

sub title

The Title

Title:
Subtitle
March 2007

Copyright © 2006-2007 BSD Certification Group, Inc.

Permission to use, copy, modify, and distribute this documentation for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE DOCUMENTATION IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS DOCUMENTATION INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS DOCUMENTATION.

NetBSD and pkgsrc are registered trademarks of the NetBSD Foundation, Inc.
FreeBSD is a registered trademark of the FreeBSD Foundation.

Contents

Introduction	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	6
1.3 Understand the difference between a pre-compiled binary and compiling from source	8
1.4 Understand when it is preferable to install a pre-compiled binary and how to do so	9
1.5 Recognize the available methods for compiling a customized binary	10
1.6 Determine what software is installed on a system	10
1.7 Determine which software requires upgrading	11
1.8 Upgrade installed software	11
1.9 Determine which software have outstanding security advisories	12
1.10 Follow the instructions in a security advisory to apply a security patch	12
2 Chapter Securing the Operating System	15
2.1 Determine the system's security level	16
2.2 Recognize basic recommended access methods	16
2.3 Configure an SSH server according to a set of requirements	21
2.4 Configure an SSH server to use a key pair for authentication	21
2.5 Preserve existing SSH host keys during a system upgrade	23
2.6 Recognize alternate authentication mechanisms	24
2.7 Recognize alternate authorization schemes	24
2.8 Recognize BSD firewalls and rulesets	25
2.9 Recognize the BSD utilities that shape traffic or control bandwidth	26
2.10 Recognize BSD mechanisms for encrypting devices	27
2.11 Recognize methods for verifying the validity of binaries	27
2.12 Recognize the BSD methods for restraining a service	28
2.13 Modify the system banner	28
3 Chapter Files, Filesystems and Disks	31
3.1 Mount or unmount local filesystems	32

3.2	Configure data to be available through NFS	33
3.3	Determine which filesystems are currently mounted and which will be mounted at system boot	34
3.4	Determine disk capacity and which files are consuming the most disk space	36
3.5	Create and view symbolic or hard links	36
3.6	View ACLs	37
3.7	View file permissions and modify them using either symbolic or octal mode	49
3.8	Modify a file's owner or group	50
3.9	Backup and restore a specified set of files and directories to local disk or tape	51
3.10	Backup and restore a file system	52
3.11	Determine the directory structure of a system	54
3.12	Manually run the file system checker and repair tool	55
3.13	View and modify file flags	56
3.14	Monitor the virtual memory system	58
4	Chapter Users and Accounts Management	59
4.1	Protect authentication data	59
4.2	Create, modify and remove user accounts	61
4.3	Create a system account	62
4.4	Control which files are copied to a new user's home directory during account creation	63
4.5	Change a password	64
4.6	Change the encryption algorithm used to encrypt the password database	66
4.7	Change a user's default shell	67
4.8	Lock a user account or reset a locked user account	68
4.9	Determine identity and group membership	69
4.10	Determine who is currently on the system or the last time a user was on the system	71
4.11	Enable accounting and view system usage statistics	74
5	Chapter Basic System Administration	75
5.1	Determine which process are consuming the most CPU	76
5.2	View and send signals to active processes	79
5.3	Use an rc.d script to determine if a service is running and start, restart or stop it as required	80
5.4	Configure a service to start at boot time	83
5.5	View and configure system hardware	84
5.6	View, load, or unload a kernel module	84
5.7	Modify a kernel parameter on the fly	85
5.8	View the status of a software RAID mirror or stripe	88
5.9	Configure system logging	91
5.10	Review log files to troubleshoot and monitor system behavior	91
5.11	Determine which MTA is being used on the system	92
5.12	Create or modify email aliases for Sendmail or Postfix	93

5.13	View the Sendmail or Postfix mail queue	93
5.14	Read mail on the local system	95
5.15	Understand basic printer troubleshooting	96
5.16	Halt, reboot, or bring the system to single-user mode	96
5.17	Recognize the difference between hard and soft limits and modify existing resource limits	99
5.18	Recognize common, possibly third-party, server configuration files	99
5.19	Configure the scripts that run periodically to perform various system maintenance tasks	101
5.20	Determine the last system boot time and the workload on the system	102
5.21	Monitor disk input/output	103
5.22	Deal with busy devices	105
5.23	Determine information regarding the operating system	105
5.24	Understand the advantages of using a BSD license	107
6	Chapter Network Administration	109
6.1	Determine the current TCP/IP settings on a system	110
6.2	Set a system's TCP/IP settings	112
6.3	Determine which TCP or UDP ports are open on a system	114
6.4	Verify the availability of a TCP/IP service	115
6.5	Query a DNS server	116
6.6	Determine who is responsible for a DNS zone	121
6.7	Change the order of name resolution	123
6.8	Convert a subnet mask between dotted decimal, hexadecimal or CIDR notation	123
6.9	Gather information using an IP address and subnet mask	125
6.10	Understand IPv6 address theory	127
6.11	Demonstrate basic tcpdump(1) skills	128
6.12	Manipulate ARP and neighbor discovery caches	128
6.13	Configure a system to use NTP	129
6.14	View and renew a DHCP lease	129
6.15	Recognize when and how to set or remove an interface alias	130
7	Chapter Basic Unix Skills	133
7.1	Demonstrate proficiency in using redirection, pipes and tees	134
7.2	Recognize, view and modify environment variables	134
7.3	Be familiar with the vi(1) editor	136
7.4	Determine if a file is a binary, text, or data file	137
7.5	Locate files and binaries on a system	139
7.6	Overcome command line length limitations	140
7.7	Find a file with a given set of attributes	140
7.8	Create a simple Bourne shell script	141
7.9	Find appropriate documentation	142
7.10	Recognize the different sections of the manual	143
7.11	Verify a file's message digest fingerprint (checksum)	145

Contents

7.12	Demonstrate familiarity with the default shell	145
7.13	Use job control	146
7.14	Demonstrate proficiency with regular expressions	149
7.15	Understand various “domain” contexts	150
7.16	Configure an action to be scheduled by cron(8)	152
Index		153

Introduction

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

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

TODO: put correct title here:

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.

Credits

This book was written by a community of BSD experts and fans who collaborated via a wiki website where anyone could contribute with writing, reviewing, proofreading and sharing valuable feedback.

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

- Yannick Cadin
- Grzegorz Czaplinski
- Ceri Davies
- Hubert Feyrer
- Mark Foster
- Kevin D. Kinsey
- Andreas Kuehl
- Cezary Morga

- Alex Nikiforov
- Jeff Quast
- Jeremy C. Reed
- Chris Silva
- Sean Swayze
- Ion-Mihai Tetcu
- Ivan Voras

Also a big thank you to Hiroki SATO and AllBSD.org for providing a server for hosting the book development website.

TODO: maybe mention software used in the creation of this project

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 ?? 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 ?? ??*

Reviewer: *name ?? ??*

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.

- 1.1
- 1.2
- 1.2
- 1.3
- 1.5
- 1.6
- 1.7
- 1.8
- 1.9
- 1.10

1.1 Recognize the installation program used by each operating system

Author: hubertf, David Stanford dthomas53_at_gmail_dot_com FreeBSD

Reviewer: David Stanford dthomas53_at_gmail_dot_com FreeBSD

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

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.

Introduction

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

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:

1.1. RECOGNIZE THE INSTALLATION PROGRAM USED BY EACH OPERATING SYSTEM

- NetBSD: 4.0_BETA, 4.1_BETA, ...
- FreeBSD: 6.2-BETA1, 6.2-BETA2, ...
- OpenBSD: 4.0-beta
- Release candidate 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: 6.1-SECURITY
 - OpenBSD:

Installer

NetBSD

Most NetBSD ports use the 'sysinst' installer, a few still provide the old script-based installer as an 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](ftp://NetBSD.org) (and its mirrors) in /pub/NetBSD, ISO images are in /pub/NetBSD/iso and daily snapshots of the various branches can 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](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's installer is a straight 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 architectures with every method of installation. Installing OpenBSD is usually done using either a floppy boot image, a bootable CD, booting across a network e.g. PXE on i386 (not available on all architectures). Only changes in the BIOS/Open Firmware will select what installer will be run.

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

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`

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 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 know which image to boot depending on the hardware.

Practice Exercises

More information

Release naming

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

http://www.freebsd.org/doc/en_US.ISO8859-1/articles/releng for FreeBSD

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

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: David Stanford dthomas53_at_gmail_dot_com FreeBSD

Reviewer: David Stanford dthomas53_at_gmail_dot_com FreeBSD

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

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.

Introduction

Binary vs. from-source

NetBSD

There are several ways to upgrade your NetBSD system. The easiest method is to boot the installation CD-ROM and follow the steps in `sysinst` for upgrading your system. But this may not be possible on all NetBSD platforms, therefore the best way is to recompile a full release from the sources and upgrade your system. When done, you need reboot into your new system.

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 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.

As an additional note, future versions of FreeBSD are also likely include a utility called CSup. CSup is nothing more than a rewrite of the current CVSup utility in the C programming language. CSup is expected to contain the same feature set as CVSup, but since it is to be rewritten in C, it will be much easier for future developers to maintain and for cross-platform compatibility.

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 system, 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/upgrade40.html>

The system can be built from source as described at <http://www.openbsd.org/faq/faq5.html> but this is for following the stable branch. Upgrading via source is NOT supported.

DragonFly BSD

XXX

Examples

Practice Exercises

More information

`make(1)` including the 'buildworld', 'installworld', and 'quickworld' and similar targets; `merge-master(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 ?? ??*

Reviewer: *name ?? ??*

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

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 precompiled binary and the advantages and disadvantages of compiling a binary from source.

Introduction

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

Examples

Practice Exercises

More information

Dragonfly and NetBSD provide pkgsrc/pkgtools/pkgchk, pkgsrc/pkgtools/pkg comp, make update and make replace; portupgrade, portsnap and cvsup 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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

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.

Introduction

Examples

Practice Exercises

More information

pkg_add(1), pkg_delete(1)

1.5 Recognize the available methods for compiling a customized binary

Author: *name ?? ??*

Reviewer: *name ?? ??*

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

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.

Introduction

Examples

Practice Exercises

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 ?? ??*

Reviewer: *name ?? ??*

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

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.

Introduction

Examples

Practice Exercises

More information

pkg_info(1)

1.7 Determine which software requires upgrading

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

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.

Introduction

Examples

Practice Exercises

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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

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.

Introduction

Examples

Practice Exercises

More information

Dragonfly and NetBSD provide pkgsrc/pkgtools/pkg_chk, pkgsrc/pkgtools/pkg_comp, 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 ?? ??*

Reviewer: *name ?? ??*

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

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.

Introduction

Examples

Practice Exercises

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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: Chris Silva racerx@makeworld.com FreeBSD

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.

Introduction

Examples

Practice Exercises

More information

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

2 Chapter Securing the Operating System

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

- 2.1
- 2.3
- 2.4
- 2.5
- 2.6
- 2.7
- 2.2
- 2.8
- 2.10
- 2.11
- 2.12
- 4.6
- 2.13
- 4.1

2.1 Determine the system's security level

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

The BSD kernels can limit – even from the superuser – changing immutable and append-only file flags, TODO. File flags are covered in 3.13 .

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

You can look at current secure level via sysctl:

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

TODO: list possible numbers and what the values mean for four BSDs.

TODO: document how to set via rc.conf for next boot

You can change this variable on the fly using sysctl. Using sysctl is covered in section 5.7 .

Examples

Practice Exercises

More information

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

2.2 Recognize basic recommended access methods

Author: Kevin D. Kinsey kdk_at_daleco_dot_biz FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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, possibly using the third-party `sudo` utility instead of `su(1)` and minimizing the use of the `wheel` group.

Introduction

Many BSD machines are used as “servers” in locations away from the system administrator. Since the earliest days of networking, methods have been developed to access one machine from a terminal or another machine. Access to remote machines, particularly those accessible from the public Internet, must be secured from access by unauthorized personnel. In addition, good security practice demands that we limit the number of people who know (or need to know) root’s password at all.

Use SSH instead of Telnet or rsh/rlogin

Telnet(1) was an early implementation of a “remote control” application. Another early program for remote administration was `rsh/rlogin`. While these are still valid for limited uses today, they should **never** be used to administrate a machine over a network, because all data in `telnet(1)` or `rsh/rlogin` is transmitted without encryption, or in “plain text”. Anyone with the ability to run a “packet sniffing” program could read your password and all other transactions between you and your system if you use `telnet` or `rsh/rlogin`. Except in limited circumstances, such as in a locked-down facility with hungry guard dogs and a single armored cable which no one else can access between you and the server, which is in plain sight and surrounded by an electric fence, don’t use **rsh**, **rlogin** or **telnet**, OK? Use SSH instead!

SSH consists of two programs developed by the OpenBSD team, `ssh(1)` and `sshd(8)`. **ssh** is a client program which provides terminal access (and other features) to another machine. **sshd** is a server daemon which runs on the “other machine”. Together, these two programs provide all the features (and more) of `rsh/rlogin` and `telnet`, and also provide you with more security — using SSH ensures your sessions are protected by strong encryption.

More about SSH appears in the following paragraphs and the next three sections.

Deny Root Logins

Since the “root” account can do anything at all to a system, good security practice requires that root logins be restricted to the physical console of the machine itself, and, usually, only in dire circumstances. Both local and remote (SSH!!) logins should be as a “normal” user, and normal users who need to run programs that require root’s authority should use an alternate method to “get root”.

Reasons for this are many, vary in importance to various individuals, and are frequently discussed in newsgroups, e-mail lists, internet forums, and other materials. Briefly, here are some things to think about:

*The “onion” principle of security: **if** someone compromises remote access, is it better that they get limited capabilities, or root privileges? What if they have access to the machine’s console and have seen someone login a few times? If so, what if their memory is photographic?

*The “oh my!” principle of sanity: someday, sometime, **someone** will have a bad day. If **someone** is logged in as root when that happens, the possibility for much more damage exists. The classic example of this argument is in the use of `rm(1)`, but plenty of Bad Stuff(tm) can happen with `chown(8)`, `chmod(1)`, and other programs. File permissions are one protection from these types of errors, but permissions don’t do any good if you’re logged in as root when a mistake occurs.

*The “oh no!” principle of spitefulness: in real life and real work, people get fired, canned, removed, terminated, and, sometimes, angry about it. If your superior fires a team member on a day when you’re not there to turn off their access to the system, and they know the root password, there is a chance, no matter how “nice” that person may seem, of them manifesting a streak of vindictiveness prior to cleaning out his/her desk or cubicle.

For more on this subject, keep reading. To set `sshd` to deny root logins, see the next section 2.3 .

su(1), the “wheel” group, and sudo(8)

Since we want all logins to be by “normal” users, how will we accomplish tasks like software installation, modifying system-wide configuration files, restarting daemons, or rebooting the system?

Traditionally, anyone who has a need to do this sort of chore is a member of the “wheel” group. Only members of this group are allowed to use the `su(1)` program to gain root privileges. In order to “get root” with `su` , you must know the root password.

More recently, many system administrators use `sudo(8)` to allow root privileges. Sudo was developed at SUNY/Buffalo in 1980 and is currently maintained by OpenBSD; it is available in their default install, or via the ports/packages system on other BSD flavors.

sudo is highly configurable, contains extra notification and security mechanisms, and can even be configured to allow groups of users access to a limited set of commands (for example, to allow the webmaster to restart the `www` daemon, but not restart the system, or to allow normal users to mount and umount media but not erase them, etc.)

For more information about sudo, install it and read the manpage, or visit <http://www.sudo.ws/>

Examples

Connecting to a remote machine:

```
$telnet myserver.example.com
Trying 6.7.8.9...
telnet: connect to address 6.7.8.9: Connection refused
Trying 6.7.8.9...
telnet: connect to address 6.7.8.9: Connection refused
telnet: Unable to connect to remote host
```

Good! Telnetd isn't enabled on your remote machine!

```
$ssh me@myserver.example.com
```

You may see a banner message at this point. If you are using password-based authentication, you will be prompted:

```
Password:
```

Enter your passphrase and you will see information from various files and programs; among them, `last(1)`, `uname(1)`, and `/etc/motd`:

```
Last login: Wed Jan 10 11:29:26 2007 from 9.8.7.6
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
    The Regents of the University of California. All rights reserved.
FreeBSD 6.0-RELEASE-p13 (GENERIC) #3: Thu Sep 28 20:02:55 CDT 2006
Welcome to FreeBSD!
```

If you use key-based authentication, no password is required. Your shell resource scripts will be called, your environment set up, and you should receive a shell prompt:

```
$
```

You are now “logged in”.

a word about “user@host” syntax

In the example above we used “user@hostname” with `ssh` to specify the remote account and remote server to connect to. Other methods to specify the username include editing the local `/.ssh/config` file or using the “-l” flag when calling `ssh`. If you simply “ssh hostname”, `ssh` will attempt a login using the name given by `id(1)`.

Using su and sudo to gain root privileges:

Let's do a chore that requires root's power to accomplish. Suppose we want to enable `natd(8)`, the “Network Address Translation Daemon”, at the next bootup:

```
$echo 'natd_enable="YES"' >> /etc/rc.conf
/etc/rc.conf: Permission denied.
```

As a “normal” user, you can't append (or edit) `/etc/rc.conf`; it's owned by root and “chmodded” to 744. So, use **su** :

```
$su
Password:
```

If you are in the wheel group, you are prompted for the root password (if you are not in the wheel group, you simply get “Su: sorry”). If you enter root's password correctly, you get a new shell with root's prompt and environment. Now try the command:

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

It succeeds silently, and the directive is added to the rc.conf file.

If sudo(8) is on your system, you must edit the sudoers file as root before attempting to use **sudo**. Use the visudo(8) program to invoke your \$EDITOR on the sudoers file, and set permissions appropriately. After this, allowed users should be able to use *sudo* as a preface to their “root level” commands, thus:

```
$sudo shutdown -r now
```

Depending on sudo’s configuration, the command may succeed silently, or the user may be prompted for *their own* password (instead of the root password). Being configured to run sudo without a password prompt is very convenient, but the password prompt is also a great behavior for at least two reasons: 1] it means that the administrators don’t have to actually know the root password to do “important” tasks, thereby protecting password integrity, and 2] the user is reminded that he/she is about to do something using root privileges, and hopefully, to look closely at the command they just typed before entering the password.

Using the first example above, something strange seems to happen when prefaced by **sudo** :

```
$sudo echo 'natd_enable="YES"' >> /etc/rc.conf
/etc/rc.conf: Permission denied.
```

This is because **sudo** executes the echo command, but stops at the redirect, so the “normal user” is attempting to redirect to the protected file. It’s a tad tricky (watch the quotes!), but here’s a way around this problem:

```
$sudo sh -c echo "'natd_enable="YES"' >> /etc/rc.conf"
```

We have sudo call a single command, “sh -c”, as root, and this spawned /bin/sh process takes care of the echo call and the redirection, all with root credentials. You could have similar problems if you try something like “sudo foo && bar”; you would need to call sudo for both commands, or do some quoting magic.

Practice Exercises

1. Attempt to use telnet(1) to a remote machine. If you get a password prompt, disconnect immediately and call the machine’s administrator as soon as possible.
2. Use ssh(1) to connect to a remote machine.
3. If you are the system administrator/owner or a member of the wheel group on a system, login as a normal user and use “su” to get root. Edit /etc/rc.conf and add the line:
This line added by *me* on *date*
4. If you are the system administrator/owner or a member of the wheel group, install sudo and configure it to allow you to use it. Call “sudo \$EDITOR /etc/rc.conf” at your prompt, then remove the line you added in exercise 3.

More information

ttys(5), sshd_config(5), ftpusers(5); the (possibly third-party, depending on your BSD flavor) utility sudo ?? which includes visudo(8), suedit and sudoers(5);

2.3 Configure an SSH server according to a set of requirements

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Setting up SSH keys is covered in section 2.4 .

Examples

Practice Exercises

More information

sshd_config(5)

2.4 Configure an SSH server to use a key pair for authentication

Author: Kevin D. Kinsey kdk_at_daleco_dot_biz FreeBSD

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

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.

Introduction

(**Note:** Some basics on configuring the SSH server are covered in section 2.3 .)

Passwords and passphrases are helpful in maintaining a trustworthy and secure system. However, passwords can be guessed, leaked, or inadvertently disclosed in several ways.

Using SSH authentication based on “Public key encryption” can eliminate some of these potential vulnerabilities. “Key-based” authentication utilizes a “public key” and “private key” pair which are strongly encrypted; the private key is kept secret, while the public key may be widely disseminated (as in PGP signatures for e-mail). In SSH key-based authentication, the public key is known to the SSH server(s) that you wish to connect to, and the private key is known only by the SSH client program you are using.

While the keys are mathematically related, it is currently impossible, or at least extremely infeasible, to calculate the private key from the contents of the public key. If password authentication is disabled on the SSH server, then only SSH clients (programs) which provide the correct key are allowed access.

To configure an SSH server to use key-based authentication, edit the server’s configuration file `/etc/ssh/sshd_config` and restart `sshd(8)` (see the previous section 2.3). You must also copy your public key to the `/.ssh/authorized_keys` file on the remote machine. To create the key pair, the program `ssh-keygen(1)` is used.

Key Generation Algorithms

Two “standard” algorithms exist for the encryption of public-private key pairs, RSA and DSA. RSA was developed in the late 1970’s by MIT researchers. The DSA algorithm was developed by the US Government in the early 1990’s. While some researchers believe that RSA encryption will become more feasible to decipher within a few years, many authorities still believe RSA encrypted key-pairs to be secure, as long as they were created with an up-to-date mechanism and the key itself is long enough. In US government work (and probably in many other circumstances/organizations) DSA keys are a required standard. `ssh-keygen(1)` can generate both RSA and DSA keys for use in key-based SSH authentication.

The SSH server daemon `sshd(8)` utilizes RSA keys when SSH protocol version 1 is used, and can use both RSA and DSA keys for protocol version 2.

Choosing an Appropriate Bit Size

The minimum bit size for an RSA-key is 768 bits; the default is 2048 bits. The default should be good enough for most applications. Some researchers believe that new computer systems may be able to break fairly small RSA keys within 20-30 years (or perhaps even more quickly), but most believe that keys as large as 1024 for RSA encryption will be safe for a very long time (“the suns going to burn out before current desktop technologies can factor it”, writes one armchair analyst). DSA keys **must** be 1024 bits according to the specification.

You can specify the bit size for RSA key generation by using the “-b” option to `ssh-keygen` .

Providing a Seed

Providing a Passphrase

Normally, both your public key and private key are stored locally in the directory `/etc/ssh`. In order to keep the private key secure from others, it is owned by the user and `chmodded 600`. However, the file is plaintext, so is possible that others might see your private key (for example, if they have root privileges). For this reason, it is often a good idea to encrypt the private key.

`ssh-keygen` can, either at the time of key creation or a later time, encrypt a private key (RSA or DSA) using the 3DES algorithm and a passphrase. It can also change a encrypted key's passphrase (assuming you can provide the current passphrase). Passphrases should be 10-30 characters in length and may include any characters you want, including whitespace. Plain words or sentences should be avoided — use something somewhat unusual!

Once your private key is encrypted, you must provide the passphrase each time you use it for authentication. A lost key passphrase can be a significant problem — a new key pair must generated and the public key copied to the SSH server. This could be fairly simple (for example, if you have console access or another account with access to the SSH server), or very challenging (for example, if your very remote, unattended SSH server will only allow key-based authentication for a single account).

Examples

Practice Exercises

More information

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

2.5 Preserve existing SSH host keys during a system upgrade

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

`/etc/ssh/ssh_host_key`

2.6 Recognize alternate authentication mechanisms

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.)

Introduction

Examples

Practice Exercises

More information

2.7 Recognize alternate authorization schemes

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Examples

Practice Exercises

More information

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

2.8 Recognize BSD firewalls and rulesets

Author: *name ?? ??*

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

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.

Introduction

Each BSD comes with at least one built-in firewall. NetBSD provides IP Filter (IPF) and PF (from OpenBSD). FreeBSD has its own IPFW, IP Filter, and PF. OpenBSD includes its own PF. And DragonFly has IPFW, IP Filter, and PF.

IP Filter (IPF)

TODO: write two or three sentences just to summarize it
TODO: show one or two examples how to detect if it is available and if it is enabled
TODO: point to default startup script for enabling
TODO: point to location of default configurations
TODO: point to included documentation and examples
TODO: show how to turn on and turn off (and mention issues with that)
TODO: show how to view loaded ruleset
TODO: show a very brief example (three rules/lines) and use same functionality for all three firewalls here

IPFW

TODO: write two or three sentences just to summarize it
TODO: show one or two examples how to detect if it is available and if it is enabled
TODO: point to default startup script for enabling
TODO: point to location of default configurations
TODO: point to included documentation and examples
TODO: show how to turn on and turn off (and mention issues with that)
TODO: show

how to view loaded ruleset TODO: show a very brief example (three rules/lines) and use same functionality for all three firewalls here

PF

TODO: write two or three sentences just to summarize it TODO: show one or two examples how to detect if it is available and if it is enabled TODO: point to default startup script for enabling TODO: point to location of default configurations TODO: point to included documentation and examples TODO: show how to turn on and turn off (and mention issues with that) TODO: show how to view loaded ruleset TODO: show a very brief example (three rules/lines) and use same functionality for all three firewalls here

Practice Exercises

More information

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

2.9 Recognize the BSD utilities that shape traffic or control bandwidth

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

TODO: do not teach how to use these tools; just say what is available, basically what they do and where to get more information. TODO: because this is beyond BSDA “associate”

Introduction

Examples

Practice Exercises

More information

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

2.10 Recognize BSD mechanisms for encrypting devices

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

TODO: do not go into detail on this advanced topic ... just briefly tell what is available and give a very brief usage example if you want –reed

Introduction

Examples

Practice Exercises

More information

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

2.11 Recognize methods for verifying the validity of binaries

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Examples

Practice Exercises

More information

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

2.12 Recognize the BSD methods for restraining a service

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

TODO: this does not teach how to setup jail or xen or systrace. Just quickly explain what is available.

TODO: show quick example or explanation of setting up chroot and using chroot

Introduction

Examples

Practice Exercises

More information

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

2.13 Modify the system banner

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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: FreeBSD and NetBSD's `login.conf` allows definition of 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??

Examples

Practice Exercises

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

More information

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

3 Chapter Files, Filesystems and Disks

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: Yannick Cadin 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.

- 3.1
- 3.2
- 3.3
- 3.4
- 3.5
- 3.5
- 3.7
- 3.8
- 3.9
- 3.10
- 3.11
- 3.12
- 3.13
- 3.14

3.1 Mount or unmount local filesystems

Author: ?? andreas dot kuehl at clicktivities dot net FreeBSD

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

Concept

Be familiar with all aspects of mounting and unmounting local filesystems including: how to mount/unmount 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.

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 hw have to cover:

1. mount
2. unmount
3. /dev/ad0s1a
4. hint to fsck
5. /etc/exports
6. mount -a
7. errors at mount -a
8. /etc/fstab ...

Examples

Mount partition e on the first slice of the primary master IDE hard disk at /usr :

```
# mount /dev/ad0s1e /usr
```

Same thing, SCSI disk

```
# mount /dev/da0s1e /usr
```

Mount all filesystems listed in /etc/fstab :


```
# mount -a
```

Note that filesystems marked “noauto” in `/etc/fstab` would not be mounted by this command.
Mount optical media in the CD-ROM drive as listed in `/etc/fstab` :

```
# mount /cdrom
```

Mount a floppy disk to `/mnt` :

```
# mount -t msdos /dev/fd0 /mnt
```

Mount a FAT32 formatted USB “key” on `/mnt` :

```
# mount_msdosfs /dev/da0s1 /mnt
```

Note that USB devices are assigned SCSI device names; if `usbd(8)` is running, the system log file `/var/log/messages` should show the name of the device within a few seconds after the device is inserted. Also, note that “`mount -t msdos`” and “`mount_msdosfs`” actually accomplish the same operation.

Practice Exercises

More information

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

3.2 Configure data to be available through NFS

Author: *name ?? ??*

Reviewer: Cezary Morga `cm@therek.net` FreeBSD

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

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: Kevin D. Kinsey kdk_at_daleco_dot_biz FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

The UNIX paradigm “everything is a file” means that almost any device can be mounted to almost any location on the filesystem hierarchy.

To see all devices currently mounted on the filesystem, call `mount(8)` with no arguments. At boot time, `init` mounts devices as shown in the file `/etc/fstab`, which may be edited by the superuser to add or remove additional boot-time mounts and change mount parameters.

Examples

```
$ cat /etc/fstab
# Device                Mountpoint      FStype  Options      Dump    Pass#
/dev/ad0s1b             none            swap    sw           0       0
/dev/ad0s1a             /               ufs     rw           1       1
/dev/ad0s1e             /usr            ufs     rw           2       2
/dev/ad0s1d             /var            ufs     rw           2       2
/dev/acd0                /cdrom          cd9660  ro,noauto    0       0
```

A typical 'fstab' from a FreeBSD 6.x machine. Note that the “noauto” option for the entry `/dev/acd0` means that `init(8)` will not mount the cdrom during bootup, but the CD can be mounted with “`mount /cdrom`” by a user with the appropriate privileges. Here is the output of `mount(8)` for the same system:

```
$mount
/dev/ad0s1a on / (ufs, local, soft-updates)
devfs on /dev (devfs, local)
/dev/ad0s1e on /usr (ufs, NFS exported, local, soft-updates)
/dev/ad0s1d on /var (ufs, local, soft-updates)
devfs on /var/named/dev (devfs, local)
```

3.3. DETERMINE WHICH FILESYSTEMS ARE CURRENTLY MOUNTED AND WHICH WILL BE MOUNTED AT SYSTEM BOOT

Mount(8) here gives a couple of details not visible in the system 'fstab' — one is the existence of devfs(5), the “device file system”, which occurs twice here because this system runs named(8) in a “sandbox” or chrooted environment. The other is the fact that /usr is exported as a Network File System. In this case, you might wish to call showmount(8) to see if anyone is connected to your exported file system before you umount(8) it or call shutdown(8). (Network mounts are introduced in section 3.2.)

The 'df' tool can also show mounted filesystems, for example:

```
$ df -m
```

Filesystem	1M-blocks	Used	Avail	Capacity	Mounted on
/dev/wd0a	1008	23	934	2%	/
/dev/wd0f	4032	10	3820	0%	/var
/dev/wd0e	15121	1541	12823	10%	/usr
/dev/wd0g	10081	500	9076	5%	/home
kernfs	0	0	0	100%	/kern

The -t switch for df can be used to specify or exclude some filesystems. To exclude, prefix the filesystem name with “no”, such as “noffs” or “noprocfs”. For example, on DragonFly and on NetBSD:

```
$ df -t nfs
```

Filesystem	1K-blocks	Used	Avail	Capacity	Mounted on
office:/pub	20644846	858522	18134738	5%	/pub

```
$ df -t noffs
```

Filesystem	512-blocks	Used	Avail	Capacity	Mounted on
kernfs	2	2	0	100%	/kern

More details on using df are covered in section 3.4.

Practice Exercises

1. Compare the output of mount(8) on your system(s) with the output shown above.
2. If the machine isn't “mission-critical”, try removing the “noauto” option above and re-booting the system with the optical drive empty. What do you predict will happen? What actually happens? (Note that you might want to have a backup copy of /etc/fstab stored somewhere in the root of your filesystem if you try this.)

More information

mount(1), df(1), fstab(5)

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

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Examples

Practice Exercises

More information

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

3.5 Create and view symbolic or hard links

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

3.6 View ACLs

Author: Grzegorz Czapliński gregory at systemics dot pl FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

ACLs provide an extended set of permissions for a file or directory. These permissions can be used in addition to the conventional UNIX permissions for files and directories. Standard UNIX file permissions (covered in section 3.7) provide read, write and execute access to three user classes:

- file owner
- file group
- others

ACLs are used to provide greater data access control for each file or directory. They enable you to define permissions for specific users and groups.

Every ACL has the following syntax:

```
[ACL tag]:[ACL qualifier]:[Access permissions]
```

ACL tag is a scope of the file permissions to the owner, group, others, specific users, specific groups or ACL's mask.

The ACL qualifier field describes the user or group associated with the ACL entry. It might be UID or user's name, GID or group's name, or empty.

Access permissions are the effective permissions for [ACL tag] and are specified as:

- r - read
- w - write
- x - execute

Entry types:

u::perm permissions for the file owner g::perm permissions for the file group o::perm permissions for the others u:UID:perm permissions for the specific user identified by UID u:username:perm permissions for the specific user identified by username g:GID:perm

permissions for the specific group identified by GID g:groupname:perm permissions for the specific group identified by groupname m::perm maximum effective permissions allowed for specific users or groups.

The mask does not set the permissions for the file owner or others. It is used as a

ACLs are part of UFS2 filesystem shipped with FreeBSD 5.0-RELEASE as an option or FreeBSD 5.1-RELEASE as the default filesystem during the installation. To check which filesystem you have on your system type:

```
# dumpfs /tmp | head -1
magic 19540119 (UFS2) time Fri Aug 15 19:23:30 2003
```

You must have ACL support compiled into the kernel too.

Add:

```
options          UFS_ACL          #Support for access control lists
```

to your kernel config compile and install a new kernel according to the instructions in the June 2003 Answerman column.

To enable ACLs on a partition, after newfs(1)'ing it issue the commands:

```
# tuneufs -a enable /dev/dals1e
# mount /dev/dals1e /mountpoint
# mount | grep acl
/dev/dals1e on /mountpoint (ufs, local, soft-updates, acls)
```

This indicates that soft-updates and acls are enabled on the /dev/dals1e partition mounted under /mountpoint.

The other way to check if ACLs are enabled is to use tuneufs(1) command:

```
# tuneufs -p /dev/dals1e
tuneufs: ACLs: (-a)          enabled
tuneufs: MAC multilabel: (-l) disabled
tuneufs: soft updates: (-n)  enabled
tuneufs: maximum blocks per file in a cylinder group: (-e) 2048
tuneufs: average file size: (-f) 16384
tuneufs: average number of files in a directory: (-s) 64
tuneufs: minimum percentage of free space: (-m) 8%
tuneufs: optimization preference: (-o) time
tuneufs: volume label: (-L)
```

Before I show some examples please read the manpage for getfacl(1). The commands and their output below are separated by one empty line for clarity. On my test system I have a user called acl and he belongs to wheel group. When you see touch(1) command in an example, that means I recreated a file after it was removed.

Create an empty file:

```
% umask 027
% touch file.txt
% ls -l file.txt
-rw-r--- 1 acl wheel 0 Aug 5 22:35 file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
group::r-
other::--
```

The file.txt is a normal file without any ACL permissions set yet.

TODO: need to clean this part up (and others with setfacl) and just show your own possible examples without using setfacl in this doc. TODO: it would be good for beginning admin to know that setfacl exists and what it is, but according to the BSD Certification Group, TODO: this doesn't need to target modifying ACLs.

```
% ls -l file.txt
-rw-rw---+ 1 acl wheel 0 Aug 5 22:41 file.txt
```

The little “+” at the end of access rights column indicates that the file has ACL set.

```
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-
group::r-
mask::rw-
other::--
```

This command shows that owner has read/write access, group has read access, and user gregory has read/write access. I have to point now that the mask indicates the maximum permissions for user gregory.

If the command was (set the mask - “m::r”):

TODO: remove setfacl from this beginning article:

```
% setfacl -m u::rw,g::r,u:gregory:rw,m::r file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-      # effective: r-
```

```
group::r-
mask::r-
other::--
```

user gregory would have read/write access, but the mask would downgrade the effective access rights to read only.

There is an “-M” switch that is used to set and modify the ACL entries. The information about actual ACLs are kept in a file (in this example acls.txt).

```
% touch file.txt
```

Create acls.txt file which looks like:

```
% cat acls.txt
u:bin:rwx
% setfacl -M acls.txt file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:bin:rwx
group::r-
mask::rwx
other::--
```

In the last example ACL entry for user bin was specified in a file acls.txt. Recalculating an ACL mask

The ACLs look as above (the last getfacl(1) command), issue a command:

```
% setfacl -m u::rw,g::r,u:bin:rw file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:bin:rw-
user:gregory:rw-
group::r-
mask::rw-
other::--
```

Now, users gregory and bin have read/write access, and the mask has been “group” ACL entries in the resulting ACL.

If the last command was:


```
% setfacl -n -m u::rw,g::r,u:bin:rw file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:bin:rw-      # effective: r-
user:gregory:rw-  # effective: r-
group::r-
mask::r-
other:--
```

the mask would not get recalculated (switch -n). Effective rights for users gregory and bin would be read only. Deleting an ACL

To delete an ACL entry for user bin do:

```
% setfacl -n -x u:bin:rw file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-  # effective: r-
group::r-
mask::r-
other:--
```

The entry for user bin was deleted. If you want the mask not to get recalculated, remember to use the “-n” switch. If you didn’t use it, the mask would be read/write now, effectively changing permissions for user gregory to read/write.

To remove permanently ACL from a file issue:

```
% setfacl -bn file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
group::r-
other:--
% ls -l file.txt
-rw-r--- 1 acl wheel 0 Aug 5 23:08 file.txt
```

Compare the above with that:

```
% setfacl -b file.txt
```

```
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
group::r-
mask::r-
other::--
```

In the next example, `setfacl(1)` command is able to change permissions for all user classes - owner, group, others.

```
% umask 027
% touch file.txt
% ls -l file.txt
-rw-r--- 1 acl wheel 0 Aug 5 23:13 file.txt
% setfacl -m u::rw,g::r,o::r,u:gregory:rw file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-
group::r-
mask::rw-
other::r-
alphax% ls -l file.txt
-rw-rw-r+ 1 acl wheel 0 Aug 5 23:12 file.txt
```

More interesting example:

```
% touch file.txt
% ls -l
total 0
-rw-r--- 1 acl wheel 0 Aug 5 23:24 file.txt
% chmod 660 file.txt
% ls -l
total 0
-rw-rw--- 1 acl wheel 0 Aug 5 23:24 file.txt
% setfacl -m u::rw,g::r,o::r,u:gregory:rw file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
```

```

user:gregory:rw-
group::r-
mask::rw-
other::r-
% ls -l
total 2
-rw-rw-r--+ 1 acl  wheel  0 Aug  5 23:25 file.txt
% chmod 644 file.txt
% ls -l
total 2
-rw-r-r--+ 1 acl  wheel  0 Aug  5 23:25 file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-      # effective: r-
group::r-
mask::r-
other::r-

```

The last setfacl(1) command set the access rights as follows:

```

user::rw-
user:gregory:rw-
group::r-
mask::rw-
other::r-

```

Then I changed explicitly access rights with chmod(1) command:

```
% chmod 644 file.txt
```

and the access rights reappeared as:

```

user::rw-
user:gregory:rw-      # effective: r-
group::r-
mask::r-
other::r-

```

Note, the mask is closely associated with group access rights. Changing Unix access rights with chmod(1), you also change the mask value.

Consider this scenario:

```

% touch file.txt
% setfacl -m u::rw,g::rw,o::r,u:gregory:rw file.txt

```

```
% ls -l file.txt
-rw-rw-r--+ 1 acl wheel 0 Aug 6 20:19 file.txt
% setfacl -m m::r file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-      # effective: r-
group::rw-           # effective: r-
mask::r-
other::r-
% ls -l file.txt
-rw-r-r--+ 1 acl wheel 0 Aug 6 20:20 file.txt
```

Changing the mask value, does change group access rights.

If you see a file with a magic “+” at the end of access rights column, check it with `getfacl(1)`.

Copying ACL entries

```
% touch file.txt
% setfacl -m u::rw,g::r,u:gregory:rw file.txt
% getfacl file.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-
group::rw-
mask::rw-
other::r-
% getfacl file.txt | setfacl -b -n -M - file1.txt
% getfacl file.txt file1.txt
#file:file.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-
group::rw-
mask::rw-
other::r-
#file:file1.txt
#owner:1009
#group:0
user::rw-
user:gregory:rw-
```

```
group::rw-
mask::rw-
other::r-
```

Creating default ACLs

Default ACL entries provide a way to propagate ACL information automatically to files and directories. New files and directories inherit ACL information from their parent directory if that parent has an ACL that contains default entries. You can set default ACL entries only on directories.

Example:

```
% umask 027
% mkdir dir
% ls -l
total 2
drwxr-x-- 2 acl wheel 512 Aug 6 11:50 dir
% getfacl dir
#file:dir
#owner:1009
#group:0
user::rwx
group::r-x
other:--
```

Before you set any default ACL entries for users or groups, you must set default ACL entries for owner, group, other, and ACL mask.

Consider this:

```
% setfacl -m u::rwx,m::rwx,g::rx,o::rx dir
% getfacl dir
#file:dir
#owner:1009
#group:0
user::rwx
group::r-x
mask::rwx
other::r-x
% setfacl -dm u:gregory:rwx,m::rwx dir
setfacl: acl_set_file() failed for dir: Invalid argument
```

The correct order is:

```
% setfacl -dm u::rwx,m::rwx,g::rx,o::rx dir
```

1. Set default ACL entries for directory owner, group, others and the mask.

```
% getfacl -d dir #file:dir #owner:1009 #group:0 user::rwx group::r-x mask::rwx other::r-x
```

To view default ACLs issue `getfacl(1)` with the “-d” switch.

```
% setfacl -dm u:gregory:rwX,m::rwX dir
```

1. Set default ACL entry for user gregory.

To see the effect of default ACLs on subdirectories issue the following commands:

```
% mkdir dir/subdir
% getfacl -d dir
#file:dir
#owner:1009
#group:0
user::rwx
user:gregory:rwx
group::r-x
mask::rwx
other::r-x
% getfacl -d dir/subdir
#file:dir/subdir
#owner:1009
#group:0
user::rwx
user:gregory:rwx
group::r-x
mask::rwx
other::r-x
```

The `subdir` directory successfully inherited default ACL entries from its parent. Suppose, you want to set default ACL entries for additional user `bin`:

```
% setfacl -dm u:bin:rwX,m::rwX dir
% getfacl -d dir
#file:dir
#owner:1009
#group:0
user::rwx
user:bin:rwx
user:gregory:rwx
group::r-x
mask::rwx
other::r-x
% getfacl -d dir/subdir
#file:dir/subdir
#owner:1009
#group:0
```

```

user::rwx
user:gregory:rwx
group::r-x
mask::rwx
other::r-x

```

That new default ACL entries for additional user bin are not visible on dir/subdir as the directory was created before the ACL entry for bin was set.

To see the effect of default ACLs on files, create a file beneath the dir directory:

```

% touch dir/file.txt
% ls -l dir/file.txt
-rw-r---+ 1 acl wheel 0 Aug 6 12:14 dir/file.txt
% getfacl dir/file.txt
#file:dir/file.txt
#owner:1009
#group:0
user::rw-
user:bin:rwx          # effective: r-
user:gregory:rwx     # effective: r-
group::r-x           # effective: r-
mask::r-
other:--

```

The `setfacl(1)` manual states: “Currently only directories may have default ACL’s. Deleting default ACLs

To delete default ACLs on directories, use `setfacl(1)` with the “-k” switch:

```

% setfacl -k dir
% getfacl -d dir
#file:dir
#owner:1009
#group:0
% getfacl dir
#file:dir
#owner:1009
#group:0
user::rwx
group::r-x
other:--

```

To delete a default ACL entry for user bin do:

```

% mkdir dir
% setfacl -dm u::rwx,m::rwx,g::rx,o::rx dir
% setfacl -dm u:gregory:rwx,u:bin:rwx,m::rwx dir

```

```
% getfacl -d dir
#file:dir
#owner:1009
#group:0
user::rwx
user:bin:rwx
user:gregory:rwx
group::r-x
mask::rwx
other::r-x
```

Create acls.txt file which looks like:

```
% cat acls.txt
u:bin:rwx
% setfacl -dX acls.txt dir
alpha% getfacl -d dir
#file:dir
#owner:1009
#group:0
user::rwx
user:gregory:rwx
group::r-x
mask::rwx
other::r-x
```

or simply type:

```
% setfacl -d -x u:bin:rwx dir
```

Things to remember

setfacl(1) always recalculates the ACL mask to allow maximum effective permissions for every ACL entry, unless the “-n” switch is used.

If you use the chmod(1) command to change the file group owner permissions on a file with ACL entries, both the file group owner permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions may change the effective permissions for additional users and groups who have ACL entries on the file.

Examples

Practice Exercises

More information

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

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

Author: Ivan Voras IvanVoras FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

An administrator is expected to have a thorough understanding of traditional Unix permissions including: how to view and modify permissions (i.e. “mode bits”), 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.

Introduction

File ownerships and mode bits are the single most important file system security feature in unix systems. Each file and directory has three attributes attached:

- User ID (uid)
- Group ID (gid)
- File mode bits

User and group IDs are simple numeric identifiers taken from /etc/passwd and /etc/group (but it’s perfectly valid, though not useful, to use a uid or gid not present in the system). File mode bits describe what permissions the user and the members of this group have on a particular file. In addition to those, there are special additional bits describing permissions all other users on the system have. The set of permissions is:

- r : read (user can read the file, or list a directory)
- w : write (user can write to the file, or create entries in the directory)
- x : execute (user can execute the file, or make the directory his current working directory)

Since the permissions form a bitmask, each has it’s numerical value. To make using numerical values of mode bits easier, they are usually written in octal notation (hexadecimal is not used because the number of mode bits is low enough):

- r : 04
- w : 02

- x : 01

Each of the above numbers is prefixed with 0 because that's how they are distinguished from decimal and hexadecimal numbers. To make a complex permissions these numbers are added together. For example, to form a rw permission (reading and writing is allowed), the correct number is 04+02=06.

To specify a compound permission which describes all mode bits for user, group and others, three digits are used (four with the 0 prefix). The first digit describes permission of the uid user, the second of the users in gid group and the third those of all other users. A common permission is 0644, which allows the owner to read and write the file, and enables all other users to just read the file.

The command to set mode bits is `chmod`.

TODO: mention discretionary control

Examples

Practice Exercises

More information

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

3.8 Modify a file's owner or group

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

A file's ownership can be changed by using the `chown` and `chgrp` tools. The change ownership tool can be used to change a file's user ID by using a username or user ID as an argument. A filename (or filenames) is the final argument.

The following example shows the current ownership of a file and then changes it to another user (by using a symbolic name and not a numeric ID):

```
$ ls -l math2.pl
-rwxr-xr-x 1 reed wheel 734 Feb 14 2006 math2.pl
```

```
$ chown austin math2.pl
$ ls -l math2.pl
-rwxr-xr-x 1 austin wheel 734 Feb 14 2006 math2.pl
```

The user ID can only be changed by the superuser (root), because a normal user should not be able to hide their data or bypass file system quotas by changing ownership. (TODO: any other reasons?) A file's group ID can be changed by any user to a group that they are a member of. (See section 4.9 about group membership.)

The chown utility can also change the file's group ID by prefixing a group name or group ID with a colon (:).

TODO: show example using numeric ID instead of symbolic name and ls -l -n

TODO: show example of using chown to change group

TODO: -R switch

TODO: point to section about file ownership attributes

TODO: document chgrp and show example

TODO: from concept "In addition, be aware of the importance of verifying one's own identity before creating files."

TODO: after mentioning -R, mention mtree can be used to reset file onwerships based on a specification. show brief example maybe? or point to other section?

Examples

Practice Exercises

More information

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

TODO: why "su" here? What is the context?

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

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

The common command-line backup utilities are tar, cpio, pax, cp, dd, and dump/restore. (Details on using the dd, dump, and restore utilities are covered in the following section 3.10 .)

TODO: explain differences and similarities between tools.

TODO: basic usage of tar with examples

TODO: basic usage of cpio with examples

TODO: basic usage of pax with examples

TODO: basic usage of using cp for backups with examples

Also the cpdup program is included with DragonFly and is available via NetBSD pkgsrc and FreeBSD ports packages collections (TODO: not in openbsd – anyone want to package it?)

TODO: if there is room, quickly list other third-party backup tools (but not examples or usage details)?

Common device names for tape drivers include: st, the SCSI and ATAPI tape driver; sa, the SCSI Sequential Access device driver.

Note it is recommended that the raw interface (not block) is used, such as /dev/rst0. (TODO: maybe the recommendation is not applicable for FreeBSD or other??) (TODO: mention no-rewind like nrst0 or nsa0? , eject on close like erst0 or esa0??)

Examples

Practice Exercises

More information

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

3.10 Backup and restore a file system

Author: Kevin D. Kinsey kdk_at_daleco_dot_biz FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Type “data loss statistics” into a search engine and you will receive a wide variety of results. Here are some current claims:

“6% of all PCs will suffer an episode of data loss in any given year” “U.S. businesses loose [sic] over \$12 billion per year because of data loss.” “60% of companies that lose their data close down within 6 months of the disaster.”

While it is exceptionally difficult to verify claims like these, one thing is certain: if *your* system’s data is lost, you can find yourself in a lot of trouble. It could simply be that you have to spend time doing a lot of extra work to restore lost configurations and files. It could be that you have to pay significant overtime to data entry personnel to restore data to a database. Your company could be adversely affected ... you might even lose your job.

One thing is fairly certain, even if we “manufacture” the claim: you can be sure that anyone who loses data *and* has a backup of that data is much happier than anyone who loses data and has no backup. And yet, talk to a sampling of small-to-medium businesspeople, and many of them will hang their heads in shame when backups are discussed.

This book doesn’t set out to discuss all the aspects of a complete “recovery plan”, but it will show you how to backup and restore your file systems ... a critical part of any plan for disaster recovery.

Various Options for Backup

There are a large number of options for backup of your data — just as there are a large number of companies who put statistics about data loss on their websites. You can `cp(1)` or `scp(1)` or `cpio(1)` everything to another location. You can burn data to an optical medium. The days are not too far gone when you could save valuable data to a floppy drive — now, many people may use a flash memory device instead. There are many fine 3rd party programs (AMANDA, Veritas, Bacula, `rsync` and many more — try ‘make search -key backup’ under `/usr/ports`) that can be used to back up files.

The canonical backup program, and one that is available on every system, is `dump(8)`. Its “partner” is `restore(8)`. `Dump(8)` was designed to backup UNIX and UNIX-like systems to tape drive quickly and efficiently. `Dump(8)` is fast, can write to a variety of media, works on live filesystems, does incremental backups, already know tons about your filesystem(s), and has a host of other options to make it work for you in almost any situation.

I’m not trying to tell you that you can’t use other programs for your backups, but I am telling you that for this book, we’re going to use `dump(8)`. Make backups regularly and often! You will someday be very glad that you did.

About `dump(8)`

For the full “dump” on `dump(8)`, see the manpage. Here are some things you should remember about `dump(8)`:

`Dump` has ten “levels” of backup (0-9). A level zero dump (`'dump -0'`) will backup all files on a filesystem. A higher level dump will only backup files changed since the last dump of a lower level. Level zero is the default.

Unless a filesystem is unmounted or mounted read-only, you should tell `dump` that the filesystem is live (`'-L'`). `Dump` will then make a snapshot of the filesystem and dump the snapshot (so that any activity on the drive doesn’t break the “dump”).

You must specify which filesystem you want dumped, either by its mount-point name (e.g. “/usr”), or by its special node name (e.g. “/dev/ad1s1a”). Finally, by default dump(8) writes to a tape drive (/dev/sa0). By using the ‘-f’ option to dump, you may have dump write its output to another file, a special device, or even the standard output.

For information on dump’s other options, including tape size and density specifications, blocksize for the dump, auto-sizing of output, operator notifications, estimation of tape requirements, manipulation of the dump date data and other features as well as environment variables that affect dump, see the dump(8) manpage.

About restore(8)

Restore(8) is used to turn a “dump” back into usable data. The general format of the restore command is as follows:

```
# restore [-flag] [-options]
```

To rebuild a file system, the “-r” option is generally used.

Restore is generally used on a “pristine” file system (one that has been recently ‘formatted’ with newfs(8)), and a level zero dump is restored before any incremental dumps can be restored. Change to the target directory before beginning the restore procedure.

Note that this is a rather abbreviated discussion of dump(8), and that you should refer to the dump(8) manpage and the examples below for more information.

Examples

Practice Exercises

More information

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

3.11 Determine the directory structure of a system

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Examples

Practice Exercises

More information

hier(7)

3.12 Manually run the file system checker and repair tool

Author: ?? andreas dot kuehl at clicktivities dot net FreeBSD

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

Concept

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

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?**)

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

Practice Exercises

More information

`fsck(8)`

3.13 View and modify file flags

Author: Ivan Voras IvanVoras FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

All current BSD system support an additional mechanism for fine-grained tuning of file security called “file flags”. Though similar, this should not be confused with standard Unix file access modes. The flags are:

- `archived` - The file may be archived
- `opaque` - The directory is opaque when viewed through a unionfs stack
- `nodump` - The file is to be ignored by backup utilities (like `dump(1)`)
- `sappend` - The file can only be appended to (cannot change it’s contents) by any and all users
- `schange` - The file can not be changed by and and all users
- `sunlink` - The file cannot be unlinked
- `uappend` - The file can only be appended to by its owner
- `uchange` - The file can only be changed by its owner

- `unlink` - The file can only be unlinked by its owner

Some of these flags can only be (re)set by the super-user. This includes archived and all `s*` flags.

A very important aspect of file flags is how they interact with `securelevel`. When `securelevel` is set to 1 or above, flags cannot be modified even by root, thus enabling creation of fortified system that cannot be damaged (deliberately or not) by the superuser. Note: Secure levels are covered in ?? .

The ideas behind the security-related flags are:

- Log files should be append-only, thus preventing an attacker to mask his trail by modifying them
- In a similar vein, log files should be prevented from deletion (i.e. `unlink`)
- System configuration files, SUID files, and other (at administrator's discretion) must be prevented from unauthorized change.

Downsides of this approach are that most administrative tasks must be done in single user mode on the machine's console, and that log rotation cannot work or is very tricky to do.

Some utilities have been modified to natively support file flags. For example, `ls` accepts `-o` argument to display file flags for each file.

Examples

```
> chflags uappend important.log
(succeeds)
> ls -l important.log
-rw-r-r-  1 ivoras  ivoras   172 Jan 20 00:42 important.log
> ls -lo important.log
-rw-r-r-  1 ivoras  ivoras  uappnd 172 Jan 20 00:42 important.log
> echo garbage > important.log
important.log: Operation not permitted.
> echo garbage >> important.log
(succeeds)
> mv important.log unimportant.log
important.log: Operation not permitted
```

Practice Exercises

1. Set `sappend` flag to `/var/log/security` and `/var/log/userlog`. Investigate what happens when log rotation is attempted.
2. List files in `/usr/bin` and inspect their flags - notice that some are marked `schange` (or `schg`) by default.
3. Not all utilities understand BSD file flags - check your backup and file management tools.

More information

ls(1), chflags(1)

3.14 Monitor the virtual memory system

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

4 Chapter Users and Accounts Management

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: Yannick Cadin 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.

- 4.2
- 4.3
- 4.8
- 4.9
- 4.10
- 4.11
- 4.7
- 4.4
- 4.5

4.1 Protect authentication data

Author: ?? ceri@FreeBSD.org FreeBSD/OpenBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

On a BSD system, user and group information is stored in a local set of password database files, namely **/etc/master.passwd**.

The password database contains user information such as the user's username, user id, real name, shell, etc.. This information is used by a large number of user programs such as `ls(1)`, `login(1)`, `id(1)` and so on, which need to determine and possibly display information about one or many users – for example, running “`ls -l`” in `/tmp` may need to retrieve a number of usernames. Of course, the password database must also contain important security related information used by the system, such as the user's encrypted password hash and information required to support features such as password aging and account expiration.

In order to prevent access to the second set of information to processes that do not require it, `/etc/master.passwd` is readable only by the root user, and a second file, **/etc/passwd**, is created which contains only the first set of non-privileged information and is readable by all users.

```
# ls -l /etc/master.passwd /etc/passwd
-rw----- 1 root  wheel  3704 Jan  7 12:58 /etc/master.passwd
-rw-r-r-  1 root  wheel  3028 Jan  7 12:58 /etc/passwd
```

On a heavily used system with a large number of users, repeatedly searching the flat files `/etc/passwd` and `/etc/master.passwd` can take a long time and cause performance issues on the system. Therefore, BSD systems maintain binary versions of these files for fast lookups. **/etc/pwd.db** is the binary version of `/etc/passwd`, while **/etc/spwd.db** is the binary equivalent of `/etc/master.passwd`. These files are created with the `pwd_mkdb(8)` command.

Since these files contain the same information as the non-binary versions, they must be similarly protected.

```
# ls -l /etc/spwd.db /etc/pwd.db
-rw-r-r-  1 root  wheel  57344 Jan  7 12:58 /etc/pwd.db
-rw----- 1 root  wheel  57344 Jan  7 12:58 /etc/spwd.db
```

Maintaining the Password Databases

Note that, as the name implies, `/etc/master.passwd` is considered the primary source for user information on a BSD system. Therefore, if you make manual changes to `/etc/passwd` as documentation for other systems may suggest, your changes can be lost. In order to ensure the integrity of your password databases, only use system provided tools such as `vipw(8)` to maintain them. See section 4.2 for information on adding and removing users and modifying the databases.

TODO: make sure we don't have redundant information between these sections. Also check or point to ?? . Also maybe move the sections together in the book.

Practice Exercises

1. Look at the entries for the root user in `/etc/master.passwd` and `/etc/passwd` on your system. Use the `passwd(5)` manual to determine which fields are not present in `/etc/passwd`
2. Rebuild the binary lookup databases on your system with “`pwd_mkdb /etc/master.passwd`”. Note that the timestamps are updated on `pwd.db` and `spwd.db`.

More information

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

4.2 Create, modify and remove user accounts

Author: *name ?? ??*

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

Reviewer: Alistair Crooks agc@NetBSD.org NetBSD

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.

Introduction

TODO: maybe section 4.1 can cover `vipw`.

Details about the password database are covered in section 4.1 .

TODO: this section is BSD specific, so break it down briefly for `pw`, `adduser`, `useradd`, etc per BSD flavor

TODO: point to section ?? about `chsh`, `chpass`, `chfn` which can do some of the same. TODO: also section 4.8 is related. We need to put these four sections together in book

Examples

Practice Exercises

More information

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

4.3 Create a system account

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

A system account is generally a user used for a specific purpose and associated with a specific daemon. They are normal accounts with a UID, but usually differ for a few reasons:

- probably do not need a usable shell
- do not need a valid password (as no one would ever login using this account)
- might not have a standard home directory

A default install of a BSD system has several system accounts, for example:

```
root::0:0::0:0:Charlie &:/root:/bin/csh
daemon:*:1:1::0:0:Owner of many system processes:/root:/sbin/nologin
operator:*:2:5::0:0:System &:/sbin/nologin
bin:*:3:7::0:0:Binaries Commands and Source:/sbin/nologin
sshd:*:22:22::0:0:Secure Shell Daemon:/var/empty:/sbin/nologin
_pflogd:*:74:74::0:0:pflogd privsep:/var/empty:/sbin/nologin
nobody:*:32767:32767::0:0:Unprivileged user:/nonexistent:/sbin/nologin
```

TODO: put some hash in the root's password field or maybe do not include "root" here

Notice that system accounts usually have an asterisk in the password field. No hash algorithm can match this, so the password is disabled. Also the default shell for many system users is commonly the nologin program.

4.4. CONTROL WHICH FILES ARE COPIED TO A NEW USER'S HOME DIRECTORY DURING ACCOUNT CREATION

TODO: explain purposes of these at least TODO: explain path to nologin differs or point to section that talks about it? TODO: point to section that explains ampersand in gecost field
TODO: explain nobody

Other common system accounts include: uucp, www, toor, bind or named, proxy, and mail-null or postfix. Installing packages may also include additional system accounts, such as cyrus, gdm, and pgsqld.

TODO: mention system groups also

TODO: mention dedicated accounts – such as users or groups for mail or source builds or backup jobs, etc.

Examples

TODO: show example using useradd and pw to create a system user

Practice Exercises

1. Manually run nologin

More information

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

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

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

4.5 Change a password

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

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: Kevin D. Kinsey kdk@daleco.biz FreeBSD

Reviewer: *name ?? ??*

Concept

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

Introduction

Passwords help to establish trust; trust that you, as a computer user, are indeed “authorized personnel”; trust that the computer you are “logging into” is indeed the machine that it is supposed to be; trust that files that bear your username as “owner” were indeed originally created by you — trust is fairly important, don’t you agree?

Good system security demands that users’ passwords or passphrases be changed from time to time, even regularly in many cases. The “root” account should be protected by a strong password to ensure that “normal users” cannot cause problems with a multi-user machine by inadvertently shutting down, unmounting devices in use, installing unapproved software, etc.

You need to know how to change your password, and, as a system administrator, change the password of the “root” user, or another user in your system. For example, you might need to do some “system recovery”, but the root password is lost (however, if you can boot in single-user mode you can change root’s password). You might wish to change a user’s password because they are afraid their original password has been compromised. You might wish to change your password because you have a new “significant other” and you want to forget the old one’s name. Good news! Although your reasons may vary, the procedure is the same! The program **passwd** (1) has everything you need for this chore!

Examples

You can change root’s password only if you are logged in as the root user, or use **su** to substitute root’s credentials for your own.

Running **passwd** with no arguments will allow you to change your password. Note that it is often a good idea to make sure that you are indeed the user you think you are before attempting to make a password change:

```
$ id
uid=1001(someuser) gid=0(wheel) groups=0(wheel)
$ passwd
Changing local password for someuser
Old Password:
New Password:
Retype New Password:
```

To change root’s password, we must first get permission:

```
$ id
uid=1001(someuser) 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 first check our own identity; since we are “someuser” instead of the root user, we know we must gain root’s credentials to change root’s password. We use **su** and type root’s current password to gain root access.

We then use **passwd** without a second argument, to change the password for the current user - in our example, root. But suppose we wish to change *another* user’s password:

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

It is not recommended to do this to your friends without consulting them first!

Practice Exercises

1. Change your password.

2. Change the root password.
3. Change the password of another user.

More information

passwd(1), vipw(8)

4.6 Change the encryption algorithm used to encrypt the password database

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

TODO: test and document on other BSD systems – as appropriate make this content not BSD specific

On NetBSD, the passwd(1) command (covered in section TODO) can use an /etc/passwd.conf configuration to choose the password algorithm. The format is ..TODO... The default is “old” which is the common crypt(3) DES encryption scheme. TODO: passwd.c doesn’t call pw_getconf() so I think it must use pam for this?

The following are examples of different results based on the same password.

- old but common crypt(3) DES: **7rpABVh3LoKjE**
- MD5: **\$1\$FSh3ps5T\$Etg/3eGiSBqdGahf29IIN1**
- NetBSD newsalt: **_GI/.Sw2RBVnj01TI6Tc**
- SHA1: **\$sha1\$21773\$uV7PTeux\$I9oHnvwPZHMO0Nq6/WgyGV/tDJIH**
- Blowfish: **\$2a\$04\$3/vwv4ibdVz2SUG3w.SRwOgI6kk7FUmMcvswZ/KUS9bngvgGEkqNq**

As you can see, the new algorithms use a format that can be recognized by routines (like TODO) so they know what to compare with. TODO: show this format

(Note while high rounds may improve security, it can be expensive – very slow to generate hash.) TODO: reword this or better explain

Examples

Practice Exercises

More information

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

4.7 Change a user's default shell

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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 and related user databases. See section 4.1 for details.

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. Or maybe not as is covered in another section.

TODO: point to section ?? and '4.8 . Put these four sections next to each other in book

Examples

Practice Exercises

More information

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

4.8 Lock a user account or reset a locked user account

Author: *name ?? ??*

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

Reviewer: *name ?? ??*

Concept

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

Introduction

Locking an account is commonly accomplished by modifying the user's password field in the user database. This can be done manually using vipw or chpass.

One FreeBSD and DragonFly, the pw utility can be used to lock and unlock an account. It locks an account by prefixing the password field with “*LOCKED*”. For example to lock a user by name:

```
$ pw lock fred
```

Or to unlock an account by the UID:

```
pw unlock 2395
```

On NetBSD, the usermod (or user mod) program with the -C switch can be used to lock accounts. It also prefixes the password hash with “*LOCKED*”. To lock an account use:

```
$ usermod -C yes julie
```

And to unlock the account:

```
$ usermod -C no julie
```

TODO: OpenBSD's version of NetBSD's usermod doesn't have this lock functionality; does OpenBSD have a tool for this? Or just do it manually?

TODO: locking accounts can also be done with password change time and expiration times. This could mention that briefly, but doesn't cover it.

TODO: point to section describing master.passwd format

Examples

Practice Exercises

1. Use `vipw` or `chpass` to manually lock the account. And then test a login. And then unlock.

More information

`vipw(8); chpass(1), pw(8), user(8)`

4.9 Determine identity and group membership

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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.

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.

```
$ 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.

```
$ groups
users audio mail cvs
$ groups john
users mail
$ groups mike
groups: mike: no such user
```

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.

```
$ id
uid=1001(user) gid=100(users) groups=100(users), 92(audio), 1003(mail), 1004(cvs)
```

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

```
$ 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:

```
$ 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:

```
$ 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 do superuser *root* . Password is not echoed in any form (not even with * marks).

```
$ whoami
user
$ su
Password:
# whoami
root
```

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

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

```
$ whoami
user
$ echo $HOME
/home/user
$ su - john
Password:
$ whoami
john
$ echo $HOME
/home/john
```

Practice Exercises

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

More information

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

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

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

After logging into an account on BSD system we can see an information like:

```
Last login: Thu Jan 11 20:18:18 2007 on ttyv4
```

This and other kind of information about users and their doings (logins and logouts) is stored in three files:

- `/var/run/utmp` which records information about current users,
- `/var/log/wtmp` containing information on users' logins and logouts, as well as system's shutdowns and reboots (which won't be discussed here),
- `/var/log/lastlog` storing information on users' last logins.

Of course, manually gathering information from aforementioned files makes no sense at all. Thus the BSD systems are equipped with a handful of simple commands that will fetch required information for us.

Examples

Determining user's last login time and date can be performed with a **lastlogin(8)** command:

```
$ lastlogin
root          ttyv2          Thu Jan 11 19:12:23 2007
mike          ttyv1 192.168.112.24 Thu Jan 11 20:43:05 2007
```

When executed with no user names **lastlogin(8)** displays information for all users. Adding user name makes **lastlogin(8)** display information regarding only specified user.

The **last(1)** command displays a list of last logins. Executed without any parameters returns a list for user executing it. To minimize the scope of returned list we can use the **-n** flag, specifying maximum number of lines.

```
$ last
-n5 mike
mike          ttyv1 192.168.112.24 Thu Jan 11 20:43 - 20:43 (00:00)
mike          ttyv4          Thu Jan 11 20:42 - 20:42 (00:00)
mike          ttyv4          Thu Jan 11 20:41 - 20:41 (00:00)
mike          ttyv0 192.168.112.24 Thu Jan 11 20:37 still logged in
mike          ttyv0 192.168.112.24 Thu Jan 11 20:18 - 20:37 (00:19)
```

The **users(1)** utility lists the login names of the users currently logged into the system.

```
$ users
root therek
```

The **w(1)** and **who(1)** tools returns a little more detailed information on current users. The **who(1)** command displays who is on the system, while the **w(1)** presents also an information on what they are doing as well as some other system information (covered in section 5.20).

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

```
$ who
root          ttyv4      Jan 11 21:27
therek       ttyp0     Jan 11 20:37 (192.168.112.24)
$ w
9:31PM up 19 days,  1:12, 2 users, load averages: 0.00, 0.02, 0.00
USER          TTY          FROM          LOGIN@      IDLE WHAT
root          v4           -             9:27PM      3 -csh (csh)
therek       p0          192.168.112.24  8:37PM      - w
```

BSD systems give us also an ability to check some more information on system users. To do so, we can use a **finger(1)** utility with optional user name.

```
$ finger
Login          Name          TTY  Idle  Login  Time  Office  Phone
root          Charlie Root  *v4   14   Thu   21:27
mike          Mike Erickson p0     Thu   20:37
$ finger
mike
Login: mike          Name: Mike Erickson
Directory: /home/mike Shell: /usr/local/bin/bash
On since Thu Jan 11 20:37 (CET) on ttyp0 from 192.168.112.24
Last login Thu Jan 11 20:43 (CET) on ttyp1 from 192.168.112.24
New mail received Thu Jan 11 21:38 2007 (CET)
  Unread since Thu Jan 11 21:28 2007 (CET)
No Plan.
```

Practice Exercises

1. Execute **lastlogin(8)** without, with only one, and with at least two user names.
2. Login to a couple of different accounts and check the result of **who(1)** command with **-H** and **-q** flags.
3. Login to a couple of different accounts and check the result of **w(1)** command executed with flags: **-d** , **-i** , **-h** .
4. Compare the output of **finger(1)** command with **-s user** and **-hs user** parameters.
5. Try out **finger(1)** with **-l** flag.

More information

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

4.11 Enable accounting and view system usage statistics

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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”

Examples

Practice Exercises

More information

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

5 Chapter Basic System Administration

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: Yannick Cadin 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.

- 5.1
- 5.2
- 5.2
- 5.5
- 5.6
- 5.7
- 5.8
- 5.11
- 5.9
- 5.10
- 5.15
- 5.12
- 5.15
- 5.17
- 2.9
- 5.17
- 5.4

- 5.19
- 5.13
- 5.20
- 5.20
- 5.22
- 5.23
- 5.24

5.1 Determine which process are consuming the most CPU

TODO: is this section header okay? “process is” versus “processes are”

- No, we definitely need one of the above. I suggest “processes are” –ceri

Author: hubertf ?? ??

Reviewer: ?? ceri@FreeBSD.org FreeBSD|OpenBSD

Reviewer: *name* ?? ??

Concept

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

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 of the screen, 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 determined 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, with the exception of `systat(1)`, 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 a “niceness” level, which goes from -20 (not nice at all, the highest priority) to 20 (very nice, or the lowest priority). Nice levels below (i.e., priorities higher than) 0 are reserved for the superuser, and a non-superuser process that has had its nice level increased (and therefore its 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)`, and 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 a last resort, as unpleasant side-effects may happen! In most cases, the default of `SIGTERM (15)` is sufficient to end a process.

Examples

Determine the process that takes most CPU:

```
% ps -aux | head -3
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT STARTED     TIME COMMAND
feyrer    5924 50.0   5.9 104588 30900 ttyp6 R+   12:17AM  0:03.31 qemu -m 64 -bo
root      25528 13.7   0.2   468   1104 ttyp4 R+   12:17AM  0:00.85 /usr/bin/find
```

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 <pgrp>...] [-u <user>..
```

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 -m 64 -bo
feyrer    1206  1.9 10.8  73896 56304 ?        Ra    11:48AM 75:11.33 /usr/pkg/lib/f
```

This is because no other process claims the CPU. If another process (e.g. your windowing system, or a compile job) would claim the CPU, the qemu process would relinquish the CPU 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; they send the SIGKILL signal to the process with process ID 5924.

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.

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 ?? ??

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

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.

Introduction

Section 5.1 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.
- **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).

Examples

See section 5.1 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
#
```

Practice Exercises

See section 5.1 and perform the same tasks with `pgrep(1)` and `pkill(1)`.

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 ?? ??

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

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.

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).

Examples

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

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

```
netbsd% ls /etc/rc.d
DAEMON      downinterfaces  lpd          postfix      sshd
LOGIN       fixsb           mixerctl     powerd       staticroute
NETWORKING  fsck            mopd         ppp          swapl
SERVERS     ftpd           motd         pwcheck      swap2
accounting  hostapd        mountall     quota        sysctl
altqd       identd         mountcritlocal  racoon       sysdb
amd         ifwatchd       mountcritremote  raidframe    syslogd
apache      inetd          mountd       raidframeparity  timed
apmd        ipfilter       moused      rarpd        tpctl
bootconf.sh ipfs           mrouted      rbootd       ttys
bootparams  ipmon          named        root         veriexec
btconfig    ipnat          ndbootd     route6d      virecover
btcontrol   ipsec          network     routed       wdogctl
bthcid      irdaattach     newsyslog   rpcbind      wscons
ccd         iscsi_target   nfsd        rtadvd       wsmoused
cgd         isdnd          nfslocking  rtclocaltime  xdm
cleartmp    kdc            ntpd        rtsold       xfs
cron        ldconfig       ntpdate     rwho         ypbind
dhclient    lkm1           pf           savecore     yppasswdd
dhcpd       lkm2           pf_boot     screenblank  ypserv
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          rpcbind
LOGIN       dhclient       kerberos     nfslocking    rtadvd
NETWORKING  dmesg          keyserv      nfsserver     rwho
SERVERS     dumpon         kldxref      nisdomain     savecore
abi         early.sh       kpasswdd     nsswitch      sdpd
accounting  encswap       ldconfig     ntpd          securelevel
addswap     fsck           local        ntpdate       sendmail
adjkerntz   ftpd          localpkg     othermta      serial
amd         gbde          lpd          pccard        spps
apm         geli          mdconfig     pcvt          sshd
apmd        geli2         mdconfig2    pf            swapl
archdep     hcsecd        mixer        pflog         syscons
atm1        hostapd       motd         pfsync        sysctl
atm2        hostname     mountcritlocal  power_profile  syslogd
atm3        ike           mountcritremote  powerd         timed
auditd      inetd         mountd       ppp           tmp
auto_linklocal  initrandom   mountlate    pppoed        ugidfw
bgfsck      ip6addrctl   moused       pwcheck       usbd
```

bluetooth	ip6fw	mroute6d	quota	var
bootparams	ipfilter	mouted	ramdisk	virecover
bridge	ipfs	msgs	ramdisk-own	watchdogd
bsnmpd	ipfw	named	random	wpa_supplicant
bthidd	ipmon	natd	rarpd	ypbind
ccd	ipnat	netif	resolv	yppasswdd
cleanvar	ipsec	netoptions	root	ypserv
cleartmp	ipxroute	network_ipv6	route6d	ypset
cron	isdnd	newsyslog	routed	ypupdated
devd	jail	nfscclient	routing	ypxfrd

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 rcvar reload)
```

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
#
```

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.

More information

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

5.4 Configure a service to start at boot time

Author: *name ?? ??*

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

5.5 View and configure system hardware

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

5.6 View, load, or unload a kernel module

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

TODO: Are they really discouraged? Where documented?

Introduction

Kernel modules provide extra functionality that a system administrator can add to their running kernel. Common examples include audio device drivers, network interfaces, RAID and other hardware drivers, extra filesystems support, packet filtering, binary compatibility support for

other operating systems (like Linux), and console screen savers. In most cases, this extra functionality or hardware support can be configured and then built in to your monolithic kernel. But using kernel modules may be more convenient.

On FreeBSD and DragonFly, the kernel module filenames have an .ko extension. NetBSD and OpenBSD kernel modules end with .o.

On FreeBSD, the default kernel modules are located at /boot/kernel. DragonFly's kernel modules are located in the /modules directory. NetBSD's kernel modules are at /usr/lkm. TODO: And OpenBSD

TODO: list a few common modules TODO: show how to find modules

FreeBSD and DragonFly use kldload, kldunload, and kldstat to load, unload and to view details, respectively. NetBSD and OpenBSD use the modload, modunload, and modstat tools.

Examples

The following is an example of running kldstat to list modules on a DragonFly system:

```
# kldstat
Id Refs Address      Size      Name
1     5 0xc0100000 5eff14   kernel
2     1 0xc06f0000 2b58     ecc.ko
3     1 0xc06f3000 591d0    acpi.ko
4     1 0xdd68c000 3000     null.ko
```

The “refs” column shows the number of modules referenced by the kernel object. TODO: The “address” column shows the load address of (the pointer to) the kernel object. The “size” is the size in hexadecimal.

In the above examples, “ecc” is for AMD64 ECC memory controller, “acpi” is for ACPI power management, and “null” provides the mount_null filesystem support.

TODO: explain why the kernel listed

TODO: should this mention -v? maybe not for BSDA?

Practice Exercises

More information

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

5.7 Modify a kernel parameter on the fly

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

Reviewer: Mark Foster mark@foster.cc FreeBSD

Reviewer: *name ?? ??*

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.

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`

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.\$\$

Practice Exercises

For OpenBSD and FreeBSD. Change on the fly these variables:

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

Set these variables in system files (as described above) and reboot, check that variables are changed after rebooting.

TODO: let's just use same variables that are common to all these for a beginning admin – by keeping few differences between the BSDs will make this book easier for new admin

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

More information

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

5.8 View the status of a software RAID mirror or stripe

Author: *name ?? ??*

Reviewer: ?? ceri@FreeBSD.org FreeBSD|OpenBSD

Reviewer: jdq ?? OpenBSD

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.

Introduction

Software RAID

Raid is performed by using a kernel device driver. Raidframe is an example of this. Software raid is a inexpensive raid solution that can be deployed on any system. Software raid is a great opportunity to practice raid without investing in an expensive raid card.

Hardware RAID

Hardware raid is performed by a controller card. The most common are produced by LSI and Adaptec. These controller cards offload the workload of parity and transactions across multiple disks, and provide the operating system with a single virtual device to represent this set. It is notable to mention that many cards advertised as raid cards are simply controller cards bundled with a driver that requires the OS to handle parity and transactions. This is not a hardware raid

solution. Also many hardware raid cards available on todays market may only be configured during bootup.

RAID Levels

RAID level 0

Raid level 0 traditionally described a grouping of disks, with data striped evenly across without parity. This means that any single disk failure results in the failure of the complete set. the term “Raid level 0” no longer necessarily means the data is evenly distributed across all disks in a stripe, just that a set of disks are not fault-tolerant.

RAID level 1

Raid level 1, also called mirroring or shadowing, groups disks into pairs. A single copy of each block is stored on an accompanying disk. Raid level 1 is highly reliable and can tolerate disk failures up to $N/2$ without losing data, as long as two disks in a pair do not fail.

RAID level 3

In Raid level 3, data is striped across data disks, and an additional parity disk stores the parity of the data. When any single disk fails, the data may be recovered by computing the incomplete data from the parity disk. Multiple disk failures may be tolerated with the addition of multiple check disks.

RAID level 5

Raid level 5 is similar to raid level 3, with the exception that the parity data is evenly distributed acrossed all disks.

Raid on Raid

It is possible to combine raid levels. For instance, You may build two raid level 1 sets of a pair of two disks each. You may then stripe these two raid level 1 sets as raid level 0. This is not commonly done due to complexity, but is available when necessary.

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.

To check the status of a raid device, `/dev/raid0`, use:

```
raidctl -s raid0
```

ccd

ccd is a software raid solution. Example output...

gstripe/raid3/mirror

gstripe, graid3, and gmirror is a software raid solution on FreeBSD(?). Example output...

View status of a gstripe set:

```
gstripe status
```

View status of a gmirror set:

```
gmirror status
```

View status of a graid3 set:

```
graid3 status
```

gvinum/vinum

gvinum on freebsd and vinum on netbsd is a software raid solution. Example output...

bioctl

bioctl is an OpenBSD userland interface to hardware raid controllers and enclosures.

Example of checking the health of a bioctl-compatible raid set:

```
loki@i386 man4$ sudo bioctl arc0
Password:
Volume  Status      Size           Device
arc0 0 Online      127999672320 sd2      RAID5
      0 Online      320072933376 0:0.0   noenc1  <ST3320620AS 3.AAD>
      1 Online      320072933376 0:1.0   noenc1  <ST3320620AS 3.AAD>
      2 Online      320072933376 0:2.0   noenc1  <ST3320620AS 3.AAC>
arc0 1 Online      127999672320 sd3      RAID0
      0 Online      320072933376 0:0.0   noenc1  <ST3320620AS 3.AAD>
      1 Online      320072933376 0:1.0   noenc1  <ST3320620AS 3.AAD>
      2 Online      320072933376 0:2.0   noenc1  <ST3320620AS 3.AAC>
arc0 2 Online      127999672320 sd4      RAID1
      0 Online      320072933376 0:0.0   noenc1  <ST3320620AS 3.AAD>
      1 Online      320072933376 0:1.0   noenc1  <ST3320620AS 3.AAD>
      2 Online      320072933376 0:2.0   noenc1  <ST3320620AS 3.AAC>
```

Definitions

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

See Also

RAIDframe

- ?? CMU RAIDframe
- ?? NetBSD and RAIDframe

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

5.9 Configure system logging

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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

Introduction

Examples

Practice Exercises

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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

5.11 Determine which MTA is being used on the system

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

By default, the BSD systems have mail transfer agent (MTA) enabled for at least handling mails sent from the local machine.

FreeBSD, OpenBSD, and DragonFly include Sendmail in their base installs. NetBSD includes Postfix. (TODO: FOOTNOTE: Older versions of NetBSD included both Sendmail and Postfix.)

Third-party MTA alternatives or sendmail replacements are available through the package collections, such as Exim, Qmail, Postfix and many others. (TODO: check Qmail – I don't think in OpenBSD ports.)

Selecting the MTA can be done by configuring the mailer.conf configuration used by the mailwrapper tool. The BSD's default /usr/sbin/sendmail (TODO: what about /usr/lib/sendmail) is really a symlink to /usr/sbin/mailwrapper. This uses /etc/mail/mailer.conf (or /etc/mailer.conf on NetBSD) to define common replacement programs.

TODO: for example

TODO: show how can be tested

TODO: note that this BSDA doesn't teach how to configure a mail server, but TODO: briefly cover how to turn on and off and how to disable at boot and where logs are at TODO: point to other sections for details

Examples

Practice Exercises

1. What is y

More information

mailer.conf(5)

5.12 Create or modify email aliases for Sendmail or Postfix

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Examples

Practice Exercises

More information

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

5.13 View the Sendmail or Postfix mail queue

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

As noted in section 5.11 , the BSD systems use Sendmail or Postfix by default for handling mail.

The mail queue can be displayed using the **mailq(1)** utility. The queue listing identifies messages that are still queued (not successfully sent or delivered yet). The output includes MTA's internal message identifier, the size of the message, the date and time the message was accepted into the queue, the sender's envelope address, and the recipient address(es), as well as a reason of failure for messages that have permanently failed.

When using Postfix, if the **mailq(1)** utility is not setup, use **postqueue -p** to display the traditional sendmail-style queue listing. To make MTA attempt to deliver all queued mail issue commands: **sendmail -q** for Sendmail and **postqueue -f** for Postfix.

Examples

Following are two examples of **mailq(1)** output. First when used with Sendmail:

```
# mailq
/var/spool/mqueue (1 request)
---Q-ID--- -Size- ---Q-Time--- -----Sender/Recipient-----
10ID36a2085983      524 Thu Jan 18 14:03 <sender@mydomain.net>
                    (Deferred: Operation timed out with otherdomain.com.)
                    <recipient@otherdomain.com>

Total requests: 1
```

And an example when used with Postfix:

```
# mailq
-Queue ID- -Size- ---Arrival Time--- -Sender/Recipient-----
D184ACAB55      709 Fri Jan 19 20:50:08 sender@mydomain.net
(delivery temporarily suspended: connect to mail.otherdomain.com[10.0.0.11]: Connection
recipient@otherdomain.com

- 709 bytes in 1 Request.
```

Practice Exercises

1. When logged into your mail server, see the output of **postqueue(1)** and/or **mailq(1)** commands (depending on which MTA is used).

More information

mailq(1), postqueue(1)

5.14 Read mail on the local system

Author: jdq ?? OpenBSD

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

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.

Introduction

cron job output is emailed. cron -l. see cron. see cron run.

(Maybe refer to section 5.19 and/or section ??).

location of mailboxes, /var/mail/... read mailboxes with mail -f

Examples

Show example of files that may indicate which MTA is in use. wrapper/alias file somewhere.

Refer to section 5.11

Practice Exercises

write an email,

```
mail
mail root, or 'm root'
begin typing your message. Lets ask the admin what version of ssh we have
how to set subject? find out with ? for help.
  s this is the subject
  e bring message to text editor (EDITOR), save
add someone to CC, see ? it is c, so:
  c junioradmin
we are done here, ^d
EOT
&
```

reply to an email,

```
sudo mail -u root
list, reply to 1
r read file into message
!man ssh , to find out the switch to print ssh version,
|ssh -V, output is put into current message
cc junioradmin
save
```

add new folder 'questions', add message 1 to folder 'questions'

More information

mail(1), /var/mail/\\$USER, cron(8)
??

5.15 Understand basic printer troubleshooting

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

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

Author: Kevin D. Kinsey kdk_at_daleco_dot_biz FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

When computers were first introduced to the general public (outside of the scientist's labs that were developing them), they were large and very expensive machines. For this reason, almost all operating systems of that time period were conceived and developed as "multi-user" systems. The BSDs, with deep roots in the UNIX tradition, are no exception.

While the advent of the PC ("personal computer") has made it possible for each user in some organizations to have their own machine, in many cases BSD machines are still used by more than one user; in fact, most system "daemons" (servers) are classed as "users" as well. So are "client" computers that are connecting to your machine's mail, web, or other services. Rebooting while a program is performing an operation can cause problems. Also, it might be considered downright rude to halt a system while a user is still working on an important project.

For this reason, it is important to consider the issue of halting or rebooting a system. Doing things the Right Way(tm) can benefit you greatly in the long run.

System States

From a purely logical point of view, the computer system has two states: OFF and ON. However, in reality, there are more possibilities: the system could be booting (going from OFF to ON), or powering-down (going from ON to OFF), or "rebooting" (going from ON to OFF to ON again). In addition, when a BSD system is ON, it can be in either single-user or multi-user mode.

Single-user and Multi-user modes

During a "normal" boot process, `init(8)` attempts to bring the system up to "multi-user" mode. System `tty`'s are made available, all daemons for the base system or installed from third-parties and enabled by `rc` configuration files are started, network interfaces are brought "up", and so on. In the event that something goes wrong during this process, `init` will start "single-user" mode instead. "Single-user" mode may also be forced from the boot menu during second stage booting. Finally, you can "downgrade" a system to single-user mode using commands similar to those used to halt, reboot, or power-down the system.

In single-user mode, there are no `ttys` available, the network is not brought up, etc. The only possible "login" is as `root`, from the console. Generally, no password is required for `root` in single-user mode. This can be changed by editing the `ttys(5)` file and marking the console as "insecure", in which case `init(8)` will ask for the root password before allowing a shell in single-user mode. In cases where unauthorized personnel may have access to the physical machine, it's a good idea to make this change.

Shutdown(8)

shutdown(8) is a slightly nicer interface to the underlying system calls halt(8), fasthalt(8), reboot(8), etc. It allows a warning message to be sent to other users, allows you to set a time in the future for the change in system state to begin, etc.

The beginning administrator should be familiar with the behavior of shutdown(8) when called with various flags as shown in the “Examples” section below.

Examples

```
# shutdown
```

When called with no flags, shutdown will bring the system to single-user from multi-user mode.

```
# shutdown -p now
```

The system will be powered off immediately. This is partly dependent on hardware support. If the system cannot be powered off automatically, the operating system will halt (equivalent to “shutdown -h now”) instead.

```
# shutdown -r 45
```

The system will be rebooted in 45 minutes. See shutdown(8) for more information about what will happen in the meantime.

```
# shutdown -p 0912312359
```

The system will power down one second prior to New Year’s Day, 2010.

```
# shutdown -k
```

“Kick” mode. Kicks off all users except root, but leaves the system in multi-user mode.

Practice Exercises

1. Use shutdown(8) to halt and reboot a system. Attempt to use it to power-down a system, and note if the machine has hardware support for an automatic power-down.

More information

shutdown(8)

5.17 Recognize the difference between hard and soft limits and modify existing resource limits

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

5.18 Recognize common, possibly third-party, server configuration files

Author: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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.

Examples

Apache HTTPD

Note: this is included as part of OpenBSD.

BIND “named”

This is included in default install.

DHCP Daemon

TODO: the BSDs have different implementations, anything common?

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

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

Samba

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

XFree86 or Xorg

TODO: should this be briefly mentioned too?

Practice Exercises

More information

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

5.19 Configure the scripts that run periodically to perform various system maintenance tasks

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

The BSD systems provide scripts for verifying integrity and security and for providing daily, weekly, and monthly system maintenance and reports. These are started via cron. (Cron is covered in section ?? .)

OpenBSD and NetBSD use shell scripts called `/etc/daily`, `/etc/weekly`, and `/etc/monthly`.

FreeBSD and DragonFly use a tool called 'periodic' that runs several other scripts found in `/etc/periodic/daily`, `/etc/periodic/weekly`, and `/etc/periodic/monthly` directories.

The output of the maintenance jobs is saved to `/var/log/daily.out`, `/var/log/weekly.out`, and `/var/log/monthly.out`.

Also, the same reports are emailed to "root" (by default).

TODO: is the output of freebsd and dragonfly saved by default?

TODO: cover basics of what each does

TODO: mention configuration files

TODO: point to docs for more details

TODO: mention security script(s)

On OpenBSD, the daily job runs at 1:30 a.m. and the weekly job at 3:30 a.m. on Saturday.

On NetBSD, the daily job runs at 3:15 a.m. and the weekly job runs at 4:30 a.m.

On FreeBSD and DragonFly, the daily jobs run at 3:01 a.m. and the weekly job runs at 4:15 a.m.

The BSDs run the monthly job at 5:30 a.m. on the first day of the month.

TODO: note about no monthly by default on NetBSD.

Examples

Practice Exercises

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

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

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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:

```
$ uptime
6:17AM up 16 days, 12:28, 3 users, load averages: 0.18, 0.14, 0.09
```

The number of users is from the "utmp" database. (This is covered in section 4.10 .)

The time the system was booted is recorded in the kern.boottime sysctl. (The sysctl functionality is covered in section 5.7 .)

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

Examples

Practice Exercises

More information

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

5.21 Monitor disk input/output

Author: Ivan Voras IvanVoras FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Monitoring disk I/O can be crucial for troubleshooting a machine. There are a few common symptoms that may indicate an overtasked disk I/O systems: slow or delayed starting of applications and shells, slow remote logins, or slowness in the specific main task the machine is doing (e.g. e-mail server, database server, etc.). Thus it's important to reliably monitor access rates and throughput.

iostat

There are several ways this can be done, but the most common one is `iostat(8)`. When started without arguments it will display one or more header lines listing devices and a single statistics line that represents the current I/O performance of those devices. This single snapshot is often not a reliable indicator of true I/O performance and it's more useful to specify the `-w N` argument to `iostat` which tells it to display statistics in a loop, every `N` seconds. On a big machine, there may be more devices than fit the screen so `iostat` will by default display only 5 devices. The portable way to override this, usable on all BSD's is to specify device names on the command line, but FreeBSD has extended `iostat` with `-n N` argument which tells it to display at most `N` devices.

vmstat

The `vmstat` utility displays low-level information from the kernel. When started without arguments it will display a snapshot of statistics, but if called with `-w N` argument it will loop and

display a line of statistics every N seconds, similar to `iostat`. The specific information `iostat` displays differs among the systems but it usually includes the amount of free memory, number of page faults, memory paging activity (swap), and CPU stats. `vmstat` is important as it's a quick way to find out if the system's high I/O rates are due to memory swapping.

systat

The `systat` utility is more complex than those already mentioned, as it's a full-screen utility that is usually used for long-term performance tracking (for example: started on a spare console in the system room that is overseen by administrators). It has several display modes, which differ among BSD systems, but the common ones are `iostat`, `vmstat`, `netstat`, `mbufs`, `swap` and `pigs`. The display mode is specified directly on the command line (but prefixed with a `-` on FreeBSD).

- `iostat` mode shows I/O statistics similar to the `iostat` utility
- `vmstat` mode shows kernel statistics similar to the `vmstat` utility
- `netstat` mode shows network I/O statistics similar to the `netstat` utility
- `mbufs` mode shows network buffers statistics
- `swap` mode shows swap usage
- `pigs` mode shows processes with highest CPU usage

Different BSD systems have some useful additions to the list of display modes, for example FreeBSD has `ifstat` mode for per-network-interface statistics, and NetBSD has a `ps` mode that displays a list of processes. See specific man pages for more information.

nfsstat

The `nfsstat` utility shows NFS statistics. When started without arguments it will display a screen-full of information about NFS, but if called with `-w N` it will display a two line statics every N seconds (about client and server NFS usage).

gstat

The `gstat` utility is specific to FreeBSD. It's a full-screen utility requiring root privileges that shows I/O statistics for all GEOM devices, including virtual devices. With `gstat`, I/O can be monitored for individual disk partitions, virtual devices such as RAID geoms, memory drives and all other GEOM devices.

Examples

The following will continuously monitor I/O statistics for first two SCSI drives on FreeBSD:

```
> iostat -w 1 da0 da1
```


To see an overview of NFS performance, use:

```
> nfsstat
```

Practice Exercises

1. Start a “fork bomb” program (usually one can be found in ports/packages/pkgsrc of the system) and monitor how the system behaves with each of above utilities.

More information

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

5.22 Deal with busy devices

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Examples

Practice Exercises

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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

The `uname` command can display the type and version of the operating system installed, including the system's hostname, the release level, and the hardware platform name and the processor architecture name. The `-a` option will show many details, and is similar to calling `uname` with the `"-m -n -r -s -v"` options.

Note that usually the kernel and userland (like `libc`, BSD tools and network daemons, et cetera) are kept in sync so the same version would apply for the operating system as a whole.

TODO: is "sync" term okay here? Maybe improve sentence. And should this mention that `uname` is for kernel only?

By default, it shows the name of the operating system:

```
$ uname
DragonFly
$ uname -s
DragonFly
```

The machine's hardware name and processor architecture name can be displayed, respectively:

```
$ uname -m
i386
$ uname -p
i386
```

TODO: add note and example about when `-m` and `-p` are different

It can also provide the time and date of the built kernel, the host it was built on, and the path to the kernel configuration used, for example:

```
$ uname -v
DragonFly 1.7.0-DEVELOPMENT #0: Fri Oct 27 12:10:01 PDT 2006
jdoe@server5.example.org:/build/usr.obj/usr/src/sys/SERVER
```

The `sysctl` tool (introduced in section 5.7) can also show some of this same information, for example:

```
$ sysctl kern.ostype
kern.ostype = NetBSD
$ sysctl kern.hostname
kern.hostname = glacier.reedmedia.net
$ sysctl kern.osrelease
kern.osrelease = 3.99.24
$ sysctl kern.version
```

5.24. UNDERSTAND THE ADVANTAGES OF USING A BSD LICENSE

```
kern.version = NetBSD 3.99.24 (JCR20060802) #0: Mon Sep 25 12:22:43 CDT 2006
reed@new-host-8:/usr/src/sys/arch/i386/compile/JCR20060802
$ sysctl hw.machine
hw.machine = i386
$ sysctl hw.machine_arch
hw.machine_arch = i386
```

On NetBSD, an `/etc/release` file also gives further details to identify the source code used to build the system.

Practice Exercises

TODO

More information

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

5.24 Understand the advantages of using a BSD license

Author: *name ?? ??*

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

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

6 Chapter Network Administration

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

- 6
- 6.1
- 6.3
- 6.3
- 6.5
- 6.6
- 6.7
- 6.8
- 6.9
- 6.10
- 6.11
- 6.12
- 6.13
- 6.14
- 6.15

6.1 Determine the current TCP/IP settings on a system

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

Reviewer: Sean Swayze swayze@pcsa.biz FreeBSD

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

Concept

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

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.

Examples

Let's start from IP address and MAC address. We can get this kind of information from **ifconfig -a** 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
    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 0xfffff00 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 - ffffffff), 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  Expire
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  1178
```

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.

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.

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 ?? ??*

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

Reviewer: jdq ?? OpenBSD

Concept

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

Introduction

This chapter one of the most important because you as network administrator must know how to convert your hardware into real server. In this part you know how to set network settings in BSD box.

Examples

You can update IP on the fly manually and via DHCP server. You must be a **root** user for change IP and other network settings.

TODO: briefly mention DHCP and refer to section 6.14 . TODO: maybe move some of this to there?

```
# dhclient fxp0
DHCPREQUEST on fxp0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.162 - renewal in 302400 seconds.
```

Now we update our fxp0 interface IP via DHCP server. You may override default dhclient options in /etc/dhclient.conf. In environments where a static IP or DHCP is not available, you must manually configure a network interface:

```
# ifconfig fxp0 192.168.1.162 netmask 255.255.255.0
```

And check the status of this interface:


```
# ifconfig fxp0
fxp0: flags=8843 mtu 1500
    options=8
    inet 192.168.1.162 netmask 0xfffff00 broadcast 192.168.1.255
    ether 00:09:6b:13:42:9f
    media: Ethernet autoselect (100baseTX )
    status: active
```

The `-a` switch may be used to check the status of all interfaces.

You may save these settings on xBSD, yBSD and FreeBSD by adding an entry to `/etc/rc.conf` :

```
ifconfig_rl1="inet 192.168.51.50 netmask 255.255.255.0"
ifconfig_rl0_alias1="inet 192.168.231.2 netmask 255.255.255.0"
ifconfig_rl0_alias0="inet 10.1.1.1 netmask 255.255.255.252"
```

For OpenBSD, create an `/etc/hostname.rl0` for the rl0 interface in OpenBSD:

```
192.168.51.50 netmask 255.255.255.0 up""
inet alias 192.168.231.2 255.255.255.0""
inet alias 10.1.1.1 255.255.255.252""
```

To configure xBSD, yBSD, and FreeBSD to use dhcp on startup:

??

And in OpenBSD, simply use the following entry in `/etc/hostname.if`:

```
"dhcp NONE NONE NONE"
```

To bring up an interface according to the configuration files, use `netstart`:

```
# sh /etc/netstart rl0
```

DHCP servers often provide a default route. If dhcp is not in use, or a default route is not provided by the DHCP server, you must configure one manually. In xBSD, yBSD and FreeBSD, add:

```
defaultrouter="192.168.1.1"
```

to `rc.conf`. Similarly, in OpenBSD, add an entry to `/etc/mygate`:

```
echo 192.168.1.1 > /etc/mygate
```

you may check the routing table by using

```
# route -n show
```

The `-n` option discarding name resolution preventing long delays. You may add a default entry by keying in:

```
# route add default 192.168.1.1
```

For update DNS servers list you must update your **resolv.conf** file. This is typical file

```
# cat /etc/resolv.conf
nameserver 192.168.10.1
nameserver 10.10.10.13
```

TODO: don't cover DNS too much here.

For more information about DNS, see section 6.7 and ...TODO.

Firstly system try to resolve address with 192.168.10.1 and then with 10.10.10.13 (truly firstly with **hosts** file)

Practice Exercises

1. Try to change your IP address
2. Create alias for network interface
3. Add DNS server.

More information

hostname (1), ifconfig(8), route(8), resolv.conf(5), rc.conf(5), hosts(5), hostname.if(5), my-name(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 ?? ??*

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

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.

Introduction

Examples

Practice Exercises

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 ???*

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

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.

Introduction

Some times you need to check some network service on remote or local system. How can we do it? Firstly we must check the firewall rule, it's common error when someone check network service on remote host when your firewall block this type of request. And then you can start your test.

Examples

ping it's a most common tool for check availability host(remember some admins blocks ping). Instead of Windows system ping in BSD will ping host until you press Ctrl+C.

```
# ping freebsd.org
PING freebsd.org (69.147.83.40): 56 data bytes
64 bytes from 69.147.83.40: icmp_seq=0 ttl=47 time=216.248 ms
64 bytes from 69.147.83.40: icmp_seq=1 ttl=48 time=227.884 ms
```

Now we ping freebsd.org and know that this server is up. Let's check service like www and ssh

```
# telnet freebsd.org 80
Trying 69.147.83.40...
Connected to freebsd.org.
Escape character is '^]'.
^C
Connection closed by foreign host.
# telnet freebsd.org 22
Trying 69.147.83.40...
Connected to freebsd.org.
Escape character is '^]'.
SSH-2.0-OpenSSH_4.2p1 FreeBSD-20060930
^C
Connection closed by foreign host.
```

www is work on freebsd.org and ssh is too. Also we can see that freebsd.org support only ssh 2(if server support 1 and 2 it will show 1.99). Not so hard but very usefull. You can use **nc** tool for this - check the manual.

Imagine that ping is filed but your system work fine and network work too. What's happen? May be some host in your route to destination service is down. We can check this via **traceroute** tool.

```
# traceroute ya.ru
traceroute to ya.ru (213.180.204.8), 64 hops max, 40 byte packets
 1  192.168.1.1 (192.168.1.1)  0.431 ms  0.402 ms  0.351 ms
 2  vpn13-10.msk.corbina.net (10.1.1.1)  17.084 ms  22.738 ms  18.418 ms
 3  hq-bb-giga2-12.msk.corbina.net (85.21.151.113)  19.142 ms  15.753 ms  29.292 ms
 4  yandex-gw.corbina.net (85.21.52.222)  11.958 ms  18.624 ms  22.089 ms
 5  ya.ru (213.180.204.8)  21.116 ms  21.629 ms  20.597 ms
```

In this case all host in route to ya.ru is up.

Practice Exercises

1. Use **ping** and check some host for availability.
2. Use **telnet** and check some service for availability.
3. Use **traceroute** and check your route.

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: Sean Swayze swayze@pcsage.biz FreeBSD/OpenBSD

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.

Introduction

The Domain Name System (DNS) stores information mainly for mapping Internet host names to IP addresses and vice versa, as well as mail routing information. It can also store many other types of information not covered here, utilized by different Internet applications.

The DNS system stores data in a tree-like hierarchy starting from a root node, through subnodes to host node. Every node of the tree is called a *domain* and is given a label. Every domain except the root is also a *subdomain*. The *domain name* of the node is the concatenation of all the labels on the path from the node to the root node separated by dots. It is written starting from the host node on left, with the root node located on right, ie. mail.example.com where mail is a host node and example is a subdomain of a top level domain com.

Everytime we're buying a domain, we're actually buying a subdomain of one of the global domains like:

- generic top level domains **gTLD** (ie. com, org, net) or
- country code top level domains **ccTLD** (ie. uk, de, pl, fr).

It may even be some second level domain like co.uk, org.pl, net.de.

For administrative purposes, the name space in the DNS system is partitioned into areas called *zones*, each starting at a node and extending down to the leaf nodes or to nodes where other zones start. The data for each zone is stored in a *name server*, which answers queries about the given zone.

A zone consists of those parts of the domain tree for which a name server has complete information and over which it has authority. It contains all domain names from a certain point downward in the domain tree except those which are delegated to other zones. A delegation point is marked by one or more NS records in the parent zone, which should be matched by equivalent NS records at the root of the delegated zone.

Resource Records

The data associated with each domain name is stored in the form of resource records (RR). The most commonly met types of RRs in IPv4 networks are:

- **A** - a host IP address located in the IN class.
- **CNAME** - a canonical name identifier for creating aliases.
- **MX** - a mail exchange identifier for given domain.
- **NS** - the authoritative name server for the domain.
- **PTR** - a pointer to another part of the domain name space.
- **SOA** - identifies the start of a zone of authority.

Authoritative Name Servers

Each zone is served by at least one authoritative name server, which contains the complete data for the zone. Most zones have at least two authoritative servers to make the DNS system immune to server and network failures.

Responses from the authoritative servers have the “authoritative answer” (AA) bit set in the response packets.

The authoritative server where the master copy of the zone data is maintained is called the *primary master server*. It holds zones configured manually by an administrator. The other authoritative servers known as the *slave servers* or *secondary servers* load the zone contents from another server using a replication process known as a *zone transfer*.

Caching Name Servers

To improve performance, *recursive servers* cache the results of the lookups they perform on behalf of DNS clients. This may be a query of a web browser as well as of a **host(1)** command. The terms *recursive server* and *caching server* are often used synonymously, and for needs of this document they’ll be treated as synonymous.

Examples

(**Note:** Basic information on reverse DNS queries is covered in section 6.6)

In BSD systems there are three basic DNS query tools: **host(1)**, **dig(1)** and **nslookup(1)**.

host(1)

Performing general DNS lookup with **host(1)** is pretty straightforward:

```
$ host google.com
google.com has address 64.233.167.99
google.com has address 64.233.187.99
google.com has address 72.14.207.99
google.com mail is handled by 10 smtp1.google.com.
google.com mail is handled by 10 smtp2.google.com.
google.com mail is handled by 10 smtp3.google.com.
google.com mail is handled by 10 smtp4.google.com.
```

Sometimes it is required to check what information on given domain can be retrieved from some other name server. This can be done by specifying queried DNS server’s name or IP address after the domain name we’re gathering information on.

```
$ host google.com 192.168.86.1
Using domain server:
Name: 192.168.86.1
Address: 192.168.86.1#53
Aliases:
```

```

google.com has address 72.14.207.99
google.com has address 64.233.167.99
google.com has address 64.233.187.99
google.com mail is handled by 10 smtp2.google.com.
google.com mail is handled by 10 smtp3.google.com.
google.com mail is handled by 10 smtp4.google.com.
google.com mail is handled by 10 smtp1.google.com.

```

Of course, information on IP addresses and mail exchange servers is not always what we're looking for. Thus, querying for given resource record is also available – with the **-t** flag.

```

$ host -t SOA google.com
google.com has SOA record ns1.google.com. dns-admin.google.com. 2007010801 7200 1800 1209

```

Finally, we'd like to get as much information as possible.

```

$ host -a google.com ns2.google.com
Trying "google.com"
Using domain server:
Name: ns2.google.com
Address: 216.239.34.10#53
Aliases:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44983
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 4, ADDITIONAL: 8
;; QUESTION SECTION:
;google.com.                IN      ANY
;; ANSWER SECTION:
google.com.                 300    IN     A      72.14.207.99
google.com.                 300    IN     A      64.233.187.99
google.com.                 300    IN     A      64.233.167.99
google.com.                 300    IN     TXT    "v=spf1 ptr ?all"
google.com.                 10800  IN     MX     10 smtp1.google.com.
google.com.                 10800  IN     MX     10 smtp2.google.com.
google.com.                 10800  IN     MX     10 smtp3.google.com.
google.com.                 10800  IN     MX     10 smtp4.google.com.
google.com.                 345600 IN     NS     ns1.google.com.
google.com.                 345600 IN     NS     ns2.google.com.
google.com.                 345600 IN     NS     ns3.google.com.
google.com.                 345600 IN     NS     ns4.google.com.
google.com.                 86400  IN     SOA    ns1.google.com. dns-admin.google.com. 200
;; AUTHORITY SECTION:
google.com.                 345600 IN     NS     ns1.google.com.
google.com.                 345600 IN     NS     ns2.google.com.
google.com.                 345600 IN     NS     ns3.google.com.
google.com.                 345600 IN     NS     ns4.google.com.

```

```
;; ADDITIONAL SECTION:
smtp1.google.com.      3600    IN      A       216.239.57.25
smtp2.google.com.      3600    IN      A       64.233.167.25
smtp3.google.com.      3600    IN      A       64.233.183.25
smtp4.google.com.      3600    IN      A       72.14.215.25
ns1.google.com.        345600  IN      A       216.239.32.10
ns2.google.com.        345600  IN      A       216.239.34.10
ns3.google.com.        345600  IN      A       216.239.36.10
ns4.google.com.        345600  IN      A       216.239.38.10
Received 494 bytes from 216.239.34.10#53 in 100 ms
```

Please notice the presence of AA bit in response packet:

```
;; flags: qr aa
rd; QUERY: 1, ANSWER: 13, AUTHORITY: 4, ADDITIONAL: 8
```

The **host(1)** utility – as simple as it may seem – allows also performing a zone transfer using AXFR query.

```
$ host -t AXFR google.com ns2.google.com
Trying "google.com"
Using domain server:
Name: ns2.google.com
Address: 216.239.34.10#53
Aliases:
Host google.com not found: 5(REFUSED)
; Transfer failed.
```

As we can see, queried name server refused a zone transfer which marks potentially well configured DNS server.

dig(1)

To be continued...

Practice Exercises

1. Check the result of **host(1)** command along with flags: **-d** , **-v** , **-C** , and **-l** .

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 FreeBSD

Reviewer: ceri ceri@FreeBSD.org FreeBSD|OpenBSD

Reviewer: *name ???*

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.

Introduction

(Note: Basic information on DNS system is covered in section 6.5)

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

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. Both the **dig(1)** and **whois(1)** commands can be used for this purpose.

A reverse DNS lookup can be performed using the **-x** flag to the **dig(1)** command. The information that we're looking for is located within the SOA record.

```
# 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. dns-admin.google.com. 2007010400
;; 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
```

One of the pieces of information obtained with this command is a contact e-mail address for the person responsible for a given DNS zone. This is located just after the hostname of the

primary DNS server for the zone and is shown with a . (dot) instead of the usual @ character. In this case it is dns-admin.google.com which should be read as dns-admin@google.com.

The **whois(1)** command does not require any additional parameters to perform a lookup and it provides far more detailed contact information.

```
# whois 216.239.32.10

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 format of the whois output depends on many factors, such as the registry for the IP address block, but each gives similarly detailed information. Notice also, that the information gained from a whois query on an IP address may differ from the information gained when querying a domain name pointing to the very same IP address. Most commonly this occurs when the domain is administered by a different organization than the IP address block.

Practice Exercises

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

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* ?? ??

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

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.

Introduction

Examples

Practice Exercises

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: ?? andreas dot kuehl at clicktivities dot net ?? FreeBSD

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

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

Concept

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

Introduction

All of 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 $255*255*255$ (16,581,375) addresses, a class B net had the first two bytes common and contained $255*255$ (65,025) addresses while as you can 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 of the same address.

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 obtained a class C net for your company and have to divide it for several purposes...

(To be continued :-)

Examples

Practice Exercises

More information

<http://www.faqs.org/rfcs/rfc791.html> <http://www.faqs.org/rfcs/rfc1519.html> [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: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

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

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.

Introduction

Configuring BSD to work in network environment requires proficiency in operating on IP addresses and subnet masks. To understand different subnet mask's notations used throughout this section please refer to ?? .

Examples

Having a host with an IP address of 192.168.1.25/24 we can determine without any further calculations:

```
subnet identifier:           192.168.1.0
broadcast address:          192.168.1.255
number of valid host addresses: 254
```

The subnet mask of /24 in CIDR notation stands for 255.255.255.0. The IPv4 uses a 32-bit IP addressing and network masks, which means that the above subnet mask have last eight bits set to zero ($32 - 24 = 8$), which also mean that we have 2^8 IP addresses available. Subtracting two addresses for subnet address (first address) and broadcast address (last address) we get number of valid hosts:

$$2^8 - 2 = 256 - 2 = 254$$

As for the address of 192.168.1.25/28 the paremeters in question are:

```

subnet identifier:          192.168.1.16
broadcast address:         192.168.1.31
number of valid host addresses: 14

```

This is quite more interesting example. Having a CIDR subnet mask of /28 (four bits set to zero)—which is 255.255.255.240 in dotted decimal notation—means that there are 16 addresses available, and only 14 of them are valid host addresses:

$$2^4 - 2 = 16 - 2 = 14$$

For networks smaller than 256 IP addresses there's a simple way to determine number of available addresses using a 'neat trick' as described in Daryl's TCP/IP Primer (link at the end of this subject). To do so we can simply subtract the last number of the subnet mask from 256. For aforementioned subnet mask of 255.255.255.240 we'll have $256 - 240 = 16$ addresses. Now, dividing the result into 256 we can determine the number of subnets ($256 / 16 = 16$), which gives us a 16 subnets of 16 addresses each. The scope of the first one is 192.168.1.0 - 192.168.1.15, the second one 192.168.1.16 - 192.168.1.31 and so forth. Our IP address of 192.168.1.25 is located within the second one.

To make it all more confusing let's try to determine subnet's parameters having a 192.168.1.25/22 IP address:

```

subnet identifier:          192.168.0.0
broadcast address:         192.168.3.255
number of valid host addresses: 1022

```

The CIDR subnet mask of /22 gives us 10 bits defining the hosts addresses.

$$2^{10} - 2 = 1024 - 2 = 1022$$

This gives us networks with a scope of IP addresses: 192.168.0.0 - 192.168.3.255, 192.168.4.0 - 192.168.7.255, etc.

In closing, the partial reference table on IPv4 subnets:

CIDR	Netmask	Addresses
/18	255.255.192.0	16384
/19	255.255.224.0	8192
/20	255.255.240.0	4096
/21	255.255.248.0	2048
/22	255.255.252.0	1024
/23	255.255.254.0	512
/24	255.255.255.0	256
/25	255.255.255.128	128
/26	255.255.255.192	64
/27	255.255.255.224	32
/28	255.255.255.240	16
/29	255.255.255.248	8

/30		255.255.255.252		4
/31		255.255.255.254		2
/32		255.255.255.255		1

Note that the /32 subnet mask actually points to only one host, and /31 subnet is simply useless as there are no addresses left for hosts.

Practice Exercises

1. Determine subnet identifier, broadcast address and number of valid host addresses having: 192.168.86.2/24, 10.0.9.7/26, 192.168.159.8/25, and 172.16.0.189/18.

More information

<http://www.ipprimer.com/bitbybit.cfm>

6.10 Understand IPv6 address theory

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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 host-bits). 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.

Introduction

Examples

Practice Exercises

More information

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

6.11 Demonstrate basic tcpdump(1) skills

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

You are having problems connecting to an application server that is on the network. What to do and how do you start. One place to start is to see what traffic is going between the nodes. Tcpdump is a utility that comes with *BSD that will enable you to see what traffic is happening.

Examples

Let's say that you know that there are people having trouble getting a DHCP address on the network but there are more than one person having problems so now you wonder if it is the server that isn't responding. Or perhaps it's a problem on the network itself. Using the command tcpdump dst port bootpc we can see what traffic is happening.

```
[root@pmax namedb]# tcpdump dst port bootpc
tcpdump: listening on le0
12:14:03.941390 pmax.smithclan.prv.bootps > dhcp-ip97.smithclan.prv.bootpc:  xid:0
```

Practice Exercises

More information

tcpdump(1)

6.12 Manipulate ARP and neighbor discovery caches

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

arp(8), ndp(8)

6.13 Configure a system to use NTP

Author: Grzegorz Czapliński gregory at systemics dot pl FreeBSD

Reviewer: Cezary Morga cm@therek.net FreeBSD

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

6.14 View and renew a DHCP lease

Author: *name ?? ??*

Reviewer: *name ?? ??*

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

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.

Introduction

Examples

Practice Exercises

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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

Introduction

Multiple IP addresses can be assigned to a network interface using the ifconfig tool using the “alias” parameter. (An introduction to ifconfig is in sections 6 and ?? .)

Interface aliases are useful for doing virtual hosting , changing network addresses, or for accepting packets for old interfaces.

TODO: Note: do not use “down” for an alias as it takes the entire interface down.

TODO: mention setting up route? examples? what systems need that route done also?

Examples

Practice Exercises

More information

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

7 Chapter Basic Unix Skills

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: Yannick Cadin 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.

- 7.1
- 7.1
- 7.3
- 7.4
- 7.5
- 7.7
- 7.8
- 7.9
- 7.10
- 7.11
- 7.12
- 5.14
- 7.13
- 7.14
- 7.6
- 7.14
- 7.16

7.1 Demonstrate proficiency in using redirection, pipes and tees

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

7.2 Recognize, view and modify environment variables

Author: *Ivan Voras IvanVoras FreeBSD*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

Concept

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

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.

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.

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.

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)

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.

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

More information

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

7.3 Be familiar with the vi(1) editor

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

ex, Bill Joy.

??

The vi editor is a screen oriented text editor. ex is a line oriented text editor. Both ex and vi are different interfaces to the same program.

FreeBSD, NetBSD and OpenBSD all use the nex/nvi versions of the ex/vi text editors, these are bug-for-bug compatible replacements for the original Fourth Berkeley Software Distribution (4BSD) ex and vi programs.

As stated above vi is a screen editor, in practice this means that it takes almost the entire screen. The screen is mainly a display of the lines in a file, except for the last line which is used for you give commands to vi. vi is a modal editor. This means that you are either entering commands or entering text. You need to be in the correct mode to do one or the other.

Examples

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

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: Ivan Voras IvanVoras FreeBSD

Reviewer: *name* ?? ??

Reviewer: *name* ?? ??

Concept

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

Introduction

File types are really not well defined in Unix, and (not going into discussion about special file-like system objects) there are really only three types of files that are recognized by the system:

1. Executable files, distinguished by having the execute (“x”) bit set
2. Directories, noted by their directory (“d”) bit
3. Everything else

The third category can really encompass anything - regular text files, images, multimedia, archives, etc. Files from all categories are not distinguished on the system level by their name (this is different from other architecture, for example Microsoft(r) Windows(tm)) but there are conventions that help users not to get lost in the listings. The most used convention is adding a file extension - a sequence of characters prefixed with dot (“.”) to the file name. Thus most shell scripts have filenames ending with .sh, readable text files end with .txt, JPEG images with .jpeg, etc. This convention can sometimes fail for various reasons, most common of which is if a file is copied from a system that doesn’t support appropriate file attributes or has filename limitations.

To help recover file type information there’s a database of detection strings (/usr/share/misc/magic) and a utility (file(1)) that are used together to inspect files and produce human readable description of its content. Because there can be infinite file types, this method cannot always work, but will probably work for nearly 100% of commonly used files.

If you’re familiar with several widely used formats you may inspect the file yourself, for example by converting it to a hex dump (with hexdump(1)) and looking at the first few lines.

Examples

To verify that the “magic” database is indeed what it’s supposed to be, use:

```
> file /usr/share/misc/magic
/usr/share/misc/magic: magic text file for file(1) cmd
```

To verify the format of an executable, use:

```
> file `which cat`
/bin/cat: ELF 32-bit LSB executable, Intel 80386, version 1 (FreeBSD), dynamically
> file `which acroread`
/usr/X11R6/bin/acroread: a /compat/linux/bin/sh script text executable
> file /compat/linux/bin/bash
/compat/linux/bin/bash: ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Lin
```

To inspect the format of a random file:

```
> file a_file_i_found
a_file_i_found: JPEG image data, JFIF standard 1.02
```

To see for yourself what does the header of a file look like, use `hexdump -C` piped to `head`:

```
> hexdump -C zlib1.dll | head
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ.....|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |.....@.....|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |.....|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 f8 00 00 00  |.....|
00000040  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68  |.....!..L.!Th|
00000050  69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode...$......|
00000080  bb 22 a6 bc ff 43 c8 ef  ff 43 c8 ef ff 43 c8 ef  |"...C...C...C..|
00000090  7c 4b 95 ef fd 43 c8 ef  ff 43 c9 ef e7 43 c8 ef  ||K...C...C...C..|
```

Practice Exercises

1. Find out the file type of your kernel (in FreeBSD, it's `/boot/kernel/kernel`)
2. Find a unicode (UTF-16) text file and use `hexdump` to examine it. Compare the results with an ASCII text file.

More information

`file(1)`, `magic(5)`

7.5 Locate files and binaries on a system

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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, manpages and files. In addition, be able to update the `locate(1)` database.

Introduction

By default, this database is built every Saturday. (See section 5.19 for details on weekly maintenance tasks.)

TODO: show how to manually build this database

If you want to find a file in real-time, you can use the find command. TODO: basic usage and easy examples

More information on using find is covered in section 7.7 .

Examples

Practice Exercises

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 Overcome command line length limitations

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

xargs(1), find(1)

7.7 Find a file with a given set of attributes

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

`find(1)`

7.8 Create a simple Bourne shell script

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

`sh(1)`, `chmod(1)`

7.9 Find appropriate documentation

Author: Kevin D Kinsey kdk_at_daleco_dot_biz FreeBSD Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

TODO: there is another section about this, so need to make sure they are not covering same details. TODO: if one section should be removed, we can just merge the concept and content together TODO: 7.10

Introduction

Some people say BSD systems “aren’t user-friendly”; others reply, “BSD *is* user-friendly; it’s just picky about who its friends *are* .” Consider this section a lesson in making friends with your BSD system.

From the earliest days of their existence, BSD systems have made extensive use of program documentation through “manpages”, that is to say, the system’s built-in manual pages. While the interface may not be familiar to people who have no background in Unix-like systems, the “man” system is exceptional in the amount of and quality of documentation available, and ease of access.

Some advantages of manual pages: 1] they include a brief synopsis of the command syntax, flags, and options at the very top of the manual page; 2] They are typically written by the developers or the development team themselves, so they tend to be complete and accurate (more than some online “HOWTO’s”, for example); 3] “manpages” are available on the system locally and from the basic terminal, and so they aren’t subject to network interruptions or the current state of the GUI.

If you *are* running a GUI, then you can have your manpages put into a GUI window (try “xman”). Also, note that it is possible to install a BSD system *without* manual pages; if typing “man man” at your shell prompt produces nothing (or an error), consult with someone about getting them installed.

RTFM! (Read the *friendly* manual)

A “manpage” exists for almost every program, device, library, system call, and configuration file on the system. If you are aware of a program (let’s say “kill”), simply typing “man kill” at the shell’s prompt will present you the documentation for kill(1), piped through your \$PAGER (sometimes this is more(1), but it is often less(1) — don’t get confused, because less(1) is really more(1)).

In addition, many programs that are “contributed” software (for example, those from the Free Software Foundation) also have “Info Pages”, which are displayed with (you guessed it) “info foo” (where “foo” is the program name). If the software is FSF software, the “info” pages have a slightly different format; otherwise, “info” simply displays the manual page in the same way that `man(1)` does.

Manual Sections

But I Don’t Know What I Want to Read!

In the event that you can only remember a portion of the file, try using `whatis(1)` or `apropos(1)`. These programs search a database of installed documentation for the string you pass in as an argument. `Whatis` returns fewer results (only exact matches), and `apropos` returns any manual page references that contain the string at all (so sometimes it’s a good idea to pipe the output of `apropos(1)` through your `$PAGER`).

The Manual is Too Cryptic!!!

The manual intends to describe every program with enough detail to describe *all* of the behaviors and options available to the user. It might be criticized because many manpages lack examples in sufficient quantity to apply to every conceivable situation (an impossible task). In addition, occasionally software developers will “skimp” on the manual page in favor of documentation in other formats. If you find a manual page to be unsatisfying.

Examples

Practice Exercises

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 online handbook and several mailing lists

7.10 Recognize the different sections of the manual

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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

Examples

Practice Exercises

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

More information

man (1), intro(1) to intro(9), “/”

TODO: why “/” in this more information?

7.11 Verify a file's message digest fingerprint (checksum)

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

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

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.

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.

Practice Exercises

Get few fingerprints from your system via md5 and sha1

More information

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

7.12 Demonstrate familiarity with the default shell

Author: *name ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Examples

Practice Exercises

More information

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

7.13 Use job control

Author: Kevin D. Kinsey kdk_at_daleco_dot_biz FreeBSD

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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”.

Introduction

Some jobs run by the shell are “small jobs” — quick and easy for the machine, with virtually no waiting for the user. Consider these small files and the use of grep(1):

```
$ ls -l
-rw-r-r-  1 myuser  wheel    315 Jan 19 13:00 notes.txt
-rw-r-r-  1 myuser  wheel   1967 Jan 18 13:49 otherstuff
-rw-r-r-  1 myuser  wheel  6335 Jan 11 23:11 packagelist
-rw-r-r-  1 myuser  wheel 14764 Jan 23 13:45 spammers
-rw-r-r-  1 myuser  wheel  2678 Jan 23 13:46 spamstuff.txt
$ grep speakeasy *
spammers:.dsl.speakeasy.net
```

The job is finished almost instantly, and your shell returns control to you. On the other hand, consider this:

```
# tar -c -z -f /backup/src.tar /usr/src/*
```

Depending on your system, it could take a *very* long time for you to make a gzipped archive of your BSD's project source tree! So, you can wait and twiddle your thumbs, or you can use job control to have the shell “put the job in the background” and return to your prompt so you can keep working.

Job Control

All modern shells feature job control. In the BSD's, the standard user's shell is `cshtcsh`, but the information presented here should apply equally well to other shells, except as noted. The manpage for `tcsh(1)` has an entire section on job control, which should be read as an additional resource to clarify and extend this section.

With a job-control shell, you can start a job so that it runs in the background, see information about currently running “backgrounded” jobs, move jobs from the foreground to background or vice-versa, or terminate them abruptly.

“Backgrounding” and “Foregrounding” jobs

The shell keeps a list of (backgrounded) jobs, including their status, which can be queried with the “jobs” shell built-in. If/when the job exits, the shell will report this, along with the job's exit status, prior to the next shell prompt.

To start a job in the background, end the command line with “&”. To see running jobs, type “jobs”. You can then use the job numbers in conjunction with “bg %” or “fg %” to move jobs from foreground to background and vice-versa.

If you start a long job in the foreground and then realize you forgot the “&”, you can suspend the job with CTL-Z, then issue “bg” to the shell and the job will continue in the background.

See the Examples section for details.

A word about “kill”

A job control shell usually has a built-in ‘kill’ command; a problem can ensue when this ‘kill’ is confused with `kill(1)`.

For example, in the `tcsh` shell you should use a “%” sign to indicate a job number to the shell; otherwise it may be confused with a process ID in the system. While “kill %1” would simply terminate the first backgrounded job, “kill 1” would send a TERM signal to the process with PID 1 (usually `/sbin/init!`), and would probably have the same effect as calling `shutdown(8)`. As a “normal” user, this would probably not be an issue, but if you were to do this as root, you might cause some problems. (Chalk this up as yet another reason not to do normal work as “root”).

Most other shells also have a “kill” built-in. If you use another shell, try “type kill” at the prompt. If the shell doesn't answer “/bin/kill”, then it will probably say something similar to “kill is a shell built-in”. If your shell has a built-in kill, check your shell's manpage for details on using “kill” under your shell.

Redirecting job output

If you intend to use job control it is useful to know about output redirection, because, under most circumstances, jobs that produce output will continue to do so, potentially “cluttering up” your terminal and whatever your “next project” is. See ?? for details.

Examples

```
$ sh /scripts/mylongscript.sh &
sh /home/myuser/scripts/mylongscript.sh &
[1] 10394
```

Execute “mylongscript.sh” in the background. The shell reports the command, the job number, and the job’s PID on the system.

```
$ tar -c -f /backup/src.tar /usr/src/*
tar: Removing leading '/' from member names
```

Oops! There’s that long job again, and we forgot to background it. While “stuck” waiting, press CTL-Z:

```
^Z
Suspended
```

Now you have your prompt back, so issue “bg”:

```
$ bg
[2] tar -c -f /backup/src.tar /usr/src/COPYRIGHT /usr/src/LOCKS ... &
```

Now run “jobs” and you should see both “backgrounded” tasks in the list:

```
$ jobs
[1] + Running sh /home/myuser/scripts/mylongscript.sh &
[2] - Running tar -c -f /backup/src.tar /usr/src/COPYRIGHT /usr/src/LOCKS ..
```

But wait! Suppose a colleague is already making an archive of the source tree, so we don’t need to.

```
$ fg 2
tar -c -f /backup/src.tar /usr/src/COPYRIGHT /usr/src/LOCKS ... &
^C
```

A quick CTL-C, and we save lots of CPU cycles, and disk space, too. Of course, in csh/tcsh, we could have just called “kill” and obtained the same basic result:

```
$ kill %2
```

Practice Exercises

1. Find a “long” job that needs to be run, and do it in the background with “&”. (Optionally, use redirection to make sure any output goes to a file or to the “bit-bucket”.)
2. Call the job to the foreground, then terminate it with CTL-C. (Do this fairly early!)
3. Start the job again in the foreground, then use CTL-Z to suspend the job. Then issue “bg”.
4. While the job (possibly more) is running, call “jobs” and then “jobs -l” (or “jobs -ls” in some shells). Study the differences in the output.
5. Use kill to terminate the job. Remember that in some shells, you must use “%” (“kill %1”). In others, it may be necessary to obtain the job’s PID and use this as a flag to kill(1).

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 ?? ??*

Reviewer: *name ?? ??*

Reviewer: *name ?? ??*

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.

Introduction

Examples

Practice Exercises

More information

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

7.15 Understand various “domain” contexts

Author: Ivan Voras IvanVoras FreeBSD

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

Reviewer: *name ?? ??*

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.

Introduction

All “domains” that we’re dealing with here are different ways of grouping certain types of information together. In particular:

- NIS, Kerberos and NTLM domains deal with system management and security - each of these allows managing system users and groups from a central location / repository that’s located on dedicated servers. Machines belonging to one of these domains query the central server for security clearance and user information.
- DNS is is a system that assigns human readable names to IP addresses. DNS names form a hierarchy in which each system’s fully qualified domain name (FQDN) is formed from the domain name part and a single system name part, and the domain names can be nested.

Examples

DNS name are hierarchical and nested; thus the name:

```
www.servers.example.com
```

refers to a machine called “www” in the domain “servers.example.com” which is nested in “example.com” which is itself nested under “.com”. The nslookup tool can be used to inspect DNS names:

```
> nslookup www.google.com
Server: dns.server.local
Address: xxx.xxx.xxx.xxx
Aliases: xxx.xxx.xxx.xxx.in-addr.arpa
Non-authoritative answer:
Name: www.l.google.com
Addresses: 216.239.37.104, 216.239.37.99
Aliases: www.google.com
```

Note that high traffic sites have multiple computers answering to the same DNS name, in order to help performance (as demonstrated in the above example). DNS databases actually contain several types of records. The most common are “A” records which are widely used to access generic resources, but arguably equally popular are “MX” records that hold addresses of e-mail servers for specific domains:

```
> nslookup
Default server: dns.server.local
...
> set type=mx
> gmail.com
Non-authoritative answer:
gmail.com preference = 50, mail exchanger = gsmtpl83.google.com
gmail.com preference = 5, mail exchanger = gmail-smtp-in.l.google.com
gmail.com preference = 10, mail exchanger = alt1.gmail-smtp-in.l.google.com
gmail.com preference = 10, mail exchanger = alt2.gmail-smtp-in.l.google.com
gmail.com preference = 50, mail exchanger = gsmtpl63.google.com
Authoritative answers can be found from:
gmail.com nameserver = ns2.google.com
gmail.com nameserver = ns3.google.com
gmail.com nameserver = ns4.google.com
gmail.com nameserver = ns1.google.com
```

A Windows NT domain (NTLM) name is formed by two backslashes followed by a case-insensitive name containing no spaces, for example:

```
\\MYCORP
```

Computers and users on the NTLM domain can be referenced either by appending a backslash and the username to the domain name or by using the (misused in this case) standard unix notation user@domain:

```
\\MYCORP\joe
joe@mycorp
```

Practice Exercises

1. Try several lookups of www.google.com with nslookup and compare results
2. See how many mail servers yahoo.com has

More information

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

7.16 Configure an action to be scheduled by cron(8)

Author: *name ?? ??*

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

Reviewer: *name ?? ??*

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.

Introduction

The cron daemon starts at boot time and is always running. Every minute it checks for updated configurations – called a crontab – and runs the jobs that match the specified time.

The system-wide cron table is commonly located at `/etc/crontab`. And user crontabs are stored under `/var/cron/tabs` directory.

Note that NetBSD and OpenBSD keep their default system crontab in the `/var/cron/tabs/root` file.

A single crontab configuration is generally placed on one text line. Space or tab delimited, it defines the minute, hour, day of month, the month, the day of week and the shell command to execute. A pound sign (#) at the beginning of a line starts a comment. A comment can not be on the same line as a crontab.

An asterisk (*) matches all ranges (first to last) for a time specification.

The `/etc/crontab` format also includes the name of the user to run the command as. The per-user crontabs do not have the user field.

TODO: show a few examples from default cron and show some more examples and explain

```
#minute hour    mday    month    wday    who    command
# rotate log files every hour, if necessary
0      *      *      *      *      root   newsyslog
# run weekly maintenance script every Saturday morning and save and email output
30     3      *      *      6      root   mask 077; /bin/sh /etc/weekly 2>&1
```

Note if this file was in the per-user crontabs, it would not have the “who” (root) field.

Note: if both the day of month field and the day of week fields are defined, then the job will run on both of these scheduled times. TODO: for example ...

A crontab can also define shell environment variables, for example TODO

Using crontab(1) to edit user cron table

TODO: note about setuid or setgid, note that OpenBSD does it different? Maybe this doesn't matter for this book

Examples

Practice Exercises

More information

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

Index

`/var/cron/allow`, 152
`/var/cron/deny`, 152

accounting, 74
alias, command, 146
alias, interface, 130
ARP, 129
audit-packages, 12
autoconfiguration, IPv6, 127

background, 146
bash, 135
Bourne shell, 141

case, 141
chmod, 50
compression, 91
CPU, 76
crontab, 152
csh, 135
cwd, 49

DHCP, 130
dig, 121
DNS, 150
documentation, 142
domain, 150

editing, 136
ELF, 138
environment variables, 134
exec, 141
expressions, 149

file types, 137
find, 141
fingerprint, 145

flavors, 6
for, 141
foreground, 146

gstat, 104

history, 146

if, 141
ifconfig, 130
installation, 2
iostat, 103
IPv6, 127, 129

jobs, 146

Kerberos, 150
kill, 77
killall, 79

lease, DHCP, 130
locate, 139
logging, 91

MAC, 127
magic, 137
mailbox, 95
maildir, 92
man pages, 144
manual, 144
mbox, 92
media, 6
message digest, 145
metacharacter, 149
mk.conf, 10
MTA, 92
MUA, 95

neighbor discovery cache, 129
networking, 128
nfsstat, 104
nice, 77
nice-level, 77
NIS, 150
nslookup, 150
NTLM, 150

package, 12
packages, 9, 10
performance, 103
pgrep, 79
PID, 77
pipe, 134
pkgsrc, 9, 10
pkill, 79
port, TCP or UDP, 128
portaudit, 12
ports, 9, 10
processes, 146
ps, 76

regular expressions, 149
release-map, 6
RELENG, 6
renice, 77

scripts, 141
sh, 135
shebang, 141
shell, 134
SIGKILL, 77, 79
signals, 79
SIGTERM, 77, 79
standard error, 134
standard input, 134
standard output, 134
subshells, 135
sysinst, 6
sysinstall, 6
systat, 77, 104
system accounting, 74

TCP, 128

tcsh, 135
terminal, 74
time, network, 129
top, 76
tty, 74

UDP, 128
unionfs, 56
upgrade, 6

variables, 134
virtual hosting, 130
vmstat, 103
vuxml, 12

while, 140, 141
whois, 121
www.bsinstaller.org, 6

xargs, 140