not started, new spare blocks are created instead.



On systems with additional ZFS dataset tuning, such as deduplication, the space may not be immediately available.

9.7. Is there TRIM support for ZFS on solid state drives?

ZFS in FreeBSD 12.3 and 12.4: TRIM is enabled by default. To disable TRIM: add the line below to `/etc/sysctl.conf`, then restart the system.

```
vfs.zfs.trim.enabled=0
```

OpenZFS in FreeBSD 13.0 and greater: see [zpool-trim\(8\)](#), and `autotrim` in [zpoolprops\(7\)](#).

Chapter 10. System Administration

10.1. Where are the system start-up configuration files?

The primary configuration file is `/etc/defaults/rc.conf` which is described in [rc.conf\(5\)](#). System startup scripts such as `/etc/rc` and `/etc/rc.d`, which are described in [rc\(8\)](#), include this file. *Do not edit this file!* Instead, to edit an entry in `/etc/defaults/rc.conf`, copy the line into `/etc/rc.conf` and change it there.

For example, to start [sshd\(8\)](#), the included OpenSSH daemon:

```
# echo 'sshd_enable="YES"' >> /etc/rc.conf
```

Alternatively, use [sysrc\(8\)](#) to modify `/etc/rc.conf`:

```
# sysrc sshd_enable="YES"
```

To start up local services, place shell scripts in the `/usr/local/etc/rc.d` directory. These shell scripts should be set executable, the default file mode is [555](#).

10.2. How do I add a user easily?

Use the [adduser\(8\)](#) command, or the [pw\(8\)](#) command for more complicated situations.

To remove the user, use the [rmuser\(8\)](#) command or, if necessary, [pw\(8\)](#).

10.3. Why do I keep getting messages like root: not found after editing `/etc/crontab`?

This is normally caused by editing the system crontab. This is not the correct way to do things as the system crontab has a different format to the per-user crontabs. The system crontab has an extra field, specifying which user to run the command as. [cron\(8\)](#) assumes this user is the first word of the command to execute. Since no such command exists, this error message is displayed.

To delete the extra, incorrect crontab:

```
# crontab -r
```

10.4. Why do I get the error, you are not in the correct group to su root when I try to su to root?

This is a security feature. In order to `su` to `root`, or any other account with superuser privileges, the user account must be a member of the `wheel` group. If this feature were not there, anybody with an account on a system who also found out `root`'s password would be able to gain superuser level access to the system.

To allow someone to `su` to `root`, put them in the `wheel` group using `pw`:

```
# pw groupmod wheel -m lisa
```

The above example will add user `lisa` to the group `wheel`.

10.5. I made a mistake in rc.conf, or another startup file, and now I cannot edit it because the file system is read-only. What should I do?

Restart the system using `boot -s` at the loader prompt to enter single-user mode. When prompted for a shell pathname, press `Enter` and run `mount -urw /` to re-mount the root file system in read/write mode. You may also need to run `mount -a -t ufs` to mount the file system where your favorite editor is defined. If that editor is on a network file system, either configure the network manually before mounting the network file systems, or use an editor which resides on a local file system, such as `ed(1)`.

In order to use a full screen editor such as `vi(1)` or `emacs(1)`, run `export TERM=xterm` so that these editors can load the correct data from the `termcap(5)` database.

After performing these steps, edit `/etc/rc.conf` to fix the syntax error. The error message displayed immediately after the kernel boot messages should indicate the number of the line in the file which is at fault.

10.6. Why am I having trouble setting up my printer?

See the [Handbook entry on printing](#) for troubleshooting tips.

10.7. How can I correct the keyboard mappings for my system?

Refer to the Handbook section on [using localization](#), specifically the section on [console setup](#).

10.8. Why can I not get user quotas to work properly?

1. It is possible that the kernel is not configured to use quotas. In this case, add the following line

to the kernel configuration file and recompile the kernel:

```
options QUOTA
```

Refer to the [Handbook entry on quotas](#) for full details.

2. Do not turn on quotas on /.
3. Put the quota file on the file system that the quotas are to be enforced on:

File System	Quota file
/usr	/usr/admin/quotas
/home	/home/admin/quotas
...	...

10.9. Does FreeBSD support System V IPC primitives?

Yes, FreeBSD supports System V-style IPC, including shared memory, messages and semaphores, in the GENERIC kernel. With a custom kernel, support may be loaded with the `sysvshm.ko`, `sysvsem.ko` and `sysvmsg.ko` kernel modules, or enabled in the custom kernel by adding the following lines to the kernel configuration file:

```
options    SYSVSHM      # enable shared memory
options    SYSVSEM      # enable for semaphores
options    SYSVMSG      # enable for messaging
```

Recompile and install the kernel.

10.10. What other mail-server software can I use instead of Sendmail?

The [Sendmail](#) server is the default mail-server software for FreeBSD, but it can be replaced with another MTA installed from the Ports Collection. Available ports include [mail/exim](#), [mail/postfix](#), and [mail/qmail](#). Search the mailing lists for discussions regarding the advantages and disadvantages of the available MTAs.

10.11. I have forgotten the root password! What do I do?

Do not panic! Restart the system, type `boot -s` at the `Boot:` prompt to enter single-user mode. At the question about the shell to use, hit `Enter` which will display a `#` prompt. Enter `mount -urw /` to remount the root file system read/write, then run `mount -a` to remount all the file systems. Run `passwd root` to change the `root` password then run `exit(1)` to continue booting.



If you are still prompted to give the `root` password when entering the single-user mode, it means that the console has been marked as `insecure` in `/etc/ttys`. In this case, it will be required to boot from a FreeBSD installation disk, choose the Live CD or Shell at the beginning of the install process and issue the commands mentioned above. Mount the specific partition in this case and then `chroot` to it. For example, replace `mount -urw /` with `mount /dev/ada0p1 /mnt; chroot /mnt` for a system on `ada0p1`.



If the root partition cannot be mounted from single-user mode, it is possible that the partitions are encrypted and it is impossible to mount them without the access keys. For more information see the section about encrypted disks in the [FreeBSD Handbook](#).

10.12. How do I keep `Control` + `Alt` + `Delete` from rebooting the system?

When using `vt(4)`, the default console driver, this can be done by setting the following `sysctl(8)`:

```
# sysctl kern.vt.kbd_reboot=0
```

10.13. How do I reformat DOS text files to UNIX[®] ones?

Use this `perl(1)` command:

```
% perl -i.bak -npe 's/\r\n/\n/g' file(s)
```

where *file(s)* is one or more files to process. The modification is done in-place, with the original file stored with a `.bak` extension.

Alternatively, use `tr(1)`:

```
% tr -d '\r' < dos-text-file > unix-file
```

dos-text-file is the file containing DOS text while *unix-file* will contain the converted output. This can be quite a bit faster than using `perl`.

Yet another way to reformat DOS text files is to use the [converters/dosunix](#) port from the Ports Collection. Consult its documentation about the details.

10.14. How do I re-read /etc/rc.conf and re-start /etc/rc without a reboot?

Go into single-user mode and then back to multi-user mode:

```
# shutdown now
# return
# exit
```

10.15. I tried to update my system to the latest -*STABLE*, but got -*BETA*_x, -*RC* or -*PRERELEASE*! What is going on?

Short answer: it is just a name. *RC* stands for "Release Candidate". It signifies that a release is imminent. In FreeBSD, *-PRERELEASE* is typically synonymous with the code freeze before a release. (For some releases, the *-BETA* label was used in the same way as *-PRERELEASE*.)

Long answer: FreeBSD derives its releases from one of two places. Major, dot-zero, releases, such as 9.0-RELEASE are branched from the head of the development stream, commonly referred to as *-CURRENT*. Minor releases, such as 6.3-RELEASE or 5.2-RELEASE, have been snapshots of the active *-STABLE* branch. Starting with 4.3-RELEASE, each release also now has its own branch which can be tracked by people requiring an extremely conservative rate of development (typically only security advisories).

When a release is about to be made, the branch from which it will be derived from has to undergo a certain process. Part of this process is a code freeze. When a code freeze is initiated, the name of the branch is changed to reflect that it is about to become a release. For example, if the branch used to be called 6.2-*STABLE*, its name will be changed to 6.3-*PRERELEASE* to signify the code freeze and signify that extra pre-release testing should be happening. Bug fixes can still be committed to be part of the release. When the source code is in shape for the release the name will be changed to 6.3-*RC* to signify that a release is about to be made from it. Once in the *RC* stage, only the most critical bugs found can be fixed. Once the release (6.3-*RELEASE* in this example) and release branch have been made, the branch will be renamed to 6.3-*STABLE*.

For more information on version numbers and the various Git branches, refer to the [Release Engineering](#) article.

10.16. I tried to install a new kernel, and the `chflags(1)` failed. How do I get around this?

Short answer: the security level is greater than 0. Reboot directly to single-user mode to install the kernel.

Long answer: FreeBSD disallows changing system flags at security levels greater than 0. To check the current security level:

```
# sysctl kern.securelevel
```

The security level cannot be lowered in multi-user mode, so boot to single-user mode to install the kernel, or change the security level in `/etc/rc.conf` then reboot. See the [init\(8\)](#) manual page for details on `securelevel`, and see `/etc/defaults/rc.conf` and the [rc.conf\(5\)](#) manual page for more information on `rc.conf`.

10.17. I cannot change the time on my system by more than one second! How do I get around this?

Short answer: the system is at a security level greater than 1. Reboot directly to single-user mode to change the date.

Long answer: FreeBSD disallows changing the time by more than one second at security levels greater than 1. To check the security level:

```
# sysctl kern.securelevel
```

The security level cannot be lowered in multi-user mode. Either boot to single-user mode to change the date or change the security level in `/etc/rc.conf` and reboot. See the [init\(8\)](#) manual page for details on `securelevel`, and see `/etc/defaults/rc.conf` and the [rc.conf\(5\)](#) manual page for more information on `rc.conf`.

10.18. Why is `rpc.statd` using 256 MB of memory?

No, there is no memory leak, and it is not using 256 MB of memory. For convenience, `rpc.statd` maps a large amount of memory into its address space. There is nothing terribly wrong with this from a technical standpoint; it just throws off things like [top\(1\)](#) and [ps\(1\)](#).

[rpc.statd\(8\)](#) maps its status file (`/var/db/statd.status`) into its address space; to save worrying about remapping the status file later when it needs to grow, it maps the status file with a generous size.

10.19. Why can I not unset the `schg` file flag?

The system is running at `securelevel` greater than 0. Lower the `securelevel` and try again. For more information, see [the FAQ entry on securelevel](#) and the [init\(8\)](#) manual page.

10.20. What is `vnlr`?

`vnlr` flushes and frees `vnodes` when the system hits the `kern.maxvnodes` limit. This kernel thread sits mostly idle, and only activates when there is a huge amount of RAM and users are accessing tens of thousands of tiny files.

10.21. What do the various memory states displayed by top mean?

- **Active**: pages recently statistically used.
- **Inactive**: pages recently statistically unused.
- **Laundry**: pages recently statistically unused but known to be dirty, that is, whose contents needs to be paged out before they can be reused.
- **Free**: pages without data content, which can be immediately reused.
- **Wired**: pages that are fixed into memory, usually for kernel purposes, but also sometimes for special use in processes.

Pages are most often written to disk (sort of a VM sync) when they are in the laundry state, but active or inactive pages can also be synced. This depends upon the CPU tracking of the modified bit being available, and in certain situations there can be an advantage for a block of VM pages to be synced, regardless of the queue they belong to. In most common cases, it is best to think of the laundry queue as a queue of relatively unused pages that might or might not be in the process of being written to disk. The inactive queue contains a mix of clean and dirty pages; clean pages near the head of the queue are reclaimed immediately to alleviate a free page shortage, and dirty pages are moved to the laundry queue for deferred processing.

There are some other flags (e.g., busy flag or busy count) that might modify some of the described rules.

10.22. How much free memory is available?

There are a couple of kinds of "free memory". The most common is the amount of memory immediately available without reclaiming memory already in use. That is the size of the free pages queue plus some other reserved pages. This amount is exported by the `vm.stats.vm.v_free_count` `sysctl(8)`, shown, for instance, by `top(1)`. Another kind of "free memory" is the total amount of virtual memory available to userland processes, which depends on the sum of swap space and usable memory. Other kinds of "free memory" descriptions are also possible, but it is relatively useless to define these, but rather it is important to make sure that the paging rate is kept low, and to avoid running out of swap space.

10.23. What is /var/empty?

`/var/empty` is a directory that the `sshd(8)` program uses when performing privilege separation. The `/var/empty` directory is empty, owned by `root` and has the `schg` flag set. This directory should not be deleted.

10.24. I just changed /etc/newsyslog.conf. How can I check if it does what I expect?

To see what `newsyslog(8)` will do, use the following:

```
% newsyslog -nrvv
```

10.25. My time is wrong, how can I change the timezone?

Use [tzsetup\(8\)](#).

Chapter 11. The X Window System and Virtual Consoles

11.1. What is the X Window System?

The X Window System (commonly **X11**) is the most widely available windowing system capable of running on UNIX® or UNIX® like systems, including FreeBSD. [The X.Org Foundation](#) administers the [X protocol standards](#), with the current reference implementation, version 11 release 7.7, so references are often shortened to **X11**.

Many implementations are available for different architectures and operating systems. An implementation of the server-side code is properly known as an **X server**.

11.2. I want to run Xorg, how do I go about it?

To install Xorg do one of the following:

Use the [x11/xorg](#) meta-port, which builds and installs every Xorg component.

Use [x11/xorg-minimal](#), which builds and installs only the necessary Xorg components.

Install Xorg from FreeBSD packages:

```
# pkg install xorg
```

After the installation of Xorg, follow the instructions from the [X11 Configuration](#) section of the FreeBSD Handbook.

11.3. I tried to run X, but I get a 'No devices detected' error when I type startx. What do I do now?

The system is probably running at a raised **securelevel**. It is not possible to start X at a raised **securelevel** because X requires write access to [io\(4\)](#). For more information, see at the [init\(8\)](#) manual page.

There are two solutions to the problem: set the **securelevel** back down to zero or run [xdm\(8\)](#) (or an alternative display manager) at boot time before the **securelevel** is raised.

See [How do I start XDM on boot?](#) for more information about running [xdm\(8\)](#) at boot time.

11.4. Why does my mouse not work with X?

When using [vt\(4\)](#), the default console driver, FreeBSD can be configured to support a mouse pointer on each virtual screen. To avoid conflicting with X, [vt\(4\)](#) supports a virtual device called `/dev/sysmouse`. All mouse events received from the real mouse device are written to the

[sysmouse\(4\)](#) device via [moused\(8\)](#). To use the mouse on one or more virtual consoles, *and* use X, see [Is it possible to use a mouse outside the X Window system?](#) and set up [moused\(8\)](#).

Then edit `/etc/X11/xorg.conf` and make sure the following lines exist:

```
Section "InputDevice"
    Option          "Protocol" "SysMouse"
    Option          "Device"  "/dev/sysmouse"
    .....
```

Starting with Xorg version 7.4, the `InputDevice` sections in `xorg.conf` are ignored in favor of autodetected devices. To restore the old behavior, add the following line to the `ServerLayout` or `ServerFlags` section:

```
Option "AutoAddDevices" "false"
```

Some people prefer to use `/dev/mouse` under X. To make this work, `/dev/mouse` should be linked to `/dev/sysmouse` (see [sysmouse\(4\)](#)) by adding the following line to `/etc/devfs.conf` (see [devfs.conf\(5\)](#)):

```
link    sysmouse    mouse
```

This link can be created by restarting [devfs\(5\)](#) with the following command (as `root`):

```
# service devfs restart
```

11.5. My mouse has a fancy wheel. Can I use it in X?

Yes, if X is configured for a 5 button mouse. To do this, add the lines `Buttons 5` and `ZAxisMapping 4 5` to the `"InputDevice"` section of `/etc/X11/xorg.conf`, as seen in this example:

```
Section "InputDevice"
    Identifier      "Mouse1"
    Driver          "mouse"
    Option          "Protocol" "auto"
    Option          "Device"  "/dev/sysmouse"
    Option          "Buttons"  "5"
    Option          "ZAxisMapping" "4 5"
EndSection
```

The mouse can be enabled in Emacs by adding these lines to `~/.emacs`:

```
;; wheel mouse
(global-set-key [mouse-4] 'scroll-down)
```



```
(global-set-key [mouse-5] 'scroll-up)
```

11.6. My laptop has a Synaptics touchpad. Can I use it in X?

Yes, after configuring a few things to make it work.

In order to use the Xorg synaptics driver, first remove `moused_enable` from `rc.conf`.

To enable synaptics, add the following line to `/boot/loader.conf`:

```
hw.psm.synaptics_support="1"
```

Add the following to `/etc/X11/xorg.conf`:

```
Section "InputDevice"
Identifier "Touchpad0"
Driver     "synaptics"
Option     "Protocol" "psm"
Option     "Device"  "/dev/psm0"
EndSection
```

Add the following into the "ServerLayout" section:

```
InputDevice    "Touchpad0" "SendCoreEvents"
```

11.7. How do I use remote X displays?

For security reasons, the default setting is to not allow a machine to remotely open a window.

To enable this feature, start X with the optional `-listen_tcp` argument:

```
% startx -listen_tcp
```

11.8. What is a virtual console and how do I make more?

Virtual consoles provide several simultaneous sessions on the same machine without doing anything complicated like setting up a network or running X.

When the system starts, it will display a login prompt on the monitor after displaying all the boot messages. Type in your login name and password to start working on the first virtual console.

To start another session, perhaps to look at documentation for a program or to read mail while waiting for an FTP transfer to finish, hold down `Alt` and press `F2`. This will display the login prompt for the second virtual console. To go back to the original session, press `Alt + F1`.

The default FreeBSD installation has eight virtual consoles enabled. `Alt + F1`, `Alt + F2`, `Alt + F3`, and so on will switch between these virtual consoles.

To enable more of virtual consoles, edit `/etc/ttys` (see [ttys\(5\)](#)) and add entries for `ttyv8` to `ttyvc`, after the comment on "Virtual terminals":

```
# Edit the existing entry for ttyv8 in /etc/ttys and change
# "off" to "on".
ttyv8  "/usr/libexec/getty Pc"      xterm  on secure
ttyv9  "/usr/libexec/getty Pc"      xterm  on secure
ttyva  "/usr/libexec/getty Pc"      xterm  on secure
ttyvb  "/usr/libexec/getty Pc"      xterm  on secure
```

The more virtual terminals, the more resources that are used. This can be problematic on systems with 8 MB RAM or less. Consider changing `secure` to `insecure`.



In order to run an X server, at least one virtual terminal must be left to `off` for it to use. This means that only eleven of the Alt-function keys can be used as virtual consoles so that one is left for the X server.

For example, to run X and eleven virtual consoles, the setting for virtual terminal 12 should be:

```
ttyvb  "/usr/libexec/getty Pc"      xterm  off secure
```

The easiest way to activate the virtual consoles is to reboot.

11.9. How do I access the virtual consoles from X?

Use `Ctrl + Alt + Fn` to switch back to a virtual console. Press `Ctrl + Alt + F1` to return to the first virtual console.

Once at a text console, use `Alt + Fn` to move between them.

To return to the X session, switch to the virtual console running X. If X was started from the command line using `startx`, the X session will attach to the next unused virtual console, not the text console from which it was invoked. For eight active virtual terminals, X will run on the ninth, so use `Alt + F9`.

11.10. How do I start XDM on boot?

There are two schools of thought on how to start `xdm(8)`. One school starts `xm` from `/etc/ttys` (see [ttys\(5\)](#)) using the supplied example, while the other sets `xm_enable=yes` in `/etc/rc.conf`. Both are equally valid, and one may work in situations where the other does not. In both cases the result is

the same: X will pop up a graphical login prompt.

The `ttys(5)` method has the advantage of documenting which vty X will start on and passing the responsibility of restarting the X server on logout to `init(8)`. The `rc(8)` method makes it easy to `kill xdm` if there is a problem starting the X server.

When using the `rc(8)` method, `xdm_tty` (default `tttyv8`) can be set in `/etc/rc.conf` to choose which vty `xdm(8)` opens on.

11.11. Why do I get Couldn't open console when I run xconsole?

When X is started with `startx`, the permissions on `/dev/console` will *not* get changed, resulting in things like `xterm -C` and `xconsole` not working.

This is because of the way console permissions are set by default. On a multi-user system, one does not necessarily want just any user to be able to write on the system console. For users who are logging directly onto a machine with a VTY, the `fbtab(5)` file exists to solve such problems.

In a nutshell, make sure an uncommented line of the form is in `/etc/fbtab` (see `fbtab(5)`):

```
/dev/ttyv0 0600 /dev/console
```

It will ensure that whomever logs in on `/dev/ttyv0` will own the console.

11.12. Why does my PS/2 mouse misbehave under X?

The mouse and the mouse driver may have become out of synchronization. In rare cases, the driver may also erroneously report synchronization errors:

```
psmintr: out of sync (xxxx != yyyy)
```

If this happens, disable the synchronization check code by setting the driver flags for the PS/2 mouse driver to `0x100`. This can be easiest achieved by adding `hint.psm.0.flags="0x100"` to `/boot/loader.conf` and rebooting.

11.13. How do I reverse the mouse buttons?

Type `xmodmap -e "pointer = 3 2 1"`. Add this command to `~/.xinitrc` or `~/.xsession` to make it happen automatically.

11.14. How do I install a splash screen and where do I find them?

The detailed answer for this question can be found in the [Boot Time Splash Screens](#) section of the FreeBSD Handbook.

11.15. Can I use the `Windows` keys on my keyboard in X?

Yes. Use `xmodmap(1)` to define which functions the keys should perform.

Assuming all Windows keyboards are standard, the keycodes for these three keys are the following:

- 115 - `Windows` key, between the left-hand `Ctrl` and `Alt` keys
- 116 - `Windows` key, to the right of `AltGr`
- 117 - `Menu`, to the left of the right-hand `Ctrl`

To have the left `Windows` key print a comma, try this.

```
# xmodmap -e "keycode 115 = comma"
```

To have the `Windows` key-mappings enabled automatically every time X is started, either put the `xmodmap` commands in `~/.xinitrc` or, preferably, create a `~/.xmodmaprc` and include the `xmodmap` options, one per line, then add the following line to `~/.xinitrc`:

```
xmodmap $HOME/.xmodmaprc
```

For example, to map the 3 keys to be `F13`, `F14`, and `F15`, respectively. This would make it easy to map them to useful functions within applications or the window manager.

To do this, put the following in `~/.xmodmaprc`.

```
keycode 115 = F13
keycode 116 = F14
keycode 117 = F15
```

For the `x11-wm/fvwm2` desktop manager, one could map the keys so that `F13` iconifies or de-iconifies the window the cursor is in, `F14` brings the window the cursor is in to the front or, if it is already at the front, pushes it to the back, and `F15` pops up the main Workplace menu even if the cursor is not on the desktop, which is useful when no part of the desktop is visible.

The following entries in `~/.fvwmrc` implement the aforementioned setup:

```
Key F13      FTIWS  A      Iconify
Key F14      FTIWS  A      RaiseLower
```

11.16. How can I get 3D hardware acceleration for OpenGL®?

The availability of 3D acceleration depends on the version of Xorg and the type of video chip. For an nVidia chip, use the binary drivers provided for FreeBSD by installing one of the following ports:

The latest versions of nVidia cards are supported by the [x11/nvidia-driver](#) port.

Older drivers are available as:

- [x11/nvidia-driver-390](#)
- [x11/nvidia-driver-340](#)
- [x11/nvidia-driver-304](#)

nVidia provides detailed information on which card is supported by which driver on their web site: http://www.nvidia.com/object/IO_32667.html.

For Matrox G200/G400, check the [x11-drivers/xf86-video-mga](#) port.

For ATI Rage 128 and Radeon see [ati\(4\)](#), [r128\(4\)](#) and [radeon\(4\)](#).

Chapter 12. Networking

12.1. Where can I get information on diskless booting?

"Diskless booting" means that the FreeBSD box is booted over a network, and reads the necessary files from a server instead of its hard disk. For full details, see [the Handbook entry on diskless booting](#).

12.2. Can a FreeBSD box be used as a dedicated network router?

Yes. Refer to the Handbook entry on [advanced networking](#), specifically the section on [routing and gateways](#).

12.3. Does FreeBSD support NAT or Masquerading?

Yes. For instructions on how to use NAT over a PPP connection, see the [Handbook entry on PPP](#). To use NAT over some other sort of network connection, look at the [natd](#) section of the Handbook.

12.4. How can I set up Ethernet aliases?

If the alias is on the same subnet as an address already configured on the interface, add `netmask 0xffffffff` to this command:

```
# ifconfig ed0 alias 192.0.2.2 netmask 0xffffffff
```

Otherwise, specify the network address and netmask as usual:

```
# ifconfig ed0 alias 172.16.141.5 netmask 0xfffff00
```

More information can be found in the FreeBSD [Handbook](#).

12.5. Why can I not NFS-mount from a Linux® box?

Some versions of the Linux® NFS code only accept mount requests from a privileged port; try to issue the following command:

```
# mount -o -P linuxbox:/blah /mnt
```

12.6. Why does mountd keep telling me it can't change attributes and that I have a bad exports list on my FreeBSD NFS server?

The most frequent problem is not understanding the correct format of `/etc/exports`. Review [exports\(5\)](#) and the [NFS](#) entry in the Handbook, especially the section on [configuring NFS](#).

12.7. How do I enable IP multicast support?

Install the [net/mrouted](#) package or port and add `mrouted_enable="YES"` to `/etc/rc.conf` start this service at boot time.

12.8. Why do I have to use the FQDN for hosts on my site?

See the answer in the FreeBSD [Handbook](#).

12.9. Why do I get an error, Permission denied, for all networking operations?

If the kernel is compiled with the `IPFIREWALL` option, be aware that the default policy is to deny all packets that are not explicitly allowed.

If the firewall is unintentionally misconfigured, restore network operability by typing the following as `root`:

```
# ipfw add 65534 allow all from any to any
```

Consider setting `firewall_type="open"` in `/etc/rc.conf`.

For further information on configuring this firewall, see the [Handbook chapter](#).

12.10. Why is my `ipfw` "fwd" rule to redirect a service to another machine not working?

Possibly because network address translation (NAT) is needed instead of just forwarding packets. A "fwd" rule only forwards packets, it does not actually change the data inside the packet. Consider this rule:

```
01000 fwd 10.0.0.1 from any to foo 21
```

When a packet with a destination address of `foo` arrives at the machine with this rule, the packet is

forwarded to *10.0.0.1*, but it still has the destination address of *foo*. The destination address of the packet is not changed to *10.0.0.1*. Most machines would probably drop a packet that they receive with a destination address that is not their own. Therefore, using a "fwd" rule does not often work the way the user expects. This behavior is a feature and not a bug.

See the [FAQ about redirecting services](#), the [natd\(8\)](#) manual, or one of the several port redirecting utilities in the [Ports Collection](#) for a correct way to do this.

12.11. How can I redirect service requests from one machine to another?

FTP and other service requests can be redirected with the [sysutils/socket](#) package or port. Replace the entry for the service in `/etc/inetd.conf` to call `socket`, as seen in this example for `ftpd`:

```
ftp stream tcp nowait nobody /usr/local/bin/socket socket ftp.example.com ftp
```

where *ftp.example.com* and *ftp* are the host and port to redirect to, respectively.

12.12. Where can I get a bandwidth management tool?

There are three bandwidth management tools available for FreeBSD. [dummynet\(4\)](#) is integrated into FreeBSD as part of [ipfw\(4\)](#). [ALTQ](#) has been integrated into FreeBSD as part of [pf\(4\)](#). Bandwidth Manager from [Emerging Technologies](#) is a commercial product.

12.13. Why do I get `/dev/bpf0: device not configured?`

The running application requires the Berkeley Packet Filter ([bpf\(4\)](#)), but it was removed from a custom kernel. Add this to the kernel config file and build a new kernel:

```
device bpf          # Berkeley Packet Filter
```

12.14. How do I mount a disk from a Windows® machine that is on my network, like `smbmount` in Linux®?

Use the SMBFS toolset. It includes a set of kernel modifications and a set of userland programs. The programs and information are available as [mount_smbfs\(8\)](#) in the base system.

12.15. What are these messages about: Limiting icmp/open port/closed port response in my log files?

This kernel message indicates that some activity is provoking it to send a large amount of ICMP or

TCP reset (RST) responses. ICMP responses are often generated as a result of attempted connections to unused UDP ports. TCP resets are generated as a result of attempted connections to unopened TCP ports. Among others, these are the kinds of activities which may cause these messages:

- Brute-force denial of service (DoS) attacks (as opposed to single-packet attacks which exploit a specific vulnerability).
- Port scans which attempt to connect to a large number of ports (as opposed to only trying a few well-known ports).

The first number in the message indicates how many packets the kernel would have sent if the limit was not in place, and the second indicates the limit. This limit is controlled using `net.inet.icmp.icmplim`. This example sets the limit to 300 packets per second:

```
# sysctl net.inet.icmp.icmplim=300
```

To disable these messages without disabling response limiting, use `net.inet.icmp.icmplim_output` to disable the output:

```
# sysctl net.inet.icmp.icmplim_output=0
```

Finally, to disable response limiting completely, set `net.inet.icmp.icmplim` to 0. Disabling response limiting is discouraged for the reasons listed above.

12.16. What are these arp: unknown hardware address format error messages?

This means that some device on the local Ethernet is using a MAC address in a format that FreeBSD does not recognize. This is probably caused by someone experimenting with an Ethernet card somewhere else on the network. This is most commonly seen on cable modem networks. It is harmless, and should not affect the performance of the FreeBSD system.

12.17. Why do I keep seeing messages like: 192.168.0.10 is on fxp1 but got reply from 00:15:17:67:cf:82 on rl0, and how do I disable it?

A packet is coming from outside the network unexpectedly. To disable them, set `net.link.ether.inet.log_arp_wrong_iface` to 0.

12.18. How do I compile an IPv6 only kernel?

Configure your kernel with these settings:

```
include GENERIC
```

```
ident GENERIC-IPV6ONLY
makeoptions MKMODULESENV+="WITHOUT_INET_SUPPORT="
nooptions INET
nodevice gre
```

Chapter 13. Security

13.1. What is a sandbox?

"Sandbox" is a security term. It can mean two things:

- A process which is placed inside a set of virtual walls that are designed to prevent someone who breaks into the process from being able to break into the wider system.

The process is only able to run inside the walls. Since nothing the process does in regards to executing code is supposed to be able to breach the walls, a detailed audit of its code is not needed in order to be able to say certain things about its security.

The walls might be a user ID, for example. This is the definition used in the [security\(7\)](#) and [named\(8\)](#) man pages.

Take the `ntalk` service, for example (see [inetd\(8\)](#)). This service used to run as user ID `root`. Now it runs as user ID `tty`. The `tty` user is a sandbox designed to make it more difficult for someone who has successfully hacked into the system via `ntalk` from being able to hack beyond that user ID.

- A process which is placed inside a simulation of the machine. It means that someone who is able to break into the process may believe that he can break into the wider machine but is, in fact, only breaking into a simulation of that machine and not modifying any real data.

The most common way to accomplish this is to build a simulated environment in a subdirectory and then run the processes in that directory chrooted so that `/` for that process is this directory, not the real `/` of the system).

Another common use is to mount an underlying file system read-only and then create a file system layer on top of it that gives a process a seemingly writeable view into that file system. The process may believe it is able to write to those files, but only the process sees the effects - other processes in the system do not, necessarily.

An attempt is made to make this sort of sandbox so transparent that the user (or hacker) does not realize that he is sitting in it.

UNIX® implements two core sandboxes. One is at the process level, and one is at the userid level.

Every UNIX® process is completely firewalled off from every other UNIX® process. One process cannot modify the address space of another.

A UNIX® process is owned by a particular userid. If the user ID is not the `root` user, it serves to firewall the process off from processes owned by other users. The user ID is also used to firewall off on-disk data.

13.2. What is `securelevel`?

`securelevel` is a security mechanism implemented in the kernel. When the `securelevel` is positive,

the kernel restricts certain tasks; not even the superuser (`root`) is allowed to do them. The `securelevel` mechanism limits the ability to:

- Unset certain file flags, such as `schg` (the system immutable flag).
- Write to kernel memory via `/dev/mem` and `/dev/kmem`.
- Load kernel modules.
- Alter firewall rules.

To check the status of the `securelevel` on a running system:

```
# sysctl -n kern.securelevel
```

The output contains the current value of the `securelevel`. If it is greater than 0, at least some of the `securelevel`'s protections are enabled.

The `securelevel` of a running system cannot be lowered as this would defeat its purpose. If a task requires that the `securelevel` be non-positive, change the `kern_securelevel` and `kern_securelevel_enable` variables in `/etc/rc.conf` and reboot.

For more information on `securelevel` and the specific things all the levels do, consult [init\(8\)](#).



`Securelevel` is not a silver bullet; it has many known deficiencies. More often than not, it provides a false sense of security.

One of its biggest problems is that in order for it to be at all effective, all files used in the boot process up until the `securelevel` is set must be protected. If an attacker can get the system to execute their code prior to the `securelevel` being set (which happens quite late in the boot process since some things the system must do at start-up cannot be done at an elevated `securelevel`), its protections are invalidated. While this task of protecting all files used in the boot process is not technically impossible, if it is achieved, system maintenance will become a nightmare since one would have to take the system down, at least to single-user mode, to modify a configuration file.

This point and others are often discussed on the mailing lists, particularly the [FreeBSD security mailing list](#). Search the archives [here](#) for an extensive discussion. A more fine-grained mechanism is preferred.

13.3. What is this UID 0 toor account? Have I been compromised?

Do not worry. `toor` is an "alternative" superuser account, where `toor` is `root` spelled backwards. It is intended to be used with a non-standard shell so the default shell for `root` does not need to change. This is important as shells which are not part of the base distribution, but are instead installed from ports or packages, are installed in `/usr/local/bin` which, by default, resides on a different file system. If `root`'s shell is located in `/usr/local/bin` and the file system containing `/usr/local/bin` is not

mounted, `root` will not be able to log in to fix a problem and will have to reboot into single-user mode in order to enter the path to a shell.

Some people use `toor` for day-to-day `root` tasks with a non-standard shell, leaving `root`, with a standard shell, for single-user mode or emergencies. By default, a user cannot log in using `toor` as it does not have a password, so log in as `root` and set a password for `toor` before using it to login.

Chapter 14. Serial Communications

This section answers common questions about serial communications with FreeBSD.

14.1. How do I get the boot: prompt to show on the serial console?

See [this section of the Handbook](#).

14.2. How do I tell if FreeBSD found my serial ports or modem cards?

As the FreeBSD kernel boots, it will probe for the serial ports for which the kernel is configured. Either watch the boot messages closely or run this command after the system is up and running:

```
% grep -E '^(sio|uart)[0-9]' < /var/run/dmesg.boot
uart0: <16550 or compatible> port 0x3f8-0x3ff irq 4 flags 0x10 on acpi0
uart0: console (115200,n,8,1)
uart1: <16550 or compatible> port 0x2f8-2x3ff irq 3 on acpi0
```

This example shows two serial ports. The first is on IRQ4, port address `0x3f8`, and has a 16550-type UART chip. The second uses the same kind of chip but is on IRQ3 and is at port address `0x2f8`. Internal modem cards are treated just like serial ports, except that they always have a modem attached to the port.

The GENERIC kernel includes support for two serial ports using the same IRQ and port address settings in the above example. If these settings are not right for the system, or if there are more modem cards or serial ports than the kernel is configured for, reconfigure using the instructions in [building a kernel](#) for more details.

14.3. How do I access the serial ports on FreeBSD? (x86-specific)

The third serial port, `sio2`, or COM3, is on `/dev/cuad2` for dial-out devices, and on `/dev/ttyd2` for dial-in devices. What is the difference between these two classes of devices?

When opening `/dev/ttydX` in blocking mode, a process will wait for the corresponding `cuadX` device to become inactive, and then wait for the carrier detect line to go active. When the `cuadX` device is opened, it makes sure the serial port is not already in use by the `tydX` device. If the port is available, it steals it from the `tydX` device. Also, the `cuadX` device does not care about carrier detect. With this scheme and an auto-answer modem, remote users can log in and local users can still dial out with the same modem and the system will take care of all the conflicts.

14.4. How do I enable support for a multi-port serial card?

The section on kernel configuration provides information about configuring the kernel. For a multi-port serial card, place an `sio(4)` line for each serial port on the card in the `device.hints(5)` file. But place the IRQ specifiers on only one of the entries. All of the ports on the card should share one IRQ. For consistency, use the last serial port to specify the IRQ. Also, specify the following option in the kernel configuration file:

```
options COM_MULTIPORT
```

The following `/boot/device.hints` example is for an AST 4-port serial card on IRQ 12:

```
hint.sio.4.at="isa"
hint.sio.4.port="0x2a0"
hint.sio.4.flags="0x701"
hint.sio.5.at="isa"
hint.sio.5.port="0x2a8"
hint.sio.5.flags="0x701"
hint.sio.6.at="isa"
hint.sio.6.port="0x2b0"
hint.sio.6.flags="0x701"
hint.sio.7.at="isa"
hint.sio.7.port="0x2b8"
hint.sio.7.flags="0x701"
hint.sio.7.irq="12"
```

The flags indicate that the master port has minor number `7 (0x700)`, and all the ports share an IRQ (`0x001`).

14.5. Can I set the default serial parameters for a port?

See the [Serial Communications](#) section in the FreeBSD Handbook.

14.6. Why can I not run `tip` or `cu`?

The built-in `tip(1)` and `cu(1)` utilities can only access the `/var/spool/lock` directory via user `uucp` and group `dialer`. Use the `dialer` group to control who has access to the modem or remote systems by adding user accounts to `dialer`.

Alternatively, everyone can be configured to run `tip(1)` and `cu(1)` by typing:

```
# chmod 4511 /usr/bin/cu
# chmod 4511 /usr/bin/tip
```

Chapter 15. Miscellaneous Questions

15.1. FreeBSD uses a lot of swap space even when the computer has free memory left. Why?

FreeBSD will proactively move entirely idle, unused pages of main memory into swap in order to make more main memory available for active use. This heavy use of swap is balanced by using the extra free memory for caching.

Note that while FreeBSD is proactive in this regard, it does not arbitrarily decide to swap pages when the system is truly idle. Thus, the system will not be all paged out after leaving it idle overnight.

15.2. Why does `top` show very little free memory even when I have very few programs running?

The simple answer is that free memory is wasted memory. Any memory that programs do not actively allocate is used within the FreeBSD kernel as disk cache. The values shown by `top(1)` labeled as **Inact** and **Laundry** are cached data at different aging levels. This cached data means the system does not have to access a slow disk again for data it has accessed recently, thus increasing overall performance. In general, a low value shown for **Free** memory in `top(1)` is good, provided it is not *very* low.

15.3. Why will `chmod` not change the permissions on symlinks?

Symlinks do not have permissions, and by default, `chmod(1)` will follow symlinks to change the permissions on the source file, if possible. For the file, `foo` with a symlink named `bar`, this command will always succeed.

```
% chmod g-w bar
```

However, the permissions on `bar` will not have changed.

When changing modes of the file hierarchies rooted in the files instead of the files themselves, use either `-H` or `-L` together with `-R` to make this work. See `chmod(1)` and `symlink(7)` for more information.



`-R` does a *recursive* `chmod(1)`. Be careful about specifying directories or symlinks to directories to `chmod(1)`. To change the permissions of a directory referenced by a symlink, use `chmod(1)` without any options and follow the symlink with a trailing slash (`/`). For example, if `foo` is a symlink to directory `bar`, to change the permissions of `foo` (actually `bar`), do something like:


```
% chmod 555 foo/
```

With the trailing slash, `chmod(1)` will follow the symlink, `foo`, to change the permissions of the directory, `bar`.

15.4. Can I run DOS binaries under FreeBSD?

Yes. A DOS emulation program, [emulators/doscmd](#), is available in the FreeBSD Ports Collection.

If `doscmd` will not suffice, [emulators/pccemu](#) emulates an 8088 and enough BIOS services to run many DOS text-mode applications. It requires the X Window System.

The Ports Collection also has [emulators/dosbox](#). The main focus of this application is emulating old DOS games using the local file system for files.

15.5. What do I need to do to translate a FreeBSD document into my native language?

See the [Translation FAQ](#) in the FreeBSD Documentation Project Primer.

15.6. Why does my email to any address at FreeBSD.org bounce?

The [FreeBSD.org](#) mail system implements some Postfix checks on incoming mail and rejects mail that is either from misconfigured relays or otherwise appears likely to be spam. Some of the specific requirements are:

- The IP address of the SMTP client must "reverse-resolve" to a forward confirmed hostname.
- The fully-qualified hostname given in the SMTP conversation (either HELO or EHLO) must resolve to the IP address of the client.

Other advice to help mail reach its destination include:

- Mail should be sent in plain text, and messages sent to mailing lists should generally be no more than 200KB in length.
- Avoid excessive cross posting. Choose *one* mailing list which seems most relevant and send it there.

If you still have trouble with email infrastructure at [FreeBSD.org](#), send a note with the details to postmaster@freebsd.org; Include a date/time interval so that logs may be reviewed - and note that we only keep one week's worth of mail logs. (Be sure to specify the time zone or offset from UTC.)

15.7. Where can I find a free FreeBSD account?

While FreeBSD does not provide open access to any of their servers, others do provide open access UNIX® systems. The charge varies and limited services may be available.

[Arboret, Inc](#), also known as *M-Net*, has been providing open access to UNIX® systems since 1983. Starting on an Altos running System III, the site switched to BSD/OS in 1991. In June of 2000, the site switched again to FreeBSD. *M-Net* can be accessed via telnet and SSH and provides basic access to the entire FreeBSD software suite. However, network access is limited to members and patrons who donate to the system, which is run as a non-profit organization. *M-Net* also provides an bulletin board system and interactive chat.

15.8. What is the cute little red guy's name?

He does not have one, and is just called "the BSD daemon". If you insist upon using a name, call him "beastie". Note that "beastie" is pronounced "BSD".

More about the BSD daemon is available on his [home page](#).

15.9. Can I use the BSD daemon image?

Perhaps. The BSD daemon is copyrighted by Marshall Kirk McKusick. Check his [Statement on the Use of the BSD Daemon Figure](#) for detailed usage terms.

In summary, the image can be used in a tasteful manner, for personal use, so long as appropriate credit is given. Before using the logo commercially, contact Kirk McKusick <mckusick@FreeBSD.org> for permission. More details are available on the [BSD Daemon's home page](#).

15.10. Do you have any BSD daemon images I could use?

Xfig and eps drawings are available under `/usr/share/examples/BSD_daemon/`.

15.11. I have seen an acronym or other term on the mailing lists and I do not understand what it means. Where should I look?

Refer to the [FreeBSD Glossary](#).

15.12. Why should I care what color the bikeshed is?

The really, really short answer is that you should not. The somewhat longer answer is that just because you are capable of building a bikeshed does not mean you should stop others from building one just because you do not like the color they plan to paint it. This is a metaphor

indicating that you need not argue about every little feature just because you know enough to do so. Some people have commented that the amount of noise generated by a change is inversely proportional to the complexity of the change.

The longer and more complete answer is that after a very long argument about whether [sleep\(1\)](#) should take fractional second arguments, Poul-Henning Kamp <phk@FreeBSD.org> posted a long message entitled [A bike shed \(any color will do\) on greener grass....](#) The appropriate portions of that message are quoted below.

“What is it about this bike shed?” some of you have asked me.

It is a long story, or rather it is an old story, but it is quite short actually. C. Northcote Parkinson wrote a book in the early 1960s, called “Parkinson’s Law”, which contains a lot of insight into the dynamics of management.

In the specific example involving the bike shed, the other vital component is an atomic power-plant, I guess that illustrates the age of the book.

Parkinson shows how you can go into the board of directors and get approval for building a multi-million or even billion dollar atomic power plant, but if you want to build a bike shed you will be tangled up in endless discussions.

Parkinson explains that this is because an atomic plant is so vast, so expensive and so complicated that people cannot grasp it, and rather than try, they fall back on the assumption that somebody else checked all the details before it got this far. Richard P. Feynmann gives a couple of interesting, and very much to the point, examples relating to Los Alamos in his books.

A bike shed on the other hand. Anyone can build one of those over a weekend, and still have time to watch the game on TV. So no matter how well prepared, no matter how reasonable you are with your proposal, somebody will seize the chance to show that he is doing his job, that he is paying attention, that he is here.

In Denmark we call it “setting your fingerprint”. It is about personal pride and prestige, it is about being able to point somewhere and say “There! I did that.”. It is a strong trait in politicians, but present in most people given the chance. Just think about footsteps in wet cement.

--Poul-Henning Kamp <phk@FreeBSD.org> on freebsd-hackers, October 2, 1999

Chapter 16. The FreeBSD Funnies

16.1. How cool is FreeBSD?

Q. Has anyone done any temperature testing while running FreeBSD? I know Linux® runs cooler than DOS, but have never seen a mention of FreeBSD. It seems to run really hot.

A. No, but we have done numerous taste tests on blindfolded volunteers who have also had 250 micrograms of LSD-25 administered beforehand. 35% of the volunteers said that FreeBSD tasted sort of orange, whereas Linux® tasted like purple haze. Neither group mentioned any significant variances in temperature. We eventually had to throw the results of this survey out entirely anyway when we found that too many volunteers were wandering out of the room during the tests, thus skewing the results. We think most of the volunteers are at Apple now, working on their new "scratch and sniff" GUI. It is a funny old business we are in!

Seriously, FreeBSD uses the HLT (halt) instruction when the system is idle thus lowering its energy consumption and therefore the heat it generates. Also if you have ACPI (Advanced Configuration and Power Interface) configured, then FreeBSD can also put the CPU into a low power mode.

16.2. Who is scratching in my memory banks??

Q. Is there anything "odd" that FreeBSD does when compiling the kernel which would cause the memory to make a scratchy sound? When compiling (and for a brief moment after recognizing the floppy drive upon startup, as well), a strange scratchy sound emanates from what appears to be the memory banks.

A. Yes! You will see frequent references to "daemons" in the BSD documentation, and what most people do not know is that this refers to genuine, non-corporeal entities that now possess your computer. The scratchy sound coming from your memory is actually high-pitched whispering exchanged among the daemons as they best decide how to deal with various system administration tasks.

If the noise gets to you, a good `fdisk /mbr` from DOS will get rid of them, but do not be surprised if they react adversely and try to stop you. In fact, if at any point during the exercise you hear the satanic voice of Bill Gates coming from the built-in speaker, take off running and do not ever look back! Freed from the counterbalancing influence of the BSD daemons, the twin demons of DOS and Windows® are often able to re-assert total control over your machine to the eternal damnation of your soul. Now that you know, given a choice you would probably prefer to get used to the scratchy noises, no?

16.3. How many FreeBSD hackers does it take to change a lightbulb?

One thousand, one hundred and sixty-nine:

Twenty-three to complain to -CURRENT about the lights being out;

Four to claim that it is a configuration problem, and that such matters really belong on -questions;

Three to submit PRs about it, one of which is misfiled under doc and consists only of "it's dark";

One to commit an untested lightbulb which breaks buildworld, then back it out five minutes later;

Eight to flame the PR originators for not including patches in their PRs;

Five to complain about buildworld being broken;

Thirty-one to answer that it works for them, and they must have updated at a bad time;

One to post a patch for a new lightbulb to -hackers;

One to complain that he had patches for this three years ago, but when he sent them to -CURRENT they were just ignored, and he has had bad experiences with the PR system; besides, the proposed new lightbulb is non-reflexive;

Thirty-seven to scream that lightbulbs do not belong in the base system, that committers have no right to do things like this without consulting the Community, and WHAT IS -CORE DOING ABOUT IT!?

Two hundred to complain about the color of the bicycle shed;

Three to point out that the patch breaks [style\(9\)](#);

Seventeen to complain that the proposed new lightbulb is under GPL;

Five hundred and eighty-six to engage in a flame war about the comparative advantages of the GPL, the BSD license, the MIT license, the NPL, and the personal hygiene of unnamed FSF founders;

Seven to move various portions of the thread to -chat and -advocacy;

One to commit the suggested lightbulb, even though it shines dimmer than the old one;

Two to back it out with a furious flame of a commit message, arguing that FreeBSD is better off in the dark than with a dim lightbulb;

Forty-six to argue vociferously about the backing out of the dim lightbulb and demanding a statement from -core;

Eleven to request a smaller lightbulb so it will fit their Tamagotchi if we ever decide to port FreeBSD to that platform;

Seventy-three to complain about the SNR on -hackers and -chat and unsubscribe in protest;

Thirteen to post "unsubscribe", "How do I unsubscribe?", or "Please remove me from the list", followed by the usual footer;

One to commit a working lightbulb while everybody is too busy flaming everybody else to notice;

Thirty-one to point out that the new lightbulb would shine 0.364% brighter if compiled with TenDRA (although it will have to be reshaped into a cube), and that FreeBSD should therefore

switch to TenDRA instead of GCC;

One to complain that the new lightbulb lacks fairings;

Nine (including the PR originators) to ask "what is MFC?";

Fifty-seven to complain about the lights being out two weeks after the bulb has been changed.

Nik Clayton <nik@FreeBSD.org> adds:

I was laughing quite hard at this.

And then I thought, "Hang on, shouldn't there be '1 to document it.' in that list somewhere?"

And then I was enlightened :-)

Thomas Abthorpe <tabthorpe@FreeBSD.org> says: "None, real FreeBSD hackers are not afraid of the dark!"

16.4. Where does data written to /dev/null go?

It goes into a special data sink in the CPU where it is converted to heat which is vented through the heatsink / fan assembly. This is why CPU cooling is increasingly important; as people get used to faster processors, they become careless with their data and more and more of it ends up in /dev/null, overheating their CPUs. If you delete /dev/null (which effectively disables the CPU data sink) your CPU may run cooler but your system will quickly become constipated with all that excess data and start to behave erratically. If you have a fast network connection you can cool down your CPU by reading data out of /dev/random and sending it off somewhere; however you run the risk of overheating your network connection and / or angering your ISP, as most of the data will end up getting converted to heat by their equipment, but they generally have good cooling, so if you do not overdo it you should be OK.

Paul Robinson adds:

There are other methods. As every good sysadmin knows, it is part of standard practice to send data to the screen of interesting variety to keep all the pixies that make up your picture happy. Screen pixies (commonly mis-typed or re-named as "pixels") are categorized by the type of hat they wear (red, green or blue) and will hide or appear (thereby showing the color of their hat) whenever they receive a little piece of food. Video cards turn data into pixie-food, and then send them to the pixies - the more expensive the card, the better the food, so the better behaved the pixies are. They also need constant stimulation - this is why screen savers exist.

To take your suggestions further, you could just throw the random data to console, thereby letting the pixies consume it. This causes no heat to be produced at all, keeps the pixies happy and gets rid of your data quite quickly, even if it does make things look a bit messy on your screen.

Incidentally, as an ex-admin of a large ISP who experienced many problems attempting to maintain a stable temperature in a server room, I would strongly discourage people sending the data they do not want out to the network. The fairies who do the packet switching and routing get annoyed by it as well.

16.5. My colleague sits at the computer too much, how can I prank her?

Install [games/sl](#) and wait for her to mistype `sl` for `ls`.

Chapter 17. Advanced Topics

17.1. How can I learn more about FreeBSD's internals?

See the [FreeBSD Architecture Handbook](#).

Additionally, much general UNIX® knowledge is directly applicable to FreeBSD.

17.2. How can I contribute to FreeBSD? What can I do to help?

We accept all types of contributions: documentation, code, and even art. See the article on [Contributing to FreeBSD](#) for specific advice on how to do this.

And thanks for the thought!

17.3. What are snapshots and releases?

There are currently 3 active/semi-active branches in the FreeBSD [Git Repository](#). (Earlier branches are only changed very rarely, which is why there are only 3 active branches of development):

- stable/12 AKA 12-STABLE
- stable/13 AKA 13-STABLE
- main AKA *-CURRENT* AKA 14.0

Right now, *-CURRENT* is the 14.X development stream; the 13-STABLE branch, stable/13, forked off from *-CURRENT* in January 2021 and the 12-STABLE branch, stable/12, forked off from *-CURRENT* in December 2018.

17.4. How can I make the most of the data I see when my kernel panics?

Here is typical kernel panic:

```
Fatal trap 12: page fault while in kernel mode
fault virtual address   = 0x40
fault code              = supervisor read, page not present
instruction pointer     = 0x8:0xf014a7e5
stack pointer          = 0x10:0xf4ed6f24
frame pointer          = 0x10:0xf4ed6f28
code segment           = base 0x0, limit 0xffff, type 0x1b
                       = DPL 0, pres 1, def32 1, gran 1
processor eflags       = interrupt enabled, resume, IOPL = 0
current process        = 80 (mount)
interrupt mask         =
```



```
trap number          = 12
panic: page fault
```

This message is not enough. While the instruction pointer value is important, it is also configuration dependent as it varies depending on the kernel image. If it is a GENERIC kernel image from one of the snapshots, it is possible for somebody else to track down the offending function, but for a custom kernel, only you can tell us where the fault occurred.

To proceed:

1. Write down the instruction pointer value. Note that the `0x8:` part at the beginning is not significant in this case: it is the `0xf0xxxxxx` part that we want.
2. When the system reboots, do the following:

```
% nm -n kernel.that.caused.the.panic | grep f0xxxxxx
```

where `f0xxxxxx` is the instruction pointer value. The odds are you will not get an exact match since the symbols in the kernel symbol table are for the entry points of functions and the instruction pointer address will be somewhere inside a function, not at the start. If you do not get an exact match, omit the last digit from the instruction pointer value and try again:

```
% nm -n kernel.that.caused.the.panic | grep f0xxxxx
```

If that does not yield any results, chop off another digit. Repeat until there is some sort of output. The result will be a possible list of functions which caused the panic. This is a less than exact mechanism for tracking down the point of failure, but it is better than nothing.

However, the best way to track down the cause of a panic is by capturing a crash dump, then using [kgdb\(1\)](#) to generate a stack trace on the crash dump.

In any case, the method is this:

1. Make sure that the following line is included in the kernel configuration file:

```
makeoptions          DEBUG=-g          # Build kernel with gdb(1) debug symbols
```

2. Change to the `/usr/src` directory:

```
# cd /usr/src
```

3. Compile the kernel:

```
# make buildkernel KERNCONF=MYKERNEL
```

4. Wait for [make\(1\)](#) to finish compiling.

```
# make installkernel KERNCONF=MYKERNEL
```

5. Reboot.



If **KERNCONF** is not included, the GENERIC kernel will instead be built and installed.

The [make\(1\)](#) process will have built two kernels. `/usr/obj/usr/src/sys/MYKERNEL/kernel` and `/usr/obj/usr/src/sys/MYKERNEL/kernel.debug`. `kernel` was installed as `/boot/kernel/kernel`, while `kernel.debug` can be used as the source of debugging symbols for [kgdb\(1\)](#).

To capture a crash dump, edit `/etc/rc.conf` and set **dumpdev** to point to either the swap partition or **AUTO**. This will cause the [rc\(8\)](#) scripts to use the [dumpon\(8\)](#) command to enable crash dumps. This command can also be run manually. After a panic, the crash dump can be recovered using [savecore\(8\)](#); if **dumpdev** is set in `/etc/rc.conf`, the [rc\(8\)](#) scripts will run [savecore\(8\)](#) automatically and put the crash dump in `/var/crash`.



FreeBSD crash dumps are usually the same size as physical RAM. Therefore, make sure there is enough space in `/var/crash` to hold the dump. Alternatively, run [savecore\(8\)](#) manually and have it recover the crash dump to another directory with more room. It is possible to limit the size of the crash dump by using **options MAXMEM=N** where *N* is the size of kernel's memory usage in KBs. For example, for 1 GB of RAM, limit the kernel's memory usage to 128 MB, so that the crash dump size will be 128 MB instead of 1 GB.

Once the crash dump has been recovered , get a stack trace as follows:

```
% kgdb /usr/obj/usr/src/sys/MYKERNEL/kernel.debug /var/crash/vmcore.0  
(kgdb) backtrace
```

Note that there may be several screens worth of information. Ideally, use [script\(1\)](#) to capture all of them. Using the unstripped kernel image with all the debug symbols should show the exact line of kernel source code where the panic occurred. The stack trace is usually read from the bottom up to trace the exact sequence of events that lead to the crash. [kgdb\(1\)](#) can also be used to print out the contents of various variables or structures to examine the system state at the time of the crash.



If a second computer is available, [kgdb\(1\)](#) can be configured to do remote debugging, including setting breakpoints and single-stepping through the kernel code.



If **ddb** is enabled and the kernel drops into the debugger, a panic and a crash dump

can be forced by typing `panic` at the `ddb` prompt. It may stop in the debugger again during the panic phase. If it does, type `continue` and it will finish the crash dump.

17.5. Why has `dlsym()` stopped working for ELF executables?

The ELF toolchain does not, by default, make the symbols defined in an executable visible to the dynamic linker. Consequently `dlsym()` searches on handles obtained from calls to `dlopen(NULL, flags)` will fail to find such symbols.

To search, using `dlsym()`, for symbols present in the main executable of a process, link the executable using the `--export-dynamic` option to the ELF linker (`ld(1)`).

17.6. How can I increase or reduce the kernel address space on i386?

By default, the kernel address space is 1 GB (2 GB for PAE) for i386. When running a network-intensive server or using ZFS, this will probably not be enough.

Add the following line to the kernel configuration file to increase available space and rebuild the kernel:

```
options KVA_PAGES=N
```

To find the correct value of N , divide the desired address space size (in megabytes) by four. (For example, it is `512` for 2 GB.)

Chapter 18. Acknowledgments

This innocent little Frequently Asked Questions document has been written, rewritten, edited, folded, spindled, mutilated, eviscerated, contemplated, discombobulated, cogitated, regurgitated, rebuilt, castigated, and reinvigorated over the last decade, by a cast of hundreds if not thousands. Repeatedly.

We wish to thank every one of the people responsible, and we encourage you to [join them](#) in making this FAQ even better.