

sub title

# **The Title**

Title:  
Subtitle  
Month 2006

NetBSD and pkgsrc are registered trademarks of the NetBSD Foundation, Inc.

FreeBSD is a registered trademark of the FreeBSD Foundation.

# Contents

- Introduction** **v**
- 0.1 Credits . . . . . v
- 0.2 Conventions . . . . . v
  
- 1 Chapter Installing and Upgrading the OS and Software** **1**
- 1.1 Recognize the installation program used by each operating system . . . . . 2
- 1.2 Recognize which commands are available for upgrading the operating system . . . . . 8
- 1.3 Understand the difference between a pre-compiled binary and compiling from source . . . . . 10
- 1.4 Understand when it is preferable to install a pre-compiled binary and how to do so . . . . . 11
- 1.5 Recognize the available methods for compiling a customized binary . . . . . 12
- 1.6 Determine what software is installed on a system . 12
- 1.7 Determine which software requires upgrading . . . 13
- 1.8 Upgrade installed software . . . . . 14
- 1.9 Determine which software have outstanding security advisories . . . . . 14
- 1.10 Follow the instructions in a security advisory to apply a security patch . . . . . 15
  
- 2 Chapter Securing the Operating System** **17**
- 2.1 Determine the system’s security level . . . . . 18
- 2.2 Configure an SSH server according to a set of requirements . . . . . 19

2.3	Configure an SSH server to use a key pair for authentication . . . . .	19
2.4	Preserve existing SSH host keys during a system upgrade . . . . .	20
2.5	Recognize alternate authentication mechanisms . .	21
2.6	Recognize alternate authorization schemes . . . .	21
2.7	Recognize basic recommended access methods . .	22
2.8	Recognize BSD firewalls and rulesets . . . . .	23
2.9	Recognize BSD mechanisms for encrypting devices	23
2.10	Recognize methods for verifying the validity of binaries . . . . .	24
2.11	Recognize the BSD methods for restraining a service	25
2.12	Change the encryption algorithm used to encrypt the password database . . . . .	25
2.13	Modify the system banner . . . . .	26
2.14	Protect authentication data . . . . .	27
<b>3</b>	<b>Chapter Files, Filesystems and Disks</b>	<b>29</b>
3.1	Mount or unmount local filesystems . . . . .	30
3.2	Configure data to be available through NFS . . . .	31
3.3	Determine which filesystems are currently mounted and which will be mounted at system boot . . . . .	32
3.4	Determine disk capacity and which files are consuming the most disk space . . . . .	32
3.5	Create and view symbolic or hard links . . . . .	33
3.6	View and modify ACLs . . . . .	33
3.7	View file permissions and modify them using either symbolic or octal mode . . . . .	34
3.8	Modify a file's owner or group . . . . .	35
3.9	Backup and restore a specified set of files and directories to local disk or tape . . . . .	35
3.10	Backup and restore a file system . . . . .	36
3.11	Determine the directory structure of a system . . .	37
3.12	Manually run the file system checker and repair tool	37
3.13	View and modify file flags . . . . .	39
3.14	Monitor the virtual memory system . . . . .	39

<b>4</b>	<b>Chapter Users and Accounts Management</b>	<b>41</b>
4.1	Create, modify and remove user accounts . . . . .	42
4.2	Create a system account . . . . .	42
4.3	Lock a user account or reset a locked user account	43
4.4	Determine identity and group membership . . . . .	43
4.5	Determine who is currently on the system or the last time a user was on the system . . . . .	47
4.6	Enable accounting and view system usage statistics	48
4.7	Change a user's default shell . . . . .	49
4.8	Control which files are copied to a new user's home directory during account creation . . . . .	50
4.9	Change a password . . . . .	51
<b>5</b>	<b>Chapter Basic System Administration</b>	<b>53</b>
5.1	Determine which process are consuming the most CPU . . . . .	54
5.2	View and send signals to active processes . . . . .	58
5.3	Use an rc.d script to determine if a service is running and start, restart or stop it as required . . . . .	59
5.4	View and configure system hardware . . . . .	63
5.5	View, load, or unload a kernel module . . . . .	64
5.6	Modify a kernel parameter on the fly . . . . .	65
5.7	View the status of a software RAID mirror or stripe	68
5.8	Determine which MTA is being used on the system	72
5.9	Configure system logging . . . . .	73
5.10	Review log files to troubleshoot and monitor system behavior . . . . .	73
5.11	Understand basic printer troubleshooting . . . . .	74
5.12	Create or modify email aliases for Sendmail or Postfix	75
5.13	Halt, reboot, or bring the system to single-user mode	75
5.14	Recognize the difference between hard and soft limits and modify existing resource limits . . . . .	76
5.15	Recognize the BSD utilities that shape traffic or control bandwidth . . . . .	77
5.16	Recognize common, possibly third-party, server configuration files . . . . .	77
5.17	Configure a service to start at boot time . . . . .	79

5.18	Configure the scripts that run periodically to perform various system maintenance tasks . . . . .	80
5.19	View the Sendmail or Postfix mail queue . . . . .	80
5.20	Determine the last system boot time and the workload on the system . . . . .	81
5.21	Monitor disk input/output . . . . .	82
5.22	Deal with busy devices . . . . .	83
5.23	Determine information regarding the operating system . . . . .	83
5.24	Understand the advantages of using a BSD license	84
<b>6</b>	<b>Chapter Network Administration</b>	<b>87</b>
6.1	Determine the current TCP/IP settings on a system	88
6.2	Set a system's TCP/IP settings . . . . .	91
6.3	Determine which TCP or UDP ports are open on a system . . . . .	91
6.4	Verify the availability of a TCP/IP service . . . . .	92
6.5	Query a DNS server . . . . .	92
6.6	Determine who is responsible for a DNS zone . . .	93
6.7	Change the order of name resolution . . . . .	96
6.8	Convert a subnet mask between dotted decimal, hexadecimal or CIDR notation . . . . .	97
6.9	Gather information using an IP address and subnet mask . . . . .	99
6.10	Understand IPv6 address theory . . . . .	99
6.11	Demonstrate basic tcpdump(1) skills . . . . .	100
6.12	Manipulate ARP and neighbor discovery caches .	101
6.13	Configure a system to use NTP . . . . .	101
6.14	View and renew a DHCP lease . . . . .	102
6.15	Recognize when and how to set or remove an interface alias . . . . .	103
<b>7</b>	<b>Chapter Basic Unix Skills</b>	<b>105</b>
7.1	Demonstrate proficiency in using redirection, pipes and tees . . . . .	106
7.2	Recognize, view and modify environmental variables	106
7.3	Be familiar with the vi(1) editor . . . . .	109

7.4	Determine if a file is a binary, text, or data file . . .	110
7.5	Locate files and binaries on a system . . . . .	111
7.6	Find a file with a given set of attributes . . . . .	111
7.7	Create a simple Bourne shell script . . . . .	112
7.8	Find appropriate documentation . . . . .	112
7.9	Recognize the different sections of the manual . .	113
7.10	Verify a file's message digest fingerprint (checksum)	115
7.11	Demonstrate familiarity with the default shell . . .	116
7.12	Read mail on the local system . . . . .	117
7.13	Use job control . . . . .	117
7.14	Demonstrate proficiency with regular expressions .	118
7.15	Overcome command line length limitations . . . .	119
7.16	Understand various "domain" contexts . . . . .	119
7.17	Configure an action to be scheduled by cron(8) . .	120
<b>Index</b>		<b>120</b>



# Introduction

Author: Jeremy C. Reed reed NetBSD/FreeBSD/OpenBSD/DragonFly

Reviewer: Cezary Morga cm at therek dot net FreeBSD

Reviewer: *name contact BSD flavour*

---

Welcome to the Quick Guide to BSD Administration. This book is a quick reference and great way to quickly learn BSD administration skills. These topics are based on the objectives published by the BSD Certification Group in the 2005 BSDA Certification Requirements Document. The BSDA (BSD Associate) Certification is for BSD Unix system administrators with light to moderate skills.

This book provides basic examples and pointers to further documentation and learning resources. This book is not a comprehensive reference. While this is a beginner's book, it is also useful for experienced administrators.

This book covers generic \*BSD administration and specific skills as necessary for NetBSD, FreeBSD, OpenBSD and DragonFly BSD.

## 0.1 Credits

TODO: this section might be partially generated from the list of known authors and technical reviewers.

## 0.2 Conventions

TODO: this section will describe the format and typefaces used for examples, input, output, pathnames, etc. as to be seen in the final printed format. The style guide will document how this can be done in the wiki.



# 1 Chapter Installing and Upgrading the OS and Software

Author: Ion-Mihai Tetcu itetcu@FreeBSD.org FreeBSD

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

---

XXX: I plan to write only the FreeBSD part so we still need authors for the rest (itecu)

An important aspect of system administration is tracking installed versions of both the operating system and third-party applications. An advantage of using BSD systems is the availability of multiple tools to assist the system administrator in determining software versions and their dependencies. These tools indicate which software is out-of-date or has existing security vulnerabilities. Assist in upgrading or patching software and its dependencies. When and how installations and upgrades are done is specific to each organization. The successful admin knows how to use the tools which are available for these purposes, and the cautions that are necessary when working on production systems under the supervision of a more senior administrator.

- Recognize the installation program used by each operating system
- Recognize which commands are available for upgrading the operating system
- Understand the difference between a pre-compiled binary and compiling from source
- Understand when it is preferable to install a pre-compiled binary and how to do so

- Recognize the available methods for compiling a customized binary
- Determine what software is installed on a system
- Determine which software requires upgrading
- Upgrade installed software
- Determine which software have outstanding security advisories
- Follow the instructions in a security advisory to apply a security patch

## 1.1 Recognize the installation program used by each operating system

Author: hubertf *contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD/OpenBSD

---

### 1.1.1 Concept

While BSDA candidates are not expected to plan an installation, they should be able to start and complete an installation according to a provided list of requirements. Since the install procedure is operating system dependent, it is recommended that the candidate have prior experience in the default install routine for each tested BSD operating system. Have some familiarity with release numbering practices in general (e.g. “dot-zero releases”) and where to find the release engineering practices at each BSD project’s website.

### 1.1.2 Introduction

This section first goes into release namings, then describes how to access the installer.

## 1.1. RECOGNIZE THE INSTALLATION PROGRAM USED BY EACH OPERATING SYSTEM

---

### 1.1.2.1 Release naming

The list below details the release names as shown e.g. by “uname -r” for a given operating system version. This may be different from the branch names used in any version control system, e.g. the stable branch that leads up to NetBSD 4.1 lists version numbers as 4.1\_BETA from “uname -r”, in CVS the branch is called “netbsd-4”. The list below covers the former data, the latter item is covered elsewhere.

The following release version numbers are available:

- Development branch (“-current”) naming scheme:
  - NetBSD: 4.99.x (bumped for kernel API/ABI changes)
  - FreeBSD: 7.0-CURRENT
  - OpenBSD: 4.0-current
- Alpha release naming scheme:
  - NetBSD: -
  - FreeBSD:
  - OpenBSD:
- Beta release naming scheme:
  - NetBSD: 4.0\_BETA, 4.1\_BETA, ...
  - FreeBSD: 6.2-BETA1, 6.2-BETA2, ...
  - OpenBSD: 4.0-beta
- Release candidat naming scheme:
  - NetBSD: 4.0\_RC1, 4.0\_RC2,
  - FreeBSD: 6.2-RC1, 6.2-RC2, ...
  - OpenBSD:
- Full (major / “dot”) release naming scheme:
  - NetBSD: 4.0, 5.0, ...
  - FreeBSD: 5.0-RELEASE, 6.0-RELEASE, ...

- OpenBSD: 3.8-release, 3.9-release, 4.0-release ....
- Stable branch version naming scheme:
  - NetBSD: 3.0\_STABLE, 3.1\_STABLE, 5.0\_STABLE
  - FreeBSD: 6.1-STABLE, 6.2-STABLE, ...
  - OpenBSD: 3.9-stable 4.0-stable
- Bugfix/feature update release naming scheme:
  - NetBSD: 3.1, 3.2, 4.1, 4.2, ...
  - FreeBSD:
  - OpenBSD:
- Security branch version naming scheme:
  - NetBSD: 3.0.1\_PATCH
  - FreeBSD: 6.1-RELEASE-p1, 6.1-RELEASE-p2, ...
  - OpenBSD: 4.0-stable
- Security update release naming scheme:
  - NetBSD: 3.1.0, 3.1.1, 4.2.1, 4.2.2, ...
  - FreeBSD:
  - OpenBSD:

### **1.1.2.2 Installer**

#### **NetBSD**

Most NetBSD ports use the 'sysinst' installer, a few still provide the old script-based installer as alternative. The installer is usually started automatically when booting install media, and doesn't need to be started manually. Install media in various formats (depending on the port) can be found in a NetBSD release's "installation" subdirectory.

Major, minor (stable) and security NetBSD releases can be found at [ftp.NetBSD.org](http://ftp.NetBSD.org) (and its mirrors) in /pub/NetBSD, ISO images are in /pub/NetBSD/iso and daily snapshots of the various branches can

## 1.1. RECOGNIZE THE INSTALLATION PROGRAM USED BY EACH OPERATING SYSTEM

---

be found on the same host in /pub/NetBSD-daily. The development branch “NetBSD-current” can be found in the “HEAD” directory.

### **FreeBSD**

For years now FreeBSD has used an installer known as ‘sysinstall’ to install its operating system on a variety of computer architectures. While most modern installers use graphical interfaces for ease of use, sysinstall is a text-based installer consisting of a series of menus used for configuring necessary installation parameters. Despite its appearance, however, sysinstall is more than adequate at performing common installation configurations, including partitioning hard disks, configuring network interfaces, creating additional users and adding third-party software. The traditional method for installing FreeBSD is often through the use of some form of boot media; for instance, floppy disks or compact discs. Booting into sysinstall is simply a matter of setting your BIOS to the correct “boot priority”, which would usually mean setting your floppy or CD-ROM drive as the first boot option.

BSDA candidates should be familiar with the menus and options presented by sysinstall during installation. This includes partitioning your system’s hard disks using the FDisk utility, applying the filesystem layout, choosing the correct distribution set, selecting the proper installation media, configuring local resources (such as network interfaces and timezone settings), adding user accounts, setting start-up services and adding additional software sets. BSDA candidates should also be capable of locating additional resources online through the use of search engines, forums and mailing list archives. The FreeBSD Handbook also covers installation using sysinstall in-depth and should be considered your primary resource for information. As with nearly every software utility in the \*BSD family, a manual page also exists describing both sysinstall’s features and purpose.

ISO images, used for creating bootable installation CDs, can be found from FreeBSD’s primary FTP server, ftp.FreeBSD.org. FreeBSD, through community support, also has numerous mirror sites available for downloading images. Depending on which site you choose, images for current and past releases may be available, including snapshots of both STABLE and CURRENT source branches. Ideally, only official RELEASE images should be used for production systems (e.g. 6.2-RELEASE). **OpenBSD**

OpenBSD installer is strait forward install script with no curses or X. For each architecture there is an INSTALL.[arch] which goes through the installation of OpenBSD on that architecture in detail. It behaves the same on all arch and/or method of installation. Installing OpenBSD is usually done using either a floppy boot image, a bootable CD, booting across a network eg PXE on i386 (not available on all architectures). Only changes in the BIOS/Open Firmware will select what installer will be run.

BSDA candidate should be familiar with different installation method, the options presented once the installer is started (Install/Upgrade/Shell), settings up disks using fdisk(8) and disklabel(8). The candidate should also know the different sets to be installed and what each one adds to the system, how to merge etc changes in case of update. After the first reboot the BSDA applicant should be able to add/delete users, group and add/remove packages and secure the system. These points will be discussed in greater depth later in the book.

### 1.1.3 Examples

#### NetBSD

Release version numbers: see above

Installer:

For NetBSD/i386, download e.g. the 'boot[12].fs' floppy images or the 'i386cd-.iso' ISO image. *Installation floppies for machines with little memory are in the 'boot-small?.fs' files, the 'bootlap-.fs' floppies have drivers for laptops, and the 'boot-com?.fs' images are useful for machines with serial consoles.* **FreeBSD**

For FreeBSD-6.2-RELEASE/i386, download the following images (for installation using floppies or CDs, respectively):

Floppy Images:

- boot.flp
- kern1.flp
- kern2.flp

ISO Images:

- 6.2-RELEASE-i386-disc1.iso

## 1.1. RECOGNIZE THE INSTALLATION PROGRAM USED BY EACH OPERATING SYSTEM

---

Verify the integrity of each downloaded image using either MD5 or SHA256. The images can then be written or burned to their respective media using a utility of your choice (such as `dd(1)` for floppy images or `burncd(8)` for the ISO images). Once the images are placed on their respective media, setting your system's BIOS to the correct boot sequence and booting the system is all that's left. Assuming everything goes without error, you should eventually be prompted with the initial `sysinstall` screen asking for you to choose your country/region.

### **OpenBSD**

For OpenBSD 4.0-release, download any of the following images (for installation using floppies or CDs, respectively) some of these images are not available on all platforms:

Floppy Images:

- floppy40.fs
- floppyB40.fs
- floppyC40.fs

ISO Images:

- cd40.iso
- cdemu40.iso

Verify the integrity of the downloaded image. Each image has a specific purpose. The candidate should be able to know which image to boot depending on the hardware.

### **1.1.4 Practice Exercises**

### **1.1.5 More information**

#### **1.1.5.1 Release naming**

<http://www.netbsd.org/Releases/release-map.html> for NetBSD

[http://www.freebsd.org/doc/en\\_US.ISO8859-1/articles/releng](http://www.freebsd.org/doc/en_US.ISO8859-1/articles/releng) for FreeBSD

<http://openbsd.org/faq/faq5.html#Flavors>

### 1.1.5.2 Installer

<http://www.bsdiinstaller.org> for DragonFly, `sysinstall(8)` for FreeBSD, `sysinst` on NetBSD install media, and `INSTALL.[arch]` on OpenBSD install media

<http://openbsd.org/faq/faq4.html#MkInsMedia>

## 1.2 Recognize which commands are available for upgrading the operating system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva [racex@makeworld.com](mailto:racex@makeworld.com) FreeBSD/OpenBSD

---

### 1.2.1 Concept

Recognize the utilities which are used to keep the operating system up-to-date. Some utilities are common to the BSDs, some are specific to certain BSD operating systems and some are third-party applications.

### 1.2.2 Introduction

Binary vs. from-source **NetBSD**

XXX **FreeBSD**

As of FreeBSD 6.2, FreeBSD offers methods for upgrading both its kernel and userland either through binary upgrades or by compiling directly from source code. As is often the case with Open Source operating systems, upgrading FreeBSD using binary packages is much easier and less time consuming than compiling from source code. However, what binary packages don't offer you is the ability to customize your kernel or modify the source code using third-party or in-house patches. As a result, upgrading from binary packages only permits you to apply a GENERIC kernel to your system, allowing no room for modifications. By compiling from source code, you are able to make changes to your kernel configuration file, adding or removing

## 1.2. RECOGNIZE WHICH COMMANDS ARE AVAILABLE FOR UPGRADING THE OPERATING SYSTEM

---

additional features and hardware modules that you may or may not need for your system. With the source code, you are able to make any changes you wish and then compile them into your system.

FreeBSD 6.2 introduces a new utility into its base system for upgrading via binary updates, appropriately called `freebsd-update(8)`. `freebsd-update` has a very simple and straightforward argument set meant to make keeping FreeBSD systems updated with minimal fuss. Using this utility, you are able to download compressed binary images of both the kernel and/or userland, and install them when ready - all without interrupting your system. Then, when ready, a simple reboot is all that is required to load your newly installed kernel. `freebsd-update` also comes with a “rollback” feature in the event that your system doesn’t take well to your newly installed binaries, in which case you can easily revert back to your previous kernel and userland, again, with minimal fuss.

Prior to FreeBSD 6.2, the only way for administrators to upgrade their systems (outside of a complete reinstall) was to compile everything from source code. Even with the introduction of `freebsd-update`, compiling the system from source is still the preferred method for many system administrators. To compile from source, you must first download the source code. Traditionally, this is done using a utility called `CVSup`, which can be found in the ports collection or added as binary package using `pkg_add(1)`. Using `CVSup` and a simple text file or command line arguments specifying, among other parameters, the CVS server to retrieve the source code from, directory to store the files in and a release tag (e.g. `'RELENG_6_2_0'` for FreeBSD 6.2-RELEASE), the entire FreeBSD source tree can be downloaded at whim.

In current version of FreeBSD(7.0) developers add new utility called `CSup`. It’s use `CVSup` updating file, but you don’t need to compile it (instead this for compile `CVSup` you must download several other tools) it’s in base. **OpenBSD**

The safest and easiest way to upgrade an OpenBSD machine is to boot from install media, and follow the upgrade steps, this process is similar to the install process. This can be achieved quickly on a running OpenBSD system by copying the upgrade version of `bsd.rd` kernel image to the `/` directory of the system, then rebooting the sys-

tem, and typing `boot bsd.rd` at the `boot>` prompt, and then choosing the Upgrade script.

The upgrade process is detailed in the FAQ at <http://www.openbsd.org/faq/upgra>

The system can be built from source as described at <http://www.openbsd.org/faq> but this is for following the stable branch. Upgrading via source is NOT supported. **DragonFly BSD**

XXX

### 1.2.3 Examples

### 1.2.4 Practice Exercises

### 1.2.5 More information

`make(1)` including the `'buildworld'`, `'installworld'`, and `'quickworld'` and similar targets; `mergemaster(8)`; `cvs(1)` and the third-party utilities `cvsup` and `cvsync`; `build.sh`, `etcupdate(8)`, `postinstall(8)` and `afterboot(8)`; `src/UPDATING` and `src/BUILDING`.

## 1.3 Understand the difference between a pre-compiled binary and compiling from source

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva [racerx@makeworld.com](mailto:racerx@makeworld.com) FreeBSD/OpenBSD

---

### 1.3.1 Concept

Be familiar with the default location of both the ports collection and the `pkgsrc` collection and which BSDs use which type of collection. Also be able to recognize the extension used by packages. In addition, be aware of the advantages and disadvantages of installing a pre-compiled binary and the advantages and disadvantages of compiling a binary from source.

## 1.4. UNDERSTAND WHEN IT IS PREFERABLE TO INSTALL A PRE-COMPILED BINARY AND HOW TO DO SO

---

### 1.3.2 Introduction

The BSD operating systems provide software build systems for installing third-party add-on software from source code.

### 1.3.3 Examples

### 1.3.4 Practice Exercises

### 1.3.5 More information

Dragonfly and NetBSD provide `pkgsrc/pkgtools/pkgchk`, `pkgsrc/pkgtools/pkgcomp`, `make update` and `make replace`; `portupgrade`, `portsnap` and `cv-sup` are available as third-party utilities

## 1.4 Understand when it is preferable to install a pre-compiled binary and how to do so

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva [racerx@makeworld.com](mailto:racerx@makeworld.com) FreeBSD

---

### 1.4.1 Concept

Be aware that while pre-compiled binaries are quick and easy to install, they don't allow the customization of the binary to a system's particular needs. Know how to install a pre-compiled binary from either a local or a remote source, as well as how to uninstall a pre-compiled binary.

### 1.4.2 Introduction

### 1.4.3 Examples

### 1.4.4 Practice Exercises

### 1.4.5 More information

`pkg_add(1)`, `pkg_delete(1)`

## 1.5 Recognize the available methods for compiling a customized binary

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD/OpenBSD

---

### 1.5.1 Concept

Many applications used by servers support `make(1)` options to compile a binary with the feature set required by a particular installation. While the BSDs all use `make(1)`, the admin should recognize that each BSD uses different mechanisms to use and preserve `make(1)` options.

### 1.5.2 Introduction

### 1.5.3 Examples

### 1.5.4 Practice Exercises

### 1.5.5 More information

DragonFly: `mk.conf(5)` or `make.conf(5)`, `PKG_OPTIONS`, `CFLAGS`

FreeBSD: `-DWITH_*` or `WITH_*=`, `pkgtools.conf(5)`, `make.conf(5)`

NetBSD: `PKG_OPTIONS.`, `CFLAGS`, `mk.conf(5)`, `PKG_DEFAULT_OPTIONS`

OpenBSD: `bsd.port.mk(5)`

## 1.6 Determine what software is installed on a system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD/OpenBSD

---

### 1.6.1 Concept

Recognize that on BSD systems, software and dependencies are tracked by a package manager if the software was installed using packages, ports or pkgsrc. Be familiar with querying the package manager to determine what software and their versions are installed on the system.

### 1.6.2 Introduction

### 1.6.3 Examples

### 1.6.4 Practice Exercises

### 1.6.5 More information

pkg\_info(1)

## 1.7 Determine which software requires upgrading

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

---

### 1.7.1 Concept

Recognize the importance of balancing the need to keep software up-to-date while minimizing the impact on a production system. Dragonfly and NetBSD use pkgsrc which provides utilities for determining which installed software is out-of-date. FreeBSD provides pkg\_version and third-party utilities are also available which integrate with the BSD package managers.

### 1.7.2 Introduction

### 1.7.3 Examples

### 1.7.4 Practice Exercises

### 1.7.5 More information

pkgsrc/pkgtool/pkg\_chk and make show-downlevel for Dragonfly and NetBSD; pkg\_version(1), and the third-party portupgrade

## 1.8 Upgrade installed software

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

---

### 1.8.1 Concept

Recognize the built-in and third-party commands which are available for upgrading installed software on BSD systems. In addition, be able to recognize which BSD systems use pkgsrc.

### 1.8.2 Introduction

### 1.8.3 Examples

### 1.8.4 Practice Exercises

### 1.8.5 More information

Dragonfly and NetBSD provide pkgsrc/pkgtools/pkg\_chk, pkgsrc/pkgtools/pkg\_c make update and make replace; portupgrade, portsnap and cvsup are available as third-party utilities

## 1.9 Determine which software have outstanding security advisories

Author: *name contact BSD flavour*

## 1.10. FOLLOW THE INSTRUCTIONS IN A SECURITY ADVISORY TO APPLY A SECURITY PATCH

---

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD/OpenBSD

---

### 1.9.1 Concept

Recognize the importance of being aware of software security vulnerabilities. Also recognize the third-party utilities which integrate with the BSD package managers to determine which software has outstanding vulnerabilities.

### 1.9.2 Introduction

### 1.9.3 Examples

### 1.9.4 Practice Exercises

### 1.9.5 More information

audit-packages for Dragonfly and NetBSD; portaudit and vuxml for FreeBSD and OpenBSD

## 1.10 Follow the instructions in a security advisory to apply a security patch

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

---

### 1.10.1 Concept

Be aware that each BSD project maintains security advisories which are available both on the Internet and via mailing lists. Be able to follow the instructions in an advisory when asked to do so by a supervisor.

**1.10.2 Introduction**

**1.10.3 Examples**

**1.10.4 Practice Exercises**

**1.10.5 More information**

patch(1), make(1), and fetch(1), ftp(1) and build.sh

## 2 Chapter Securing the Operating System

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

The mark of a good system administrator is the awareness of and adherence to best security practices. An administrator is expected to be familiar with common security practices. BSD systems are designed with security in mind and provide many mechanisms which allow the system administrator to tune systems to the security requirements of an organization. While the BSDA candidate won't always be responsible for implementing these mechanisms, being able to recognize the features and commands available for securing BSD systems is still an essential aspect of overall security administration.

- Determine the system's security level
- Configure an SSH server according to a set of requirements
- Configure an SSH server to use a key pair for authentication
- Preserve existing SSH host keys during a system upgrade
- Recognize alternate authentication mechanisms
- Recognize alternate authorization schemes
- Recognize basic recommended access methods
- Recognize BSD firewalls and rulesets
- Recognize BSD mechanisms for encrypting devices

- Recognize methods for verifying the validity of binaries
- Recognize the BSD methods for restraining a service
- Change the encryption algorithm used to encrypt the password database
- Modify the system banner
- Protect authentication data

## 2.1 Determine the system's security level

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.1.1 Concept

BSD systems provide security profiles known as securelevels. Be able to recognize the restrictions set by each securelevel for each BSD operating system. Also understand under what circumstances a securelevel can be raised or lowered.

### 2.1.2 Introduction

The BSD kernels can limit – even from the superuser – changing immutable and append-only file flags, \*\*

In addition on NetBSD, the verified exec in-kernel fingerprint table can't be modified.

File flags are covered in View and modify file flags.

In FreeBSD you can look at current secure level via sysctl.

```
# sysctl kern.securelevel
kern.securelevel: -1
```

You can change this virable on the fly. This parts covered in Modify a kernel parameter on the fly.

## 2.2. CONFIGURE AN SSH SERVER ACCORDING TO A SET OF REQUIREMENTS

---

### 2.1.3 Examples

### 2.1.4 Practice Exercises

### 2.1.5 More information

init(8), sysctl(8), rc.conf(5)

## 2.2 Configure an SSH server according to a set of requirements

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.2.1 Concept

Be aware that the sshd(8) built into BSD systems can be configured to limit who can access a system via SSH.

### 2.2.2 Introduction

### 2.2.3 Examples

### 2.2.4 Practice Exercises

### 2.2.5 More information

sshd\_config(5)

## 2.3 Configure an SSH server to use a key pair for authentication

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.3.1 Concept

Understand private/public key theory including: which protocols are available for generating key pairs, choosing an appropriate bit size, providing a seed, providing a passphrase, and verifying a fingerprint. In addition, able to generate their own keys and use them for authentication.

### 2.3.2 Introduction

### 2.3.3 Examples

### 2.3.4 Practice Exercises

### 2.3.5 More information

ssh-keygen(1) including these keywords: `authorized_keys`, `id_rsa`, and `id_rsa.pub`

## 2.4 Preserve existing SSH host keys during a system upgrade

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.4.1 Concept

In addition to knowing how to generate a system's SSH keys, know where host keys are located and how to preserve them if the system is upgraded or replaced.

### 2.4.2 Introduction

### 2.4.3 Examples

### 2.4.4 Practice Exercises

### 2.4.5 More information

*/etc/ssh/ssh\_host\_key*

## 2.5 Recognize alternate authentication mechanisms

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.5.1 Concept

Understand basic authentication theory and be aware that providing a username and password is only one way to authenticate on BSD systems. Have a basic understand of PAM and know it is available on Dragonfly, FreeBSD and NetBSD 3.x. Also understand basic theory regarding Kerberos, OTP and RADIUS. (Note: The BSDA candidate is not expected to know how to configure an alternate authentication mechanism.)

### 2.5.2 Introduction

### 2.5.3 Examples

### 2.5.4 Practice Exercises

### 2.5.5 More information

## 2.6 Recognize alternate authorization schemes

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **2.6.1 Concept**

Admins should understand basic authorization theory and how MAC and ACLs extend the features provided by the standard Unix permissions.

### **2.6.2 Introduction**

### **2.6.3 Examples**

### **2.6.4 Practice Exercises**

### **2.6.5 More information**

mac(4) and acl(3) on FreeBSD; systrace(1) on NetBSD and OpenBSD

## **2.7 Recognize basic recommended access methods**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **2.7.1 Concept**

Be familiar with standard system administration practices used to minimize the risks associated with accessing a system. These include using ssh(1) instead of telnet(1), denying root logins, using the possibly third-party sudo utility instead of su(1) and minimizing the use of the wheel group.

## 2.7.2 Introduction

## 2.7.3 Examples

## 2.7.4 Practice Exercises

## 2.7.5 More information

ttys(5), sshd\_config(5), ftpusers(5); the possibly third-party utility sudo which includes visudo, suedit and sudoers

# 2.8 Recognize BSD firewalls and rulesets

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

## 2.8.1 Concept

Each BSD comes with at least one built-in firewall. Recognize which firewalls are available on each BSD and which commands are used to view each firewall's ruleset.

## 2.8.2 Introduction

## 2.8.3 Examples

## 2.8.4 Practice Exercises

## 2.8.5 More information

ipfw(8), ipf(8), ipfstat(8), pf(4), pfctl(8) and firewall(7)

# 2.9 Recognize BSD mechanisms for encrypting devices

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **2.9.1 Concept**

Be aware that it is possible to encrypt devices on BSD systems and which utilities are available on each BSD system.

### **2.9.2 Introduction**

### **2.9.3 Examples**

### **2.9.4 Practice Exercises**

### **2.9.5 More information**

gbde(4) and gbde(8) on FreeBSD; cgd(4) on NetBSD; vnd(4) on OpenBSD

## **2.10 Recognize methods for verifying the validity of binaries**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **2.10.1 Concept**

Recognize the utility of file integrity utilities such as tripwire. Recognize the built-in checks provided on some of the BSDs.

### 2.10.2 Introduction

### 2.10.3 Examples

### 2.10.4 Practice Exercises

### 2.10.5 More information

security(7) or (8); security.conf(5); veriexecctl(8)

## 2.11 Recognize the BSD methods for restraining a service

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.11.1 Concept

Recognize the advantages of restraining a service on an Internet facing system and which utilities are available to do so on each of the BSDs.

### 2.11.2 Introduction

### 2.11.3 Examples

### 2.11.4 Practice Exercises

### 2.11.5 More information

chroot(8); jail(8); systrace(1); the third-party Xen application

## 2.12 Change the encryption algorithm used to encrypt the password database

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

### 2.12.1 Concept

Given a screenshot of a password database, an admin should be able to recognize the encryption algorithm in use and how to select another algorithm. Have a basic understanding of when to use DES, MD5 and Blowfish.

### 2.12.2 Introduction

### 2.12.3 Examples

### 2.12.4 Practice Exercises

### 2.12.5 More information

login.conf(5); auth.conf(5); passwd.conf(5); adduser.conf(5) and adduser(8)

## 2.13 Modify the system banner

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 2.13.1 Concept

Be aware of the banner(s) that may be seen depending on how a user accesses a system and which files are used to configure each banner.

### 2.13.2 Introduction

Various banners and welcome messages are available to introduce a BSD system and to possibly share news, system policies, or important announcements. The most common message is the `/etc/motd` file. For normal local or remote logins, this plain text file is displayed. While it is called the “message of the day,” this message is not always

updated every day and is only displayed on logins, so may not be read everyday. The administrator for the system modifies this file.

TODO: NetBSD's `login.conf` allows defining a “welcome” capability to override this when logging in via `sshd` or `login(8)`.

The `gettytab` defines an initial banner message (`im`) displayed before the console login prompt. It defaults to:

```
\r\n%s/%m (%h) (%t)\r\n\r\n
```

The format is described in the `gettytab(5)` manual page.

- `\r\n` carriage return and line feed
- `%s` name of operating system
- `%m` type of machine, such as `TODO`
- `%h` the hostname
- `%t` the tty name, such as `TODO`

The SSH server can be configured to send a banner message before the authentication. And it also can be configured to disable displaying the “message of the day”. `TODO`

TODO: `telnetd` uses standard login??

### 2.13.3 Examples

### 2.13.4 Practice Exercises

1. View your `/etc/motd` file.

### 2.13.5 More information

`motd(5)`, `login.conf(5)`, `gettytab(5)`, `sshd_config(5)`

## 2.14 Protect authentication data

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **2.14.1 Concept**

To prevent attacks against system security with password cracking attacks, BSD systems keep encrypted passwords visible to system processes only. An admin should have an understanding of the location of the password database files and their proper permission sets.

### **2.14.2 Introduction**

### **2.14.3 Examples**

### **2.14.4 Practice Exercises**

### **2.14.5 More information**

`passwd(5)`, `pwd_mkdb(8)`

# 3 Chapter Files, Filesystems and Disks

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

The usefulness of any computing system is related to the accessibility of the data stored on it. An admin is expected to thoroughly understand how to make data available both locally and remotely and how to use permissions to ensure authorized users can access that data. Be experienced in backing up data and in resolving common disk issues.

- Mount or unmount local filesystems
- Configure data to be available through NFS
- Determine which filesystems are currently mounted and which will be mounted at system boot
- Determine disk capacity and which files are consuming the most disk space
- Create and view symbolic or hard links
- View and modify ACLs
- View file permissions and modify them using either symbolic or octal mode
- Modify a file's owner or group
- Backup and restore a specified set of files and directories to local disk or tape

- Backup and restore a file system
- Determine the directory structure of a system
- Manually run the file system checker and repair tool
- View and modify file flags
- Monitor the virtual memory system

### **3.1 Mount or unmount local filesystems**

Author: AndreasKuehl andreas dot kuehl at clicktivities dot net  
FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

#### **3.1.1 Concept**

Be familiar with all aspects of mounting and unmounting local filesystems including: how to mount/umount a specified filesystem, how to mount all filesystems, configuring filesystems to be mounted at boot, passing options to mount(1), and resolving mount(1) errors.

#### **3.1.2 Introduction**

During system boot the file systems from disk or nfs or other network protocols are mounted, are available during the operation time and are unmounted at shutdown time. In normal operation, no one cares about file systems, mountpoints and other stuff. They are just there. It's your job to handle all the other times :-)

What we have to cover:

1. mount
2. unmount
3. /dev/ad0s1a

## 3.2. CONFIGURE DATA TO BE AVAILABLE THROUGH NFS

4. hint to fsck
5. /etc/exports
6. mount -a
7. errors at mount -a
8. /etc/fstab ...

### **3.1.3 Examples**

### **3.1.4 Practice Exercises**

### **3.1.5 More information**

mount(8), umount(8), fstab(5)

## **3.2 Configure data to be available through NFS**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **3.2.1 Concept**

Be aware of the utilities associated with NFS and the security risks associated with allowing RPC through a firewall. In addition, be able to configure a NFS server or client according to a set of requirements on the data to be made available.

### **3.2.2 Introduction**

### **3.2.3 Examples**

### **3.2.4 Practice Exercises**

### **3.2.5 More information**

exports(5), nfsd(8), mountd(8), rpcbind(8) or portmap(8), rpc.lockd(8), rpc.statd(8), rc.conf(5) and mount\_nfs(8)

### **3.3 Determine which filesystems are currently mounted and which will be mounted at system boot**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

#### **3.3.1 Concept**

Be able to determine which filesystems are currently mounted and which will be mounted at boot time.

#### **3.3.2 Introduction**

#### **3.3.3 Examples**

#### **3.3.4 Practice Exercises**

#### **3.3.5 More information**

mount(1), du(1), fstab(5)

### **3.4 Determine disk capacity and which files are consuming the most disk space**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

#### **3.4.1 Concept**

Be able to combine common Unix command line utilities to quickly determine which files are consuming the most disk space.

### 3.4.2 Introduction

### 3.4.3 Examples

### 3.4.4 Practice Exercises

### 3.4.5 More information

du(1), df(1), find(1), sort(1), systat(1)

## 3.5 Create and view symbolic or hard links

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 3.5.1 Concept

Know the difference between symbolic and hard links as well as how to create, view and remove both types of links. In addition, be able to temporarily resolve a low disk space issue using a symbolic link.

### 3.5.2 Introduction

### 3.5.3 Examples

### 3.5.4 Practice Exercises

### 3.5.5 More information

ln(1), ls(1), rm(1), stat(1)

## 3.6 View and modify ACLs

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **3.6.1 Concept**

Be able to determine if a FreeBSD system is using ACLs, and if so, on which filesystems. In addition, be able to view and modify a file's ACL on a FreeBSD system.

### **3.6.2 Introduction**

### **3.6.3 Examples**

### **3.6.4 Practice Exercises**

### **3.6.5 More information**

mount(8), ls(1), getfacl(1)

## **3.7 View file permissions and modify them using either symbolic or octal mode**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **3.7.1 Concept**

An admin is expected to have a thorough understanding of traditional Unix permissions including: how to view and modify permissions, why the sticky bit is important on /tmp and other shared directories, recognizing and using the SUID and SGID bits, and the difference between symbolic and octal mode. In addition, understand that a shell setting determines the default file and directory permissions and, given a umask value, be able to determine the default permission set.

### 3.7.2 Introduction

### 3.7.3 Examples

### 3.7.4 Practice Exercises

### 3.7.5 More information

ls(1), chmod(1), umask(1) or umask(2)

## 3.8 Modify a file's owner or group

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 3.8.1 Concept

Be able to modify a file's ownership as required. In addition, be aware of the importance of verifying one's own identity before creating files.

### 3.8.2 Introduction

### 3.8.3 Examples

### 3.8.4 Practice Exercises

### 3.8.5 More information

chown(8), chgrp(1); su(1), mtree(8)

## 3.9 Backup and restore a specified set of files and directories to local disk or tape

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **3.9.1 Concept**

Admins should have experience using common Unix command line backup utilities. In addition, be able to recognize the device names for tape devices on BSD systems.

### **3.9.2 Introduction**

### **3.9.3 Examples**

### **3.9.4 Practice Exercises**

### **3.9.5 More information**

tar(1), cpio(1), pax(1), cp(1), cpdup(1)

## **3.10 Backup and restore a file system**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **3.10.1 Concept**

Recognize the utilities used to backup an entire filesystem and the various dump(1) levels.

### **3.10.2 Introduction**

### **3.10.3 Examples**

### **3.10.4 Practice Exercises**

### **3.10.5 More information**

dump(8), restore(8), dd(1)

## 3.11 Determine the directory structure of a system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 3.11.1 Concept

Be able to quickly determine the directory layout used by BSD systems.

### 3.11.2 Introduction

### 3.11.3 Examples

### 3.11.4 Practice Exercises

### 3.11.5 More information

hier(7)

## 3.12 Manually run the file system checker and repair tool

Author: AndreasKuehl andreas dot kuehl at clicktivities dot net  
FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 3.12.1 Concept

Be aware of the utilities available to check the consistency of a file system and to use them under supervision.

### 3.12.2 Introduction

Under certain circumstances, the ffs/BSD file system can get corrupt or broken. It may be better to say: The metainformation is corrupt/damaged. As a result of this, places where data live could not be found or space is marked empty but old data is overwritten, when new data is written to the filesystem.

To prevent this, a file system is marked as unclean by certain mechanism in the operating system and can not be mounted. During the booting process, unclean filesystems are checked to rebuild the metainformation. Newer FreeBSDs (**What about the other BSDs?**) can mount a file system and do a check in the background after the booting process.

Sometimes, the automatic check breaks and the system stops in the booting process. (**Why?**) (**What is single user mode?**) Sometimes it is necessary to check a filesystem as you attach a foreign disk by firewire or usb or scsi or something else.

The command for this operation is fsck. You can name the filesystem you want to check by the devicename i.e. /dev/ad0s3h or, if the filesystem is in the /etc/fstab by the mountpoint.

During the check, fsck will ask you questions about what to do with data, that was found in the filesystem without being accounted in the metainformation. It is safe to answer with "y". (**Really?**) Recovered data will appear in a directory called lost+found at the base of the filesystem. This could be examined to find lost data. Most times, and with Soft updates switched on, almost always, you will find (parts of) already deleted files. (**Really?**)

### 3.12.3 Examples

```
fsck /dev/ad0s1a
```

will check first ide disk, partition 1, slice 1

```
fsck /usr
```

will check the filesystem, that is normally mounted at /usr

### 3.12.4 Practice Exercises

### 3.12.5 More information

fscck(8)

## 3.13 View and modify file flags

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 3.13.1 Concept

Understand how file flags augment traditional Unix permissions and should recognize how to view and modify the immutable, append-only and undelete flags.

### 3.13.2 Introduction

Secure levels are covered in Determine the system's security level.

### 3.13.3 Examples

### 3.13.4 Practice Exercises

### 3.13.5 More information

ls(1), chflags(1)

## 3.14 Monitor the virtual memory system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **3.14.1 Concept**

The virtual memory subsystem may have an important impact on a system's overall performance. Be able to configure a swap device and review swap usage.

### **3.14.2 Introduction**

### **3.14.3 Examples**

### **3.14.4 Practice Exercises**

### **3.14.5 More information**

`pstat(8)`; `systat(1)`; `top(1)`; `vmstat(8)`; `swapctl(8)`; `swapinfo(8)`

# 4 Chapter Users and Accounts Management

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

All systems require at least one user account, and depending upon the role of the system, an admin's job duties may include supporting end-users in the maintenance of their accounts. Be able to create user accounts, modify account settings, disable accounts, and reset passwords. Know how to track account activity and determine which accounts are currently accessing a system.

- Create, modify and remove user accounts
- Create a system account
- Lock a user account or reset a locked user account
- Determine identity and group membership
- Determine who is currently on the system or the last time a user was on the system
- Enable accounting and view system usage statistics
- Change a user's default shell
- Control which files are copied to a new user's home directory during account creation
- Change a password

## 4.1 Create, modify and remove user accounts

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 4.1.1 Concept

Managing user accounts is an important aspect of system administration. Be aware that the account management utilities differ across BSD systems and should be comfortable using each utility according to a set of requirements.

### 4.1.2 Introduction

### 4.1.3 Examples

### 4.1.4 Practice Exercises

### 4.1.5 More information

`vipw(8)`; `pw(8)`, `adduser(8)`, `adduser.conf(5)`, `useradd(8)`, `userdel(8)`, `rmuser(8)`, `userinfo(8)`, `usermod(8)`, and `user(8)`

## 4.2 Create a system account

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 4.2.1 Concept

Understand that many services require an account and that such accounts should not be available for logins.

#### 4.2.2 Introduction

#### 4.2.3 Examples

#### 4.2.4 Practice Exercises

#### 4.2.5 More information

nologin(8); using a \* in the password field of passwd(5)

### 4.3 Lock a user account or reset a locked user account

Author: *name contact BSD flavour*

Reviewer: Jeremy C. Reed reed AT reedmedia DOT net FreeBSD/NetBSD/DragonFly

Reviewer: *name contact BSD flavour*

---

#### 4.3.1 Concept

Know how to recognize a locked account and how to remove the lock.

#### 4.3.2 Introduction

#### 4.3.3 Examples

#### 4.3.4 Practice Exercises

#### 4.3.5 More information

vipw(8); chpass(1), chfn(1), chsh(1), pw(8), user(8)

### 4.4 Determine identity and group membership

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 4.4.1 Concept

In the context of the Unix permission system, determining one's identity and group membership is essential to determine what authorizations are available. Be able to determine, and as required, change identity or group membership.

### 4.4.2 Introduction

The user's privileges determine what kind of access (if any) to given files and directories a user have. Groups are a mean to simplify user management.

### 4.4.3 Examples

We can determine our identity – that is our username and groups to which we belong – using **id** , **groups** and **whoami** commands.

Our username can be determined by simply executing **whoami** command without any parameters.

```
$ \textbf{whoami}
```

```
user
```

In the above example we're logged into the system as a *user* . The **whoami** command is equivalent to **id -un** .

The **groups** command let us check to which groups we're currently begin assigned to. It can also be used to check other existing user's group membership. Executing **groups** without a username will display information on us.

```
$ \textbf{groups}
```

```
users audio mail cvs
```

```
$ \textbf{groups john}
```

```
users mail
```

```
$ \textbf{groups mike}
```

```
groups: mike: no such user
```

#### 4.4. DETERMINE IDENTITY AND GROUP MEMBERSHIP

---

The **groups** command is equivalent to **id -Gn** .

The **id** command may take few arguments and can output many informations on given user. In most basic usage it displays our user ID (uid), our basic group id (gid) and groups to which we belong to.

```
$ \textbf{id}
```

```
uid=1001(user) gid=100(users) groups=100(users), 92(audio), 1003(mai
```

It can also be used to display the very same information on other user.

```
$ \textbf{id john}
```

```
uid=1002(john) gid=100(users) groups=100(users), 1003(mail)
```

Note, that the above mentioned commands will not display our new groups membership until we'll logout and login again.

As explained above, some commands let us peek into other user's identity information, which might be useful to system administrators for checking other logged in users. To see who is currently logged in execute **who** command:

```
$ \textbf{who}
```

```
root          ttyv1      Jan  4 23:16
user          tty0      Jan  5 22:19 (192.168.86.11)
```

This command outputs some more information on all logged users: username, tty name, date and time of login and remote host's IP address if it is not local. It can also display the very same information only about us:

```
$ \textbf{who am I}
```

```
user          tty0      Jan  5 22:19 (192.168.86.11)
```

Finally, having determined who we are – our username and groups membership – we may sometimes need to switch to more privileged account (most commonly *root* ) without completely logging out current user. To do so, we'll use the **su** command.

The **su** command may be given with or without a username. Given without a username **su** switches to superuser *root*. Password is not echoed in any form (not even with \* marks).

```
$ \textbf{whoami}
```

```
user
```

```
$ \textbf{su}
```

```
Password:
```

```
# \textbf{whoami}
```

```
root
```

Most commonly, when switching to normal user account, we'd like to simulate a full login. This is done with the **-** parameter:

```
$ \textbf{whoami}
```

```
user
```

```
$ \textbf{echo $HOME}
```

```
/home/user
```

```
$ \textbf{su - john}
```

```
Password:
```

```
$ \textbf{whoami}
```

```
john
```

```
$ \textbf{echo $HOME}
```

```
/home/john
```

#### 4.4.4 Practice Exercises

1. Compare the output of **whoami** and **id -un** commands.
2. Compare the output of **groups** and **id -Gn** commands.

#### 4.5. DETERMINE WHO IS CURRENTLY ON THE SYSTEM OR THE LAST TIME A USER WAS ON THE SYSTEM

---

3. Try executing **id** with a variation of all parameters described in `id(1)` system manual.
4. Try checking information on both existing and not existing users.
5. Try executing **who** with arguments: **-H** , **-q** , **-m** , and **-u** .
6. Check the result of **su** command with parameters: **-** , **-l** , and **-m** .

#### 4.4.5 More information

`id(1)`, `groups(1)`, `who(1)`, `whoami(1)`, `su(1)`

### 4.5 Determine who is currently on the system or the last time a user was on the system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

#### 4.5.1 Concept

BSD systems maintain databases which can be queried for details regarding logins. Be familiar with the database names and the utilities available for determining login information.

#### 4.5.2 Introduction

#### 4.5.3 Examples

#### 4.5.4 Practice Exercises

#### 4.5.5 More information

`wtmp(5)`, `utmp(5)`, `w(1)`, `who(1)`, `users(1)`, `last(1)`, `lastlogin(8)`, `lastlog(5)`, `finger(1)`

## 4.6 Enable accounting and view system usage statistics

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 4.6.1 Concept

Be aware of when it is appropriate to enable system accounting, recognize which utilities are available to do so, and know how to view the resulting statistics.

### 4.6.2 Introduction

The kernel keeps track of various attributes of all processes and, when system accounting is enabled, this information is saved when the process terminates. The accounting information includes the command name, the starting time, the amount of time used by the system and the user (TODO: explain that), the elapsed time, the user ID and group ID, the average amount of memory used, the count of I/O operations, and the terminal (tty) where the process was started. The accounting also records if the process was forked without replacing the parent process (exec) and how the process was terminated (such as with core dump or killed by a signal).

The system accounting is enabled by running the `accton` command with the path to the file to store the data as the argument, commonly at `/var/account/acct`.

System accounting is turned off by running `accton` without any arguments.

TODO: show how is enabled at boot time on all BSDs

TODO: show example of data; show examples of `sa`, `ac`, `accton`, `lastcomm`

TODO: not the same, but also cover “last”

### 4.6.3 Examples

### 4.6.4 Practice Exercises

### 4.6.5 More information

ac(8), sa(8), accton(8), lastcomm(1), last(1)

## 4.7 Change a user's default shell

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 4.7.1 Concept

Know the default shells for both user accounts and the superuser account for each BSD. In addition, know how to change the default shell for each BSD operating system.

### 4.7.2 Introduction

BSD systems historically use the standard C shell (`/bin/csh`) as root's login shell. OpenBSD uses `/bin/ksh` for root's shell.

TODO: should C shell be spelled out? Or called Csh? or csh?

If no shell is set (the 7th field in the `passwd` database is empty), then `login` and some other programs will default to standard `/bin/sh`.

Many system users use `/sbin/nologin` as the default shell. This utility will simply exit after outputting "This account is currently not available." (Note: On FreeBSD, the `nologin(8)` utility is located at `/usr/sbin/nologin`.)

The standard tool for changing the user's login shell is `chsh(1)`. (It is a link to `chpass(1)`.) Running the `chsh` utility will start up your preferred editor where the user can modify the selected shell (and other user database information).

The chsh program is setuid root, so it runs with root's privilege so it can modify the user databases. TODO: should this setuid be noted here?

TODO: should this cover EDITOR or VISUAL here? Or point to which topic page?

TODO: what systems don't default to vi for VISUAL or EDITOR??

The vipw tool can also be used to manually edit the master.passwd database.

TODO: point to topic about master.passwd.

TODO: where is pwd\_mkdb, pwd.db and spwd.db covered? Should it be covered here or point to it.

TODO: show how to use chsh from command line without using editor. Do all BSDs support that?

TODO: show how to use pw (FreeBSD and DragonFly) to set shell and show how to use "usermod" (NetBSD and OpenBSD) for this.

### 4.7.3 Examples

### 4.7.4 Practice Exercises

### 4.7.5 More information

vipw(8); chpass(1), chfn(1), chsh(1), pw(8), user(8)

## 4.8 Control which files are copied to a new user's home directory during account creation

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 4.8.1 Concept

BSD systems use a "skel" directory containing files which are copied over to a user's home directory when a user account is made. Be aware of the location of the skel directory on each BSD, as well as how to override the copying of its contents during account creation.

### 4.8.2 Introduction

### 4.8.3 Examples

### 4.8.4 Practice Exercises

### 4.8.5 More information

pw(8), adduser.conf(5), useradd(8) and usermgmt.conf(5)

## 4.9 Change a password

Author: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name contact BSD flavour*

---

### 4.9.1 Concept

Be able to change own password as well as the passwords of other users as required.

### 4.9.2 Introduction

Sometimes you need to change password of root user or another user in your system(For example you must recover some system but password is lost, for example in FreeBSD you can boot in single mode and change root password).

### 4.9.3 Examples

You can change root password only if you logged as root user, or use **su** for substitute user identity to root. Try to change root password.

```
> id
uid=1001(user) gid=0(wheel) groups=0(wheel)
> su
Password:
# id
```

```
uid=0(root) gid=0(wheel) groups=0(wheel), 5(operator)
# passwd
Changing local password for root
New Password:
Retype New Password:
```

We use `su`, then we type `passwd` without second argument, that means change password for current user - in our example root. But you can use **passwd user** for change **user** password under root.

#### **4.9.4 Practice Exercises**

1. Change root password
2. Change password of another user

#### **4.9.5 More information**

`passwd(1)`, `vipw(8)`

# 5 Chapter Basic System Administration

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

An important component of system administration is an awareness of its subsystems and their interactions, as well as how to monitor the health of a running system. Demonstrate experience in interacting with BSD processes, a running kernel, and the BSD boot process. Demonstrate familiarity with BSD devices, the disk subsystem and the mail and print daemons.

- Determine which process are consuming the most CPU
- View and send signals to active processes
- Use an rc(8) script to determine if a service is running and start, restart or stop it as required
- View and configure system hardware
- View, load, or unload a kernel module
- Modify a kernel parameter on the fly
- View the status of a software RAID mirror or stripe
- Determine which MTA is being used on the system
- Configure system logging
- Review log files to troubleshoot and monitor system behavior
- Understand basic printer troubleshooting

- Create or modify email aliases for Sendmail or Postfix
- Halt, reboot, or bring the system to single-user mode
- Recognize the difference between hard and soft limits and modify existing resource limits
- Recognize the BSD utilities that shape traffic or control bandwidth
- Recognize common, possibly third-party, server configuration files
- Configure a service to start at boot time
- Configure the scripts that run periodically to perform various system maintenance tasks
- View the Sendmail or Postfix mail queue
- Determine the last system boot time and the workload on the system
- Monitor disk input–output
- Deal with busy devices
- Determine information regarding the operating system
- Understand the advantages of using a BSD license

## **5.1 Determine which process are consuming the most CPU**

Author: *hubertf contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.1.1 Concept

Be able to view active processes and recognize inordinate CPU usage. In addition, know how to end a process or change its priority.

### 5.1.2 Introduction

There are several programs that allow showing CPU utilization on a Unix system. Some of them can be found on every kind of system, some are specific to others. Here's a list:

- **ps(1)** : The command is available on all Unix systems, but the evil thing is that the set of options differs between systems. The good news is that all BSD systems use the same set of options, and by running "ps -aux" the list is sorted to have the process using the most CPU time on the top.
- **top(1)** : This interactive command is not available on all Unix systems, but it's part of every BSD system. Running it will display some system statistics on the top, and then provide a list of processes that's sorted by CPU utilization by default. The display is updated every few seconds, so any process that starts hogging the CPU can be determine easily.
- **systat(1)** : This program can only be found on BSD systems. It can display a wide range of system statistics, and the default is to display processes and their CPU utilization. Unfortunately no process ID is shown, so if a certain process misbehaves some other method needs to be used to precisely determine the guilty party.

Now that we know how to determine general process stats, managing them should be discussed. For that, processes need to be identified, which is done via a process ID (PID) that is unique for each running process on a system. The above programs can be used to determine the PID of a running process.

Operations that can be done on processes include:

- **change priority**: this is usually done using the renice(1) program or shell builtin. The priority there is given as "niceness"

level, which goes from -20 (not nice at all == high priority) to 20 (very nice == low priority). Nice levels below (= priorities higher than) 0 are reserved for the superuser, and a process that had its nice level increased (= priority decreased) cannot undo this change later.

- **start with different priority:** If a process is known to need less or more CPU time than is assigned by default, it can be started with a different nice level. This is done using the nice(1) command.
- **abort the process:** there are several commands that can be told to a process, by sending it a certain “signal”. The command to send signals to a process is kill(1), a list of possible signal names can be printed with “kill -l”. See the signal(7) manpage for a description of the signals and their default handlers.

Please note that a process can ignore most signals, or install a new handler to do whatever it likes to do with a signal. There’s only one signal that can’t be ignored, and which also doesn’t give a process the chance to clean up after itself, SIGKILL (9). Be sure to only use this as last resort, as unpleasant side-effects may happen! In most cases, the default of SIGTERM (15) is sufficient to end a process.

### 5.1.3 Examples

Determine the process that takes most CPU:

```
% ps -aux | head -3
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT STARTED     TIME
feyrer    5924 50.0   5.9 104588 30900 tttyp6 R+   12:17AM   0:03.31
root      25528 13.7   0.2   468   1104 tttyp4 R+   12:17AM   0:00.85
```

Now that we know qemu hogs the CPU, nice it down a bit. Running ‘renice’ without arguments will show its usage:

```
% renice
Usage: renice [<priority> | -n <incr>] [[-p] <pids>...] [-g <pgid>]
```

## 5.1. DETERMINE WHICH PROCESS ARE CONSUMING THE MOST CPU

---

Let's say we want to change the nice level of process 5924 (=qemu) from the default of 0 to 10:

```
% renice 10 -p 5924
5924: old priority 0, new priority 10
```

Upon observation we will still see that the process takes the most CPU:

```
% ps -aux | head -3
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT STARTED      TIME COMMAND
feyrer    5924 27.1 11.6 104704 60636 tttyp6 RN+   12:17AM    0:31.38 qemu
feyrer    1206  1.9 10.8  73896 56304 ?        Ra    11:48AM    75:11.33 /usr
```

This is because no other process claims the CPU. If another process (e.g. your window system for interaction, or a compile job) would claim the CPU, the qemu process would have the CPU taken until no other job needs it.

If the command still uses too much CPU and you are very certain that there is no other way to end it (e.g. by properly ending it, in the case of Qemu by shutting down the system being emulated), it can be killed using the kill(1) command:

```
% kill 5924
```

If, for some reason, a process catches the default signal (SIGTERM, 15) a different signal number can be given to the kill(1) command either by name or by signal number that is known to terminate the process unconditionally - be very careful with this:

```
% kill -9 5924
% kill -KILL 5924
```

Both of the preceding commands have the same effect.

### 5.1.4 Practice Exercises

- Determine a list of processes running on your systems using top(1), ps(1) and systat(1).
- Determine which process consumes the most CPU time

- Make sure the process is not critical to the system's operation, and lower its priority by increasing its nice-level
- Try to increase the process' priority again, i.e. lower the nice-level, and see it fail during this operation.
- Send the process the SIGTERM signal.
- Possibly restart the process.

### **5.1.5 More information**

top(1), systat(1), ps(1), nice(1), renice(1), kill(1), signal(7)

## **5.2 View and send signals to active processes**

Author: hubertf *contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.2.1 Concept**

Be familiar with both the names and numbers of the most commonly used Unix signals and how to send a signal to an active process. Recognize the difference between a SIGTERM and a SIGKILL.

### **5.2.2 Introduction**

Determine which process are consuming the most CPU talks about processes, and how to list and manage them. This topic is covered in a bit more depth here, by listing other tools besides kill(1):

- **pgrep(1)**: Many times you will find yourself wanting to look for a certain process, using a pipeline of “ps ... | grep ...”. The pgrep(1) command automates this - you give it a program name, and it will print the process ID of the process(es) that match the command name. This command is available on all BSD systems.

### 5.3. USE AN RC.D SCRIPT TO DETERMINE IF A SERVICE IS RUNNING AND START, RESTART OR STOP IT AS REQUIRED

- **pkill(1)**: Like `pgrep(1)` this command looks through the list of processes running on a system, and sends a certain signal to all processes matching a given name.
- **killall(1)**: This command performs the same operation as `pgrep(1)`. It is only available on FreeBSD, its existence predates that of `pgrep(1)`.

#### 5.2.3 Examples

See Determine which process are consuming the most CPU for examples on using `ps(1)` and `kill(1)`. The following example achieves the same goal with the commands introduced here:

```
# pgrep -lf named
338 /usr/sbin/named -u bind
# pgrep named
338
# kill named
# pgrep named
#
```

#### 5.2.4 Practice Exercises

See Determine which process are consuming the most CPU and perform the same tasks with `pgrep(1)` and `pkill(1)`.

#### 5.2.5 More information

`ps(1)`; `kill(1)`; `killall(1)`; `pkill(1)`; `pgrep(1)`

### 5.3 Use an rc.d script to determine if a service is running and start, restart or stop it as required

Author: *hubertf contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.3.1 Concept

In addition to directly sending signals to processes, realize that BSD systems provide scripts which can be used to check the status of services and to stop, start and restart them as required. Be aware of the locations of these scripts on each of the BSD systems. Note: this objective does not apply to OpenBSD.

### 5.3.2 Introduction

NetBSD and FreeBSD have broken up the traditional system startup script `/etc/rc` into tiny scripts that start and stop single services, like System V release 4 Unix systems have done for some time. Each script is ran at system boot time, and it determines via variables set in `/etc/rc.conf` if the service it provides should be started or not. Similar operation is performed at system shutdown time.

The advantage this approach has to the system administrator that he doesn't need to know any details about how to start or stop a system - running the corresponding `rc.d` script with an argument of either 'start' or 'stop' is sufficient. As an extension over the System V behaviour, an argument of 'status' displays the service's status, and 'restart' stops and then starts the service again.

A list of scripts (and thus services) that can be ran can be found in the `/etc/rc.d` directory (hence the name of these scripts, `rc.d` scripts).

### 5.3.3 Examples

Here is an example of `rc.d` scripts that are available on a NetBSD 4.0 system:

```
netbsd% ls /etc/rc.d
DAEMON          downinterfaces  lpd             postfix
LOGIN           fixsb           mixerctl       powerd
NETWORKING     fsck            mopd           ppp
SERVERS        ftpd            motd           pwcheck
```

### 5.3. USE AN RC.D SCRIPT TO DETERMINE IF A SERVICE IS RUNNING AND START, RESTART OR STOP IT AS REQUIRED

accounting	hostapd	mountall	quota	sysct
altqd	identd	mountcritlocal	racoona	sysd
amd	ifwatchd	mountcritremote	raidframe	sysl
apache	inetd	mountd	raidframeparity	time
apmd	ipfilter	moused	rarpd	tpct
bootconf.sh	ipfs	mrouted	rbootd	ttys
bootparams	ipmon	named	root	verif
btconfig	ipnat	ndbootd	route6d	vire
btcontrol	ipsec	network	routed	wdog
bthcid	irdaattach	newsyslog	rpcbind	wscor
ccd	iscsi_target	nfsd	rtadvd	wsmo
cgd	isdnd	nfslocking	rtclocaltime	xdm
cleartmp	kdc	ntpd	rtsold	xfs
cron	ldconfig	ntpdate	rwho	ypbin
dhclient	lkm1	pf	savecore	yppar
dhcpd	lkm2	pf_boot	screenblank	ypse
dhcrelay	lkm3	pflogd	sdpd	
dmesg	local	poffd	securelevel	

Here is the same example from a system running FreeBSD 6.2:

```
freebsd% ls /etc/rc.d
DAEMON      devfs      kadmind    nfsd       rpcb
LOGIN       dhclient   kerberos   nfslocking rtadv
NETWORKING  dmesg      keyserv    nfsserver  rwho
SERVERS     dumpon     kldxref    nisdomain  savec
abi         early.sh   kpasswdd   nsswitch   sdpd
accounting  encswap    ldconfig   ntpd       secu
addswap     fsck       local      ntpdate    sendr
adjkerntz   ftpd       localpkg   othermta   seri
amd         gbde       lpd        pccard     sppp
apm         geli       mdconfig   pcvt       sshd
apmd        geli2      mdconfig2  pf         swa
archdep     hcsecd     mixer      pflog      sysc
atm1        hostapd    motd       pfsync     syst
atm2        hostname  mountcritlocal power_profile sysl
atm3        ike       mountcritremote powerd      time
auditd     inetd     mountd     ppp        tmp
```

## CHAPTER 5. CHAPTER BASIC SYSTEM ADMINISTRATION

auto_linklocal	initrandom	mountlate	pppoed
bgfsck	ip6addrctl	moused	pwcheck
bluetooth	ip6fw	mroute6d	quota
bootparams	ipfilter	mrouted	ramdisk
bridge	ipfs	msgs	ramdisk-own
bsnmpd	ipfw	named	random
bthidd	ipmon	natd	rarpd
ccd	ipnat	netif	resolv
cleanvar	ipsec	netoptions	root
cleartmp	ipxouted	network_ipv6	route6d
cron	isdnd	newsyslog	routed
devd	jail	nfsclient	routing

To determine the status of a service, run the rc.d script with 'status':

```
netbsd# sh /etc/rc.d/ipfilter status
ipf: IP Filter: v4.1.13 (396)
Kernel: IP Filter: v4.1.13
Running: yes
Log Flags: 0 = none set
Default: pass all, Logging: available
Active list: 0
Feature mask: 0x10e
```

Note that the 'status' command is not available for all scripts:

```
netbsd# sh /etc/rc.d/postfix status
/etc/rc.d/postfix: unknown directive 'status'.
Usage: /etc/rc.d/postfix [fast|force|one](start stop restart rcv)
```

To stop a service, run its rc.d script with the 'stop' argument:

```
# pgrep -lf postfix
166 /usr/libexec/postfix/master
# sh /etc/rc.d/postfix stop
postfix/postfix-script: stopping the Postfix mail system
# pgrep -lf postfix
#
```

To start it (again), use the same script with the 'start' argument:

```
# sh /etc/rc.d/postfix start
postfix/postfix-script: starting the Postfix mail system
# pgrep -lf postfix
12101 /usr/libexec/postfix/master
#
```

Now let's do this again in one command:

```
# pgrep -lf postfix
12101 /usr/libexec/postfix/master
# sh /etc/rc.d/postfix restart
postfix/postfix-script: stopping the Postfix mail system
postfix/postfix-script: starting the Postfix mail system
# pgrep -lf postfix
472 /usr/libexec/postfix/master
#
```

### 5.3.4 Practice Exercises

- Determine if your system has rc.d scripts by looking into `/etc/rc.d`
- Determine what scripts your system has
- Check if the cron daemon runs
- Assuming the cron daemon does run, stop it using the corresponding rc.d script
- Restart the cron daemon, and verify with a tool of your choice.

### 5.3.5 More information

rc(8), rc.conf(5), rc.subr(8)

## 5.4 View and configure system hardware

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.4.1 Concept**

BSD systems come with many utilities to determine what hardware is installed on a system. Know how to determine which hardware was probed at boot time as well as some BSD specific utilities which can be used to troubleshoot and manipulate PCI, ATA, and SCSI devices.

### **5.4.2 Introduction**

### **5.4.3 Examples**

### **5.4.4 Practice Exercises**

### **5.4.5 More information**

`dmesg(8)`, `/var/run/dmesg.boot`, `pciconf(8)`, `atacontrol(8)` and `cam-control(8)`; `atactl(8)` and `/kern/msgbuf`; `scsictl(8)` or `scsi(8)`

## **5.5 View, load, or unload a kernel module**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.5.1 Concept**

Understand the difference between a statically compiled kernel and one that uses loadable kernel modules. Be able to view, load and unload kernel modules on each BSD system but should be aware that kernel modules are discouraged on NetBSD and OpenBSD systems.

### 5.5.2 Introduction

### 5.5.3 Examples

### 5.5.4 Practice Exercises

### 5.5.5 More information

kldstat(8), kldload(8), kldunload(8), and loader.conf(5); modstat(8), modload(8), modunload(8), and lkm.conf(5)

## 5.6 Modify a kernel parameter on the fly

Author: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: Mark Foster mark@foster.cc FreeBSD

Reviewer: *name contact BSD flavour*

---

### 5.6.1 Concept

BSD systems maintain kernel MIB variables which allow a system administrator to both view and modify the kernel state of a running system. Be able to view and modify these MIBs both at run-time and permanently over a system boot. Recognize how to modify a read-only MIB.

### 5.6.2 Introduction

Consider this excerpt from the sysctl(8) man page on FreeBSD:

*The sysctl utility retrieves kernel state and allows processes with appropriate privilege to set kernel state. The state to be retrieved or set is described using a “Management Information Base” (MIB) style name, described as a dotted set of components.*

As you can see *sysctl* is a powerful technology to tune your system. Some *sysctl* variables can be modified on-the-fly and thus change how your system works without rebooting. Other values, when changed,

only take effect after a reboot. When this is the case, it makes (more) sense to update your `sysctl.conf/loader.conf` and reboot your system.

TODO: mention that there are a lot and the total amount varies

Some common `sysctl` variables include:

TODO: add brief description of each:

- `hw.machine_arch`
- `kern.clockrate`
- `kern.maxfiles`
- `kern.maxproc`
- `kern.ostype`
- `kern.securelevel` TODO: point to other wiki page for details
- `kern.version`
- `net.inet.ip.forwarding` TODO: point to other wiki page for details
- `vm.loadavg`

### **5.6.3 Examples**

List all `sysctl` variables:

```
# sysctl -a
```

Show subset of `sysctl` variables relevant to `cpu`:

```
# sysctl -a | grep cpu
```

Show subset of `sysctl` variables for a top-level identifier or for a sub-level identifier:

```
# sysctl kern
```

Or:

```
# sysctl net.inet
```

List only the specific variable that you need:

```
# sysctl kern.ostype
kern.ostype: FreeBSD
```

TODO: maxusers is not portable, please replace this example with maxproc or maxfiles

```
# sysctl kern.maxusers
kern.maxusers: 93
```

Update a sysctl variable:

TODO: blackhole is not portable, maybe replace with something that is portable and applicable to beginning admin

```
# sysctl net.inet.tcp.blackhole
net.inet.tcp.blackhole: 0
# sysctl net.inet.tcp.blackhole=2
net.inet.tcp.blackhole: 0 -> 2
# sysctl net.inet.tcp.blackhole
net.inet.tcp.blackhole: 2
```

Now you can test tcp blackhole with some tools like nmap. When you understand that variables you want do change in your system, you must update sysctl.conf file. In new system sysctl.conf is empty(only comment line). You can update sysctl.conf with editor like vi an save it.

```
# cat sysctl.conf
net.tcp.blackhole=2
```

Some variables, such as hardware variables that are read-only on the running system, cannot be set in sysctl.conf. In that case and you need add lines in loader.conf which is read earlier in the boot process.

The information presented here is also applicable to OpenBSD, although the kernel MIB variables do differ. Hence the blackhole example will not work on OpenBSD. In addition OpenBSD does not use a loader.conf file for adjusting kernel MIB variables.

TODO: explain how to know which values can be modified on the fly, and which require a reboot.

TODO: show on NetBSD for proc.PID or proc.\$\$

### **5.6.4 Practice Exercises**

For FreeBSD! Change on the fly these variables:

- net.inet.ip.portrange.last to 50000
- kern.maxfiles to 5000

Set these variables in system files (as described above) and re-boot, check that variables are changed after rebooting. <<<<<<<<<  
[/home/bsdwiki/source/Modifya kernelparameter onthe fly.mdwn](#)

For OpenBSD Change on the fly these variables:

- kern.maxproc to 1000
- net.inet.ip.forwarding to 1 (What does this do?)

Set these variables such that the changes will remain following subsequent reboots.

=====

1.25

### **5.6.5 More information**

sysctl(8), sysctl.conf(5), loader.conf(5)

## **5.7 View the status of a software RAID mirror or stripe**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *jdq af.dingo@gmail.com OpenBSD*

### 5.7.1 Concept

In addition to providing drivers for hardware RAID devices, BSD systems also provide built-in mechanisms for configuring software RAID systems. Know the difference between RAID levels 0, 1, 3 and 5 and recognize which utilities are available to configure software RAID on each BSD system.

### 5.7.2 Introduction

#### Software RAID

##### Hardware RAID

RAID is not a replacement for backups

##### RAID Levels

1. RAID level 0
2. RAID level 1
3. RAID level 3
4. RAID level 5

**RAIDframe:** framework for rapid prototyping of RAID structures  
RAIDframe is a software RAID solution. It is generally used when hardware raid solutions are not cost effective.

RAIDframe was developed at Carnegie Mellon University. RAIDframe, as distributed by CMU, provides a RAID simulator for a number of different architectures, and a user-level device driver and a kernel device driver for Digital Unix. Greg Oster developed this framework as a NetBSD kernel-level device driver. It has since been ported to OpenBSD and FreeBSD.

RAIDframe is not enabled by default. It enlarges a kernel image by about 500K. The significant increase in size is not acceptable for some architectures and most bootable media. Using `raidctl` improperly can lead to kernel panics.

##### **ccd**

##### **gstripe/raid/mirror**

### 5.7.3 Examples

#### RAIDframe

To view status of a RAIDframe set:

```
raidctl -vs raid0
```

All commands accept `-v` to for verbosity.

Configuration file is in `/etc/raid[0-3].conf`:

```
START array
  # numRows numCol numSpare
  1 3 1

  START disks
  /dev/sd1e
  /dev/sd2e
  /dev/sd3e

  START spare
  /dev/sd4e

  START layout
  # sectPerSU SUsPerParityUnit SUsPerReconUnit RAID\emph{level}
5
  32 1 1 5

  START queue
  fifo 100`
```

parity check of raid set raid0:

```
raidctl -P raid0
```

Fail a disk sd2 of a raid set raid0:

```
raidctl -f /dev/sd2e raid0
```

Failed disk sd2 of raid set raid0 has been replaced. Begin reconstruction:

## 5.7. VIEW THE STATUS OF A SOFTWARE RAID MIRROR OR STRIPE

---

```
raidctl -R /dev/sd2e raid0
```

Fail disk sd2 of raid set raid0:

```
faidctl -f /dev/sd2e raid0
```

Fail disk sd2 of raid set raid0 *and* begin reconstruction onto any available spare:

```
raidctl -F /dev/sd2e raid0
```

Add sd4 as hot spare to raid set raid0:

```
raidctl -a /dev/sd4e raid0
```

### 5.7.3.1 ccd

### 5.7.3.2 gstripe/raid/mirror

## 5.7.4 Practice Exercises

### RAIDframe

1. Create a set
2. Fail a disk
3. Add a hot spare
4. Reconstruct
5. Modify raid.conf

### 5.7.5 More information

### 5.7.6 Definitions

- Raid set
- Raid level
- parity
- reconstruction
- degraded mode

### **5.7.6.1 RAIDframe**

- <http://www.pdl.cmu.edu/RAIDframe/> CMU RAIDframe
- <http://www.cs.usask.ca/staff/oster/raid.html> NetBSD and RAID-frame

vinum(8), gmirror(8), gstripe(8), graid3(8), raidctl(8), ccdconfig(8)

## **5.8 Determine which MTA is being used on the system**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.8.1 Concept**

Recognize the role of the MTA, recognize which MTA(s) are available during each BSD's operating system install routine and which configuration file indicates the MTA in use on the system. Recognize the difference between the mbox or maildir mail destination file format type.

### **5.8.2 Introduction**

### **5.8.3 Examples**

### **5.8.4 Practice Exercises**

### **5.8.5 More information**

mailer.conf(5)

## 5.9 Configure system logging

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.9.1 Concept

Understand that the system automatically maintains the creation and maintenance of many different logs. Be able to configure log rotation by either time or size, understand logging facilities and priorities, as well as view compressed logs.

TODO: I think this could be split into two topic wikipages. 1) introduce syslogd and syslog.conf and logger basic facilities and levels; and 2) introduce newsyslog for rotations. –reed

### 5.9.2 Introduction

### 5.9.3 Examples

### 5.9.4 Practice Exercises

### 5.9.5 More information

Note that the newsyslog(8) implementations vary by BSD. newsyslog(8), newsyslog.conf(5), syslog.conf(5), zmore(1), bzcat(1)

## 5.10 Review log files to troubleshoot and monitor system behavior

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.10.1 Concept**

Be aware of the importance of reviewing log files on a regular basis as well as how to watch a log file when troubleshooting.

### **5.10.2 Introduction**

### **5.10.3 Examples**

### **5.10.4 Practice Exercises**

### **5.10.5 More information**

tail(1), /var/log/\*, syslog.conf(5), grep(1), dmesg(8)

## **5.11 Understand basic printer troubleshooting**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.11.1 Concept**

Be able to view the print queue and manipulate the jobs within the queue. Be able to recognize the meaning of the first two field in an /etc/printcap entry.

### **5.11.2 Introduction**

### **5.11.3 Examples**

### **5.11.4 Practice Exercises**

### **5.11.5 More information**

lpc(8), lpq(1), lprm(1), printcap(5)

## 5.12 Create or modify email aliases for Sendmail or Postfix

Author: *name contact BSD flavour*

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name contact BSD flavour*

---

### 5.12.1 Concept

Understand when to create an email alias and how to do so for either Sendmail or Postfix.

### 5.12.2 Introduction

### 5.12.3 Examples

### 5.12.4 Practice Exercises

### 5.12.5 More information

newaliases(1), aliases(5), postalias(1)

## 5.13 Halt, reboot, or bring the system to single-user mode

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.13.1 Concept

Understand the ramifications associated with halting, rebooting, or bringing a system to single-user mode, recognize when it may be necessary to do so and how to minimize the impact on a server system.

### **5.13.2 Introduction**

### **5.13.3 Examples**

### **5.13.4 Practice Exercises**

### **5.13.5 More information**

shutdown(8)

## **5.14 Recognize the difference between hard and soft limits and modify existing resource limits**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.14.1 Concept**

Understand that resource limits are inherited by the shell as well as how to view their limits and change them both temporarily and permanently. In addition, understand the difference between soft and hard limits.

### **5.14.2 Introduction**

### **5.14.3 Examples**

### **5.14.4 Practice Exercises**

### **5.14.5 More information**

limit(1), limits(1), login.conf(5); sysctl(8) on NetBSD

## 5.15 Recognize the BSD utilities that shape traffic or control bandwidth

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.15.1 Concept

Understand when it is advantageous to create policies controlling the amount of bandwidth available to specified services. In addition, recognize the utilities available on BSD systems to create bandwidth policies.

### 5.15.2 Introduction

### 5.15.3 Examples

### 5.15.4 Practice Exercises

### 5.15.5 More information

ipfw(8), altq(4), dumynet(4), altq(9), altqd(8), altq.conf(5)

## 5.16 Recognize common, possibly third-party, server configuration files

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.16.1 Concept

BSD systems are often used to provide Internet services. Be able to view or make a specified change to a service's configuration file and

recognize the names of the most commonly used configuration files and which applications they are associated with.

### **5.16.2 Introduction**

Here is a quick listing of common server program names and configuration filenames with brief descriptions and sample configuration syntax. This book doesn't cover these server configurations or maintenance.

TODO: please keep this section under a few printed pages.

### **5.16.3 Examples**

#### **5.16.3.1 Apache HTTPD**

Note: this is included as part of OpenBSD.

#### **5.16.3.2 BIND "named"**

This is included in default install.

#### **5.16.3.3 DHCP Daemon**

TODO: the BSDs have different implementations, anything common?

#### **5.16.3.4 Postfix Mail Server**

Note: Postfix is included in default install of NetBSD.

TODO: this book doesn't cover Postfix administration, but at least cross-reference to two email sections

#### **5.16.3.5 Sendmail Mail Server**

This is included in default install of DragonFly, FreeBSD, OpenBSD, and old versions of NetBSD.

TODO: this book doesn't cover sendmail administration, but at least cross-reference to two email sections

### 5.16.3.6 Samba

TODO: note that this is third-party, but also point out some native “smb” tools/features too.

### 5.16.3.7 XFree86 or Xorg

TODO: should this be briefly mentioned too?

## 5.16.4 Practice Exercises

### 5.16.5 More information

httpd.conf(5), sendmail.cf, master.cf, dhcpd.conf(5), named.conf(5), smb.conf(5)

## 5.17 Configure a service to start at boot time

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.17.1 Concept

Recognize that the BSD boot process does not use runlevels. Be able to configure essential services to start at boot time to minimize the impact of a system reboot.

### 5.17.2 Introduction

### 5.17.3 Examples

### 5.17.4 Practice Exercises

### 5.17.5 More information

rc.conf(5), rc(8), inetd(8)

## **5.18 Configure the scripts that run periodically to perform various system maintenance tasks**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.18.1 Concept**

BSD systems provide many scripts that are used to maintain and verify the integrity of the system. Be able to locate and run these scripts manually as required as well as configure which scripts run daily, weekly and monthly on each BSD system.

### **5.18.2 Introduction**

### **5.18.3 Examples**

### **5.18.4 Practice Exercises**

### **5.18.5 More information**

periodic.conf(5) and periodic(8) on Dragonfly and FreeBSD; security.conf(5), daily.conf(5), weekly.conf(5), and monthly.conf(5) on NetBSD; daily(8), weekly(8), and monthly(8) on OpenBSD

## **5.19 View the Sendmail or Postfix mail queue**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.19.1 Concept**

Be able to view the mail queue to determine if any mail is stuck in the queue, and if necessary, ask the MTA to reprocess or flush the queue.

## 5.20. DETERMINE THE LAST SYSTEM BOOT TIME AND THE WORKLOAD ON THE SYSTEM

---

### 5.19.2 Introduction

As noted in section **TODO** , the BSD systems use Sendmail or Postfix by default for handling mail.

The mail queue can be displayed using the mailq utility. The queue listing shows: TODO: see the

When using Postfix, if the mailq utility is not setup, then use “postqueue -p” to display the traditional sendmail-style queue listing.

### 5.19.3 Examples

### 5.19.4 Practice Exercises

### 5.19.5 More information

mailq(1), postqueue(1)

## 5.20 Determine the last system boot time and the workload on the system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 5.20.1 Concept

Be able to monitor the system’s workload using the time since last system reboot, as well as the system load over the last 1, 5 and 15 minutes in order to determine operation parameters.

### 5.20.2 Introduction

The uptime command can be used to show how long the system has been running since it last booted. It also shows the current time, how many users are logged in, and the system’s load averages over the past minute, five minutes and 15 minutes. For example:

```
$ \textbf{uptime}
```

```
6:17AM up 16 days, 12:28, 3 users, load averages: 0.18, 0.14, 0.10
```

The number of users is from the “utmp” database. (TODO: point to section about utmp or related??).

The time the system was booted is recorded in the kern.boottime sysctl. (The sysctl functionality is covered in section Modify a kernel parameter on the fly.)

TODO: why doesn't that sysctl have any data on DragonFly, even though it exists? And what about FreeBSD and OpenBSD?

The load average, also available from the vm.loadavg sysctl, is basically the number of processes in the system's run queue averaged over one minute, five minutes, and 15 minutes. These are processes that are ready to run – not sleeping. The system is fully utilized when this number is above 1.0. TODO: what about I/O blocking? TODO: discuss workload and performance related to this load average TODO: discuss that load average may be different per system or architecture and is not always a good reference

### **5.20.3 Examples**

### **5.20.4 Practice Exercises**

### **5.20.5 More information**

```
uptime(1), w(1), top(1)
```

## **5.21 Monitor disk input/output**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.21.1 Concept**

A system's disk input/output can have a dramatic impact on performance. Know how to use the utilities available on BSD systems to

monitor disk I/O and interpret their results.

### **5.21.2 Introduction**

### **5.21.3 Examples**

### **5.21.4 Practice Exercises**

### **5.21.5 More information**

iostat(8), systat(1), vmstat(1), nfsstat(1)

## **5.22 Deal with busy devices**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.22.1 Concept**

Understand what can cause a process to hang, how to detect related processes and how to fix the situation.

### **5.22.2 Introduction**

### **5.22.3 Examples**

### **5.22.4 Practice Exercises**

### **5.22.5 More information**

ps(1), fstat(1), kill(1), umount(8) and the third-party lsof utility

## **5.23 Determine information regarding the operating system**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **5.23.1 Concept**

Be able to determine the type and version of the operating system installed.

### **5.23.2 Introduction**

### **5.23.3 Examples**

### **5.23.4 Practice Exercises**

### **5.23.5 More information**

uname(1), sysctl(8); /etc/release on NetBSD

## **5.24 Understand the advantages of using a BSD license**

Author: *name contact BSD flavour*

Reviewer: Jeremy C. Reed, reed, NetBSD/FreeBSD/DragonFly/OpenBSD

Reviewer: *name contact BSD flavour*

---

### **5.24.1 Concept**

Recognize the 2-clause BSD license and how the license does not place restrictions on whether BSD licensed code remains Open Source or becomes integrated into a commercial product.

TODO: might as well cover 3- and 4-clause licenses too since a lot still uses that and also mention the UC removal of advertising clause – note this is important as that only applies to UCB’s code and not to third-party code included with BSDs that may have used advertising clause. NetBSD for example continues to use full old style license.

**5.24.2 Introduction**

**5.24.3 Examples**

**5.24.4 Practice Exercises**

**5.24.5 More information**



# 6 Chapter Network Administration

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

TCP/IP was originally implemented on BSD systems and BSD systems continue to provide core networking services for a substantial portion of the Internet. Demonstrate a strong understanding of both IPv4 and IPv6 addressing as well as basic networking theory. Trainers and material providers should provide conceptual depth similar to that found in Network+ or in the networking theory section of CCNA.

- Determine the current TCP-IP settings on a system
- Set a system's TCP-IP settings
- Determine which TCP or UDP ports are open on a system
- Verify the availability of a TCP-IP service
- Query a DNS server
- Determine who is responsible for a DNS zone
- Change the order of name resolution
- Convert a subnet mask between dotted decimal, hexadecimal or CIDR notation
- Gather information using an IP address and subnet mask
- Understand IPv6 address theory
- Demonstrate basic tcpdump(1) skills

- Manipulate ARP and neighbor discovery caches
- Configure a system to use NTP
- View and renew a DHCP lease
- Recognize when and how to set or remove an interface alias

## 6.1 Determine the current TCP/IP settings on a system

Author: Alex Nikiforov [nikiforov.al@gmail.com](mailto:nikiforov.al@gmail.com) FreeBSD

Reviewer: Sean Swayze [swayze@pcsage.biz](mailto:swayze@pcsage.biz) FreeBSD

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

### 6.1.1 Concept

Be able to determine a system's IP address(es), subnet mask, default gateway, primary and secondary DNS servers and hostname.

### 6.1.2 Introduction

If you are a BSD user/administrator you must understand where and how you can get any information about a system such as its network settings. What interesting information about a network can we get from the system? We can obtain its IP address, default gateway, the DNS server, the MAC address of any network interface on the system and other relevant information related to networking.

### 6.1.3 Examples

Let's start from IP address and MAC address. We can get this kind of information from **ifconfig** command. For example

```
wi0: flags=8802 <BROADCAST,SIMPLEX,MULTICAST> mtu 1500
    ether 00:05:3c:08:8f:7e
    media: IEEE 802.11 Wireless Ethernet autoselect (none)
    status: no carrier
```

## 6.1. DETERMINE THE CURRENT TCP/IP SETTINGS ON A SYSTEM

---

```
ssid "" channel 1
stationname "FreeBSD WaveLAN/IEEE node"
authmode OPEN privacy OFF txpovmax 100 bmiss 7
fxp0: flags=8843 <UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
options=8
inet 192.168.1.162 netmask 0xffffffff broadcast 192.168.1.255
ether 00:09:6b:13:42:9f
media: Ethernet autoselect (100baseTX <full-duplex>)
status: active
lo0: flags=8049 <UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
inet6 ::1 prefixlen 128
inet 127.0.0.1 netmask 0xff000000
```

As we can see the fxp0 interface has IP 192.168.1.162/24 (/24 means that the network mask is 255.255.255.0 - fffffff0), broadcast address 192.168.1.255, MAC address 00:09:6b:13:42:9f and 100baseTX full-duplex connection to the switch. Also the system has a wifi interface wi0 and also lo0 - the loopback interface.

Next step is to determine the DNS servers and default route.

```
#netstat -rn
```

```
Routing tables
```

```
Internet:
```

Destination	Gateway	Flags	Refs	Use	Netif
default	192.168.1.1	UGS	0	919	fxp0
127.0.0.1	127.0.0.1	UH	0	0	lo0
192.168.1	link#2	UC	0	0	fxp0
192.168.1.1	00:13:46:56:cf:15	UHLW	2	0	fxp0

That means that the default gateway IP is 192.168.1.1.

```
> cat /etc/resolv.conf
```

```
nameserver 192.168.1.1
```

```
nameserver 10.2.2.1
```

```
>
```

**resolv.conf** has IP addresses of DNS server. For this example the system will first try to resolve DNS name with 192.168.1.1, secondly

with 10.2.2.1(The system will really try to resolve DNS name with hosts file, if the name is not in the hosts file system (hosts.conf) try to resolve it with a DNS server). You can edit **resolv.conf** on the fly.

Some times system have some static route for hosts on the network. For save this you can use **rc.conf** file. And you can update routes on the fly. For example, if you need change default route. Let's try changing the default route:

```
# route flush
default          192.168.1.1      done
# route add 0.0.0.0 192.168.1.1
add net 0.0.0.0: gateway 192.168.1.1
#
```

Route flush means that you want flush all routes on your system, instead of this you can use the **route delete** command (look at the manual for your system). **route add 0.0.0.0** means that you want add route for 0.0.0.0 network - all networks(also you can do it like that **route add default 192.168.1.1** ) and 192.168.1.1 it's IP for your default router.

#### 6.1.4 Practice Exercises

1. Try to access your DNS-servers.
2. List the IP addresses of each interface, the default router, list the DNS servers.
3. Log into your system and verify that the DNS servers correspond to that of your ISP or your own.
4. Log into your system and verify that you have a valid, IP address and default gateway.

#### 6.1.5 More information

ifconfig(8), netstat(1), resolv.conf(5), route(8), hostname(1)

## 6.2 Set a system's TCP/IP settings

Author: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin yannick@diablotin.fr FreeBSD/OpenBSD

---

### 6.2.1 Concept

Be able to modify required TCP/IP settings both temporarily and permanently in order to remain after a reboot.

### 6.2.2 Introduction

### 6.2.3 Examples

### 6.2.4 Practice Exercises

### 6.2.5 More information

hostname (1), ifconfig(8), route(8), resolv.conf(5), rc.conf(5), hosts(5), hostname.if(5), myname(5), mygate(5), netstart(8)

## 6.3 Determine which TCP or UDP ports are open on a system

Author: Mark Foster mark@foster.cc FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin yannick@diablotin.fr FreeBSD/OpenBSD

---

### 6.3.1 Concept

Be able to use the utilities found on BSD systems as well as third-party programs to determine which ports are open on a system and which ports are being seen through a firewall.

### 6.3.2 Introduction

### 6.3.3 Examples

### 6.3.4 Practice Exercises

### 6.3.5 More information

netstat(1), services(5) and fstat(1); sockstat(1) and third-party nmap and lsof

## 6.4 Verify the availability of a TCP/IP service

Author: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin yannick@diablotin.fr FreeBSD/OpenBSD

---

### 6.4.1 Concept

Be able to determine if a remote system is available via TCP/IP, and if so, telnet(1) to a particular TCP service to determine if it is responding to client requests.

### 6.4.2 Introduction

### 6.4.3 Examples

### 6.4.4 Practice Exercises

### 6.4.5 More information

ping(8), traceroute(8), telnet(1); nc(1) on FreeBSD and OpenBSD

## 6.5 Query a DNS server

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: Mark Foster mark@foster.cc FreeBSD

Reviewer: *name contact BSD flavour*

---

## 6.6. DETERMINE WHO IS RESPONSIBLE FOR A DNS ZONE

### **6.5.1 Concept**

Understand basic DNS theory, including types of resource records, types of DNS servers, reverse lookups and zone transfers. Be able to query a DNS server for a particular type of resource record, understand which servers are authoritative for a zone and determine if a DNS server is willing to do a zone transfer.

### **6.5.2 Introduction**

### **6.5.3 Examples**

### **6.5.4 Practice Exercises**

### **6.5.5 More information**

dig(1), host(1), nslookup(1), ping(8), telnet(1)

## **6.6 Determine who is responsible for a DNS zone**

Author: Cezary Morga [cm@therek.net](mailto:cm@therek.net) FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **6.6.1 Concept**

Be able to perform a reverse DNS lookup to determine the network associated with an IP address and gather contact information regarding that network.

### **6.6.2 Introduction**

Being a BSD system administrator requires the knowledge of obtaining contact informations of persons responsible for a given DNS zone. This is most commonly achieved through a reverse DNS lookup.

### 6.6.3 Examples

Having only an IP address, the first step is to perform a reverse DNS lookup for a given address to obtain information on domain to which this machine belongs to. As it will be shown below using **dig** and **whois** commands give us slightly different output.

With the **dig** command the reverse DNS lookup is provided by the use of **-x** flag. Information we're looking for is placed within SOA record.

```
# \textbf{dig SOA -x 216.239.32.10}

; <<>> DiG 9.3.3 <<>> SOA -x 216.239.32.10
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36277
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;10.32.239.216.in-addr.arpa.      IN      SOA

;; AUTHORITY SECTION:
32.239.216.in-addr.arpa. 10300  IN      SOA      ns1.google.com.

;; Query time: 2 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Jan  4 23:36:06 2007
;; MSG SIZE rcvd: 104
```

Information obtained with this command is an e-mail address to person responsible for a given DNS zone. This information is located just after the hostname and noted using **.** (dot) instead of **@** character. In this case it is `dns-admin.google.com` which should be understood as `dns-admin@google.com`.

The **whois** command does not require any additional parameters to perform a reverse DNS lookup and it provides a far more detailed contact information.

```
# \textbf{whois 216.239.32.10}
```

## 6.6. DETERMINE WHO IS RESPONSIBLE FOR A DNS ZONE

```
OrgName:    Google Inc.
OrgID:      GOGL
Address:    1600 Amphitheatre Parkway
City:       Mountain View
StateProv:  CA
PostalCode: 94043
Country:    US

NetRange:   216.239.32.0 - 216.239.63.255
CIDR:       216.239.32.0/19
NetName:    GOOGLE
NetHandle:  NET-216-239-32-0-1
Parent:     NET-216-0-0-0-0
NetType:    Direct Allocation
NameServer: NS1.GOOGLE.COM
NameServer: NS2.GOOGLE.COM
NameServer: NS3.GOOGLE.COM
NameServer: NS4.GOOGLE.COM
Comment:
RegDate:    2000-11-22
Updated:    2001-05-11
```

```
RTechHandle: ZG39-ARIN
RTechName:   Google Inc.
RTechPhone:  +1-650-318-0200
RTechEmail:  arin-contact@google.com
```

```
OrgTechHandle: ZG39-ARIN
OrgTechName:   Google Inc.
OrgTechPhone:  +1-650-318-0200
OrgTechEmail:  arin-contact@google.com
```

```
# ARIN WHOIS database, last updated 2007-01-03 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

Notice, that the layout of **whois** query output depends on many factors, but still gives similiar detailed information. Notice also, that

the information gained from **whois** query on an IP address may be different than the information gained when querying a domain name pointing to the very same IP address. Most commonly it takes place when the domain administrator is a different organization than an IP address provider.

#### 6.6.4 Practice Exercises

1. Using both commands check contact informations available for your domain.
2. Add different server's names/addresses (ie. your own, your ISP's) to **dig** @server parameter.
3. Perform **whois** query on your domain name and IP address.

#### 6.6.5 More information

dig(1) and whois(1)

### 6.7 Change the order of name resolution

Author: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin yannick@diablotin.fr FreeBSD/OpenBSD

---

#### 6.7.1 Concept

Be able to determine the default order of host name resolution on BSD systems and recognize which configuration file controls the order of host name resolution.

## 6.8. CONVERT A SUBNET MASK BETWEEN DOTTED DECIMAL, HEXADECIMAL OR CIDR NOTATION

---

### 6.7.2 Introduction

### 6.7.3 Examples

### 6.7.4 Practice Exercises

### 6.7.5 More information

ping(8), telnet(1), nsswitch.conf(5), resolv.conf(5), host.conf(5)

## 6.8 Convert a subnet mask between dotted decimal, hexadecimal or CIDR notation

Author: AndreasKuehl andreas dot kuehl at clicktivities dot net *BSD flavour* FreeBSD

Reviewer: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: Yannick Cadin yannick@diablotin.fr FreeBSD/OpenBSD

---

### 6.8.1 Concept

Be familiar with IPv4 addressing and how to convert a subnet mask from a given notation to another specified notation.

### 6.8.2 Introduction

All the internet address space is divided into subnets. In the old times, there were class A, class B and class C nets. A subnet means, that you divide an IPv4 address in a front part and a back part. The front part is common in the subnet, all addresses of a subnet have the same front part. All computers/devices in the subnet are distinguished by different values for the back part. A class A net had the first byte of an IPv4 address common and could contain  $255*255*255$  addresses, a class B net had the first two bytes common and contained  $255*255$  addresses while, you guess, a class C net had the first three bytes common and contained 255 addresses. Nowadays, the address space is precious and nobody wants to block a complete class C net for only 6 addresses. Until 1993, the internet routers did not know how to

distinguish, whether a certain address was contained in a class A, B or C net. Instead, certain blocks of IP addresses contained only class C nets and other blocks contained only class B or class A nets. Since 1993 the borders of net sizes are free. Additionally, the length of the first part of an IPv4 address is not bound any more to the byte and could be somewhere.

There are three commonly known and used methods to write the so called subnetmask, which shows the border between front or prefix and back part.

(You need to know how to convert between decimal, hexadecimal, and binary numbers. If you can not do so, go elsewhere and learn!)

255.255.255.0	dotted decimal
ff.ff.ff.00	hexadecimal
/24	CIDR

Every of this netmasks work on the binary representation of an IP address.

192.168.6.4	is a decimally written address
11000000 10101000 00000110 00000100	is the binary representation

If you convert the dotted decimal or hexadecimal form to binary, you will get something like this.

11111111 11111111 11111111 00000000

If you count from left to right, you count 24 times figure 1.

Dotted decimal and hexadecimal are two different representations for the same system. If you convert them, you get the same. The CIDR form says just: count from left to right.

But know, what does it mean And what do we do with it?

Let's say, you got a class C net for your company and have to divide it for several purposes...

(To be continued :-)

### 6.8.3 Examples

### 6.8.4 Practice Exercises

### 6.8.5 More information

[http://en.wikipedia.org/wiki/ClasslessInter-Domain Routing](http://en.wikipedia.org/wiki/ClasslessInter-Domain_Routing)

## 6.9 Gather information using an IP address and subnet mask

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

### 6.9.1 Concept

Given an IPv4 address and subnet mask, be able to determine the subnet address, broadcast address and the valid host addresses available on that subnet address.

### 6.9.2 Introduction

### 6.9.3 Examples

### 6.9.4 Practice Exercises

### 6.9.5 More information

## 6.10 Understand IPv6 address theory

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 6.10.1 Concept

Be able to recognize basic IPv6 addressing theory including: the components of an IPv6 address; the support for multiple addresses (link, local, global) per interface; address and prefix representation (aaaa:bbbb::dddd/17) and the address format (48bit prefix, 16bit subnet, 64 hostbits). In addition, understand the autoconfiguration process where the router sends its prefix or gets queried and the host adds its 64 host-bits which are derived from its MAC address. Finally, be able to troubleshoot basic IPv6 connectivity.

### 6.10.2 Introduction

### 6.10.3 Examples

### 6.10.4 Practice Exercises

### 6.10.5 More information

ifconfig(8), ping6(8), rtsol(8)

## 6.11 Demonstrate basic tcpdump(1) skills

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 6.11.1 Concept

Given some tcpdump(1) output, an admin should be able to answer basic network connectivity questions. Recognize common TCP and UDP port numbers, the difference between a TCP/IP server and a TCP/IP client, and the TCP three-way handshake.

### 6.11.2 Introduction

### 6.11.3 Examples

### 6.11.4 Practice Exercises

### 6.11.5 More information

tcpdump(1)

## 6.12 Manipulate ARP and neighbor discovery caches

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 6.12.1 Concept

Understand basic ARP theory as well as the neighbor discovery cache used on IPv6 networks. Be able to view, modify and clear these caches and recognize when it is necessary to do so.

### 6.12.2 Introduction

### 6.12.3 Examples

### 6.12.4 Practice Exercises

### 6.12.5 More information

arp(8), ndp(8)

## 6.13 Configure a system to use NTP

Author: *name contact BSD flavour*

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name contact BSD flavour*

---

### 6.13.1 Concept

Be familiar with the concepts in RFC 868, the importance of synchronizing time on server systems and which services in particular are time sensitive. Be able to configure NTP and manually synchronize with a time server as required.

### 6.13.2 Introduction

### 6.13.3 Examples

### 6.13.4 Practice Exercises

### 6.13.5 More information

`ntpd(8)`, `ntpd.conf(5)`, `rc.conf(5)`, `rdate(8)`

## 6.14 View and renew a DHCP lease

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

### 6.14.1 Concept

An admin should have a basic understanding of DHCP leases and how to configure a client to override the settings received from a DHCP server. In addition, be able to view the current lease, release it and renew a lease. Since the DHCP client used varies, be familiar with using the DHCP client commands on each BSD.

### 6.14.2 Introduction

### 6.14.3 Examples

### 6.14.4 Practice Exercises

### 6.14.5 More information

`dhclient(8)`, `dhclient.leases(5)`, `dhclient.conf(5)`, `rc.conf(5)`

## 6.15 Recognize when and how to set or remove an interface alias

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 6.15.1 Concept

Recognize when it is appropriate to set or remove an interface alias and the available commands on each of the BSDs.

### 6.15.2 Introduction

### 6.15.3 Examples

### 6.15.4 Practice Exercises

### 6.15.5 More information

ifconfig(8), rc.conf(5), ifaliases(5), hostname.if(5)



# 7 Chapter Basic Unix Skills

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: Yannick Cadin [yannick@diablotin.fr](mailto:yannick@diablotin.fr) FreeBSD/OpenBSD

---

BSD has its roots in Unix and many Unix utilities were originally developed on BSD systems. Demonstrate proficiency in the most commonly used Unix command line utilities.

- Demonstrate proficiency in using redirection, pipes and tees
- Recognize, view and modify environmental variables
- Be familiar with the vi(1) editor
- Determine if a file is a binary, text, or data file
- Locate files and binaries on a system
- Find a file with a given set of attributes
- Create a simple Bourne shell script
- Find appropriate documentation
- Recognize the different sections of the manual
- Verify a file's message digest fingerprint (checksum)
- Demonstrate familiarity with the default shell
- Read mail on the local system
- Use job control
- Demonstrate proficiency with regular expressions

- Overcome command line length limitations
- Understand various “domain” contexts
- Configure an action to be scheduled by cron(8)

## 7.1 Demonstrate proficiency in using redirection, pipes and tees

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.1.1 Concept

Be able to to redirect standard input, output or error, use a pipe to send the output of one command to another command or file, and use a tee to copy standard input to standard output.

### 7.1.2 Introduction

### 7.1.3 Examples

### 7.1.4 Practice Exercises

### 7.1.5 More information

<, >, |, tee(1), >\& and | \&

## 7.2 Recognize, view and modify environmental variables

Author: *Ivan Voras IvanVoras FreeBSD*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.2.1 Concept

Be able to view and modify environmental variables both temporarily and permanently for each of the default shells found on BSD systems.

### 7.2.2 Introduction

Environment variables are key-value pairs available to executing processes. By way of environment variables, users (and other processes) can pass data to new processes. Both keys and values can only be strings, and both are usually case sensitive. In shell scripts interpreted by `/bin/sh` (as well as many others), environment variable contents are referenced by `${KEY}`.

Some environment variables need only to be set, without regards for their content (one example is the `DEBUG` variable), and there are usually command shortcuts for this operation.

Most shells have their own internal variables, which should not be confused with global environment variables as they are not passed to newly started processes.

Different shells have different commands for manipulating environment variables. Read more about their syntax in the appropriate manual pages.

#### 7.2.2.1 `sh`, `bash`

Internal shell variables can be set simply by issuing a statement like “`key=value`“, and inspected with the `set` command. An internal variable can then be promoted to a global environment variable with the `export` command. Internal shell variables can be deleted with `unset`. If the internal variable was exported at the time, it will also be deleted from the environment.

Shell variables are only valid within a shell process instance (spawned subshells will not contain their parent’s internal variables).

In order to just set an environment variable with empty content, use the form “`export NAME`“ without defining the internal shell variable.

### 7.2.2.2 csh, tcsh

Internal shell variables can be set and inspected with the `set` command, and environment variables by the `setenv` command. Internal shell variables can be deleted with `unset`, and environment variables deleted with `unsetenv` command.

To set an environment variable to empty content, use `setenv NAME`.

### 7.2.2.3 Common environment variables

There are environment variables which have well defined meanings for a Unix process. Some of them are:

- `USER` : Currently logged-in user (e.g. username)
- `HOME` : Currently logged-in user's home directory (e.g. `/home/ivoras`)
- `TERM` : Active terminal (console) type (e.g. `xterm`)
- `EDITOR` : User's preferred text file editor (e.g. `vi`)
- `VISUAL` : User's preferred visual file editor (e.g. `emacs`)
- `PAGER` : User's preferred pager (e.g. `/usr/bin/more`)
- `PATH` : User's search path for executables (e.g. `/bin:/usr/bin:/usr/local/bin`)

### 7.2.3 Examples

Automatically set and export an environment variable called "VEGETABLE" to "Carrot", in `bash` :

```
$ export VEGETABLE=Carrot
```

Create an environment variable called "VEHICLE" containing the string "Truck", in `tcsh` :

```
> setenv VEHICLE Truck
```

List environment variables, in `tcsh` :

```
> setenv
```

Note that *(ba)sh* uses "=" to set environment variables, and *(t)csh* doesn't.

### 7.2.4 Practice Exercises

1. Investigate what does PWD environment variable do
2. Experiment with setting the PAGER environment variable and the behavior of the manual page viewer (man)
3. Investigate how does internal variable shlvl behave in *(t)csH* when spawning subshells
4. Unset the PATH environment variable and see if you can start programs without specifying their full path

### 7.2.5 More information

env(1), sh(1), csh(1), tcsh(1), environ(7)

## 7.3 Be familiar with the vi(1) editor

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.3.1 Concept

The default editor on BSD systems is often vi(1) and many system utilities require familiarity with vi(1) commands. Be able to edit files using this editor, as well as modify a read-only file or exit vi(1) without saving any edits to the file.

### 7.3.2 Introduction

ex, Bill Joy.

USD/12.vi

### 7.3.3 Examples

### 7.3.4 Practice Exercises

1. Arrow keys
2. Getting out of the editor
3. Moving around in the file
4. Making simple changes
5. Writing, quitting, editing new files

### 7.3.5 More information

vi(1) including: :w, :wq, :wq!, :q!, dd, y, p, x, i, a, /, :, :r, ZZ, :set number, :set list

## 7.4 Determine if a file is a binary, text, or data file

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.4.1 Concept

While BSD systems use naming conventions to help determine the type of file, an admin should be aware that these are conventions only and that there is a magic database to help determine file type.

### 7.4.2 Introduction

### 7.4.3 Examples

### 7.4.4 Practice Exercises

### 7.4.5 More information

file(1), magic(5)

## 7.5 Locate files and binaries on a system

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.5.1 Concept

Be able to quickly find the location of any file on the system as needed and know which utilities can be used to find binaries, source, man-pages and files. In addition, be able to update the locate(1) database.

### 7.5.2 Introduction

### 7.5.3 Examples

### 7.5.4 Practice Exercises

### 7.5.5 More information

whatis(1); whereis(1); which(1); locate(1); find(1); sh(1) including “type” built-in, -v and -V; locate.updatedb(8) or locate.conf(5)

## 7.6 Find a file with a given set of attributes

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.6.1 Concept

The find(1) utility is invaluable when searching for files matching a specific set of attributes. Be comfortable in using this utility and may be asked to locate files according to last modification time, size, type, file flags, UID or GID, permissions or by a text pattern.

## 7.6.2 Introduction

## 7.6.3 Examples

## 7.6.4 Practice Exercises

## 7.6.5 More information

find(1)

# 7.7 Create a simple Bourne shell script

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

## 7.7.1 Concept

Most system administration tasks can be automated with shell scripts. Be aware of the advantages and disadvantages of using a Bourne shell script rather than a csh(1) or bash(1) shell script. Be able to recognize a shebang, comments, positional parameters and special parameters, wildcards, the proper use of quotes and backslashes and: for, while, if, case, and exec. In addition, know how to make a script executable and how to troubleshoot a script.

## 7.7.2 Introduction

## 7.7.3 Examples

## 7.7.4 Practice Exercises

## 7.7.5 More information

sh(1), chmod(1)

# 7.8 Find appropriate documentation

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **7.8.1 Concept**

BSD systems are well documented and there are many detailed resources available to the system administrator. Be able to use the documentation found on the system itself as well as be aware of the resources available on the Internet.

### **7.8.2 Introduction**

### **7.8.3 Examples**

### **7.8.4 Practice Exercises**

### **7.8.5 More information**

apropos(1), man(1), man.conf(5), whatis(1), and info(1); share/doc/ and share/examples/; in addition, each BSD project maintains an on-line handbook and several mailing lists

## **7.9 Recognize the different sections of the manual**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **7.9.1 Concept**

Recognize what type of information is found in each section of the manual. In addition, be able to specify a specific section of the manual, ask to see all sections of the manual, and do a search query within the manual.

## 7.9.2 Introduction

The BSD system provides useful and detailed documentation for most utilities, common configuration files, programming functions, and various procedures. These are known as manual (or man) pages and the manual may be read using the “man” command.

The manuals are categorized by various sections, usually by number but sometimes by letter or a word or other description. The standard categories are:

- 1 General documentation covering standard tools and utilities
- 2 Programmer manual pages covering system calls and definitions
- 3 Programmer documentation covering library functions
- 4 Documentation covering hardware devices, kernel interfaces and drivers
- 5 Documentation covering various binary and configuration file formats
- 6 Documentation for games and amusement
- 7 Miscellaneous documentation covering concepts and procedures not categorized in other sections.
- 8 Documentation for system maintenance tools, utilities and procedures
- 9 Programmer documentation covering kernel interfaces and driver development

TODO: maybe give a few examples

TODO: Search order

TODO: other sections

TODO: brief intro to nroff

TODO: brief intro to cat pages (preformatted man pages)

TODO: how to see all sections?

TODO: mention man pages in other locations like from installed packages or third-party software

## 7.10. VERIFY A FILE'S MESSAGE DIGEST FINGERPRINT (CHECKSUM)

---

### 7.9.3 Examples

### 7.9.4 Practice Exercises

TODO: show difference between “ed” and “ed” as an example

### 7.9.5 More information

man (1), intro(1) to intro(9), “/”

TODO: why “/” in this more information?

## 7.10 Verify a file's message digest fingerprint (checksum)

Author: Alex Nikiforov nikiforov.al@gmail.com FreeBSD

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.10.1 Concept

Be familiar with the theory behind a message digest fingerprint and why it is important to verify a file's fingerprint. In addition, be able to create a fingerprint as well as verify an existing fingerprint.

### 7.10.2 Introduction

When you download some file from a server and you don't trust this server how can you verify that file is real, without any bogus parts? You can use fingerprint of this file for verify it.

### 7.10.3 Examples

You download **file** from some mirror in your country and want to verify it. Let's do it.

```
> md5 file
```

```
MD5 (file) = d3762ac7a4e45f8262aeb3362bb1f9b7
```

```
> sha1 file
```

SHA1 (file) = 67d59b7fe01074dba2462e13633bd163453bff47

Now we have fingerprint for **file** and can verify md5 and sha1 fingerprint from server.

#### 7.10.4 Practice Exercises

Get few fingerprints from your system via md5 and sha1

#### 7.10.5 More information

md5(1), openssl(1), sha1(1), cksum(1)

### 7.11 Demonstrate familiarity with the default shell

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

#### 7.11.1 Concept

Be comfortable using the sh(1), csh(1) or tcsh(1) shells. Be able to modify shell behavior both temporarily and permanently including: prevent the shell from clobbering existing files, use history substitution, and set command aliases to save time at the command line. Know how to temporarily bypass a command alias.

#### 7.11.2 Examples

#### 7.11.3 Practice Exercises

#### 7.11.4 More information

sh(1), csh(1), and tcsh(1) including: !, !!, \\$, 0, h, t, r, p, Introduction

## 7.12 Read mail on the local system

Author: jdq af.dingo@gmail.com OpenBSD

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.12.1 Concept

Be aware that by default, system messages may be emailed to the root user on the local system and that a third-party MUA may not be installed. Be able to both read and send mail using the built-in mail(1) command. Know the location of user mailbox files.

### 7.12.2 Introduction

summary of the topics covered under the Mail Reference Manual would be fitting here.

### 7.12.3 Examples

### 7.12.4 Practice Exercises

### 7.12.5 More information

mail(1), /var/mail/\\$USER

## 7.13 Use job control

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.13.1 Concept

Know how to start a process in the background, place an existing process into the background, and return a background process to the foreground. Be able to verify if any jobs are currently in the background

and be aware of the difference between `kill(1)` and the shell built-in “kill”.

### 7.13.2 Introduction

### 7.13.3 Examples

### 7.13.4 Practice Exercises

### 7.13.5 More information

`\&`, `CTRL-Z`, `jobs`, `bg`, `fg`, and “kill” which are all built-in to the shell

## 7.14 Demonstrate proficiency with regular expressions

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### 7.14.1 Concept

Regular expressions are part of the daily life of a system administrator. Be able to match text patterns when analyzing program output or searching through files. Be able to specify a range of characters within brackets `[]`, specify a literal, use a repetition operator, recognize a metacharacter and create an inverse filter.

### 7.14.2 Introduction

### 7.14.3 Examples

### 7.14.4 Practice Exercises

### 7.14.5 More information

`grep(1)`, `egrep(1)`, `fgrep(1)`, `re_format(7)`

## **7.15 Overcome command line length limitations**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **7.15.1 Concept**

The command line length is limited, and often a command should be applied to more arguments than fit on a command line. Understand how to run the command multiple times with different arguments for each call using `xargs(1)` or a shell “while” read loop.

### **7.15.2 Introduction**

### **7.15.3 Examples**

### **7.15.4 Practice Exercises**

### **7.15.5 More information**

`xargs(1)`, `find(1)`

## **7.16 Understand various “domain” contexts**

Author: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

Reviewer: *name contact BSD flavour*

---

### **7.16.1 Concept**

The term “domain” is used in Unix for several facilities. Understand the meaning of the term in the context of the Network Information System (NIS), the Domain Name System (DNS), Kerberos, and NTLM domains.

### **7.16.2 Introduction**

### **7.16.3 Examples**

### **7.16.4 Practice Exercises**

### **7.16.5 More information**

domainname(1), resolv.conf(5), krb5.conf(5), smb.conf(5)

## **7.17 Configure an action to be scheduled by cron(8)**

Author: *name contact BSD flavour*

Reviewer: Sean Swayze swayze@pcsaage.biz FreeBSD/OpenBSD

Reviewer: *name contact BSD flavour*

---

### **7.17.1 Concept**

Understand the difference between the system crontab and user crontabs. In addition, be familiar with using the crontab editor, be able to recognize the time fields seen in a crontab, and understand the importance of testing scripts before scheduling their execution through cron(8). Recognize that the files `/var/cron/allow` and `/var/cron/deny` can be created to control which users can create their own crontabs.

### **7.17.2 Introduction**

### **7.17.3 Examples**

### **7.17.4 Practice Exercises**

### **7.17.5 More information**

crontab(1), cron(8), crontab(5)

# Index

`/var/cron/allow`, 120  
`/var/cron/deny`, 120

accounting, 48  
alias, command, 116  
alias, interface, 103  
ARP, 101  
audit-packages, 15  
autoconfiguration, IPv6, 100

background, 117  
Bourne shell, 112

case, 112  
compression, 73  
CPU, 55  
crontab, 120

DHCP, 102  
DNS, 119  
documentation, 113  
domain, 119

editing, 109  
environment variables, 107  
exec, 112  
expressions, 118

file types, 110  
find, 111  
fingerprint, 115

flavors, 7  
for, 112  
foreground, 117

history, 116

if, 112  
installation, 2  
IPv6, 100, 101

jobs, 117

Kerberos, 119  
kill(1, 56  
killall(1, 59

lease, DHCP, 102  
locate, 111  
logging, 73

MAC, 100  
magic, 110  
mailbox, 117  
maildir, 72  
man pages, 113  
manual, 113  
mbox, 72  
media, 8  
message digest, 115  
metacharacter, 118  
mk.conf, 12

MTA, 72  
MUA, 117

neighbor discovery cache, 101  
networking, 100  
nice, 56  
nice-level, 56  
NIS, 119  
NTLM, 119

package, 15  
packages, 10, 13  
pgrep(1, 58  
PID, 55  
pipe, 106  
pkgsrc, 10, 13  
pkill(1, 59  
port, TCP or UDP, 100  
portaudit, 15  
ports, 10, 13  
processes, 117  
ps(1, 55

regular expressions, 118  
release-map, 7  
RELENG, 7  
renice, 55

scripts, 112  
shebang, 112  
shell, 107  
SIGKILL, 56, 58  
signals, 58  
SIGTERM, 56, 58  
standard error, 106  
standard input, 106  
standard output, 106  
sysinst, 8  
sysinstall, 8  
systat(1, 55  
system accounting, 48

TCP, 100  
terminal, 48  
time, network, 102  
top(1, 55  
tty, 48

UDP, 100  
upgrade, 8

variables, 107  
vuxml, 15

while, 112, 119  
www.bsdinstaller.org, 8

xargs, 119