Up and Running

1. **First Time Install Aids -** OpenBSD & GNU
    - Introduction
    - Configuring Removable Storage Devices (CDs, Zip Drives, etc.)
    - Adding additional Packages
    - Adding a New User with root access privileges
    - Changing details of a User
    - Configuring bash
    - Afterboot Install:
        - Date. Setting the Date & Time
        - TimeZone. Setting the Time Zone
        - Network. Setting the basic network services.
        - Daily, Weekly, Monthly Scripts
    - Miscellaneous:
        - Making it easier to find files
        - Booting in Single User Mode
        - Moving Directories Safely
        - General Tools I install

2. **X a friendly Window on Unix** - XFree86, KDE, & vnc
    - Introduction
    - Installing needed X-Files
    - Allowing X to run (kernel config)
    - Determining your System Configuration
    - Configuring the base X environment
    - Quick Troubleshoot - mouse not working
    - Booting OpenBSD straight into X
    - KDE X window manager and OpenBSD 2.7
    - Setting KDE as default desktop
    - Vnc Remote Administration - in X11

3. **Multibooting** - Living with another OS on the drive
    - Introduction
    - Partitioning the Hard Disk
    - Installing OpenBSD - fdisk
    - Disklabel
    - Installing OS Boot Selector
    - OS-BS 2.0Beta8
    - mattsoft Boot Manager
    - Partition Magic 5.0
    - NTFS - Windows NT 4 / 2000 and OpenBSD
    - Relative Reference

4. **Mail Services** - Sendmail, pop, imap
    - Introduction
    - Process Queued Mail and Recieve incoming Mail
    - the inetd (alternative)
    - Processing pop requests
    - using the IMAP Toolkit (alternative)
    - Sendmail Configuration

- who am i?
- slow startup - gethostbyname() blocks
- Relaying Access Denied
- Simple Diagnostics
- What's in the QUEUE
- Debug and Verbose Mode
- Looking up MX Records

Server Services

5. **File & Printer sharing, MS Windows** – Samba
    - Introduction
    - Installing the Packaged Version
    - Starting samba with each reboot
    - Starting through inetd
    - Testing the installation
    - SWAT - The Samba Web Administration Tool
    - Adding Users
    - Rolling your own Samba Server
    - Co-habiting with Windows NT PDC
    - Adding the Samba Server to the Primary Domain Controller
    - Joining the Samba server to the Primary Domain Controller
    - Updating the /etc/samba/smb.conf
    - Using stunnel to secure SWAT password communications

6. **Database Server** - mySQL
    - Introduction
    - Installing
    - Testing the Installation
    - Starting MySQL
    - Stopping MySQL
    - Usability Assistance Tip
    - Related Reference
7. **FTP** - Setting up a secure ftp server - ftpd
    - Introduction
    - Configure ftp Login
    - Configure Directory ownership, permissions
    - Restrict User Access
    - Enable ftpd through /etc/rc.conf
8. **DNS Server** - named
    - Introduction
    - Preliminary Information
    - Starting named
    - named.boot - Configuring DNS
    - resolv.conf - name resolution path
    - db.mydomain.com.zone - Authoritative forward lookup
    - db.mydomain.com.rev - Authoritative reverse name lookup
    - db.localhost.zone - The special localhost ip
    - db.localhost.rev - reverse name lookup on localhost
    - db.all-zero.rev - reverse name lookup on 0.0.0.0 address
    - db.all-one.rev - reverse name lookup no 255.255.255.255 address
    - root.cache

Creating Dynamic Web Services

9. **Web Services** - Apache
    - Introduction
    - Setting Apache to start every time the system is started/restarted
    - Manually starting Apache
    - Testing that it works
    - Setting some status configurations
    - Creating User personal web pages
    - Restart apache
    - Create public_html in user accounts
    - Access user accounts with the URL form http://server-name/~user-id/
    - Securing the Site with SSL

10. **Server Scripting PHP** - php3
    - Introduction
    - Installing php3
    - compiling from source
    - configuring apache
    - testing the installation

11. **Horde/PHPLib** - horde, PHPLib
    - Introduction
    - Installing
    - Configuring horde/PHPLib
    - Configuring Apache
    - Testing the horde installation
    - Testing the PHPLib installation

12. **Webmail** - IMP
    - Introduction
    - Pre-requisites
    - Installing
    - Configuring IMP
    - Securing the Installation
    - Customizing IMP
    - Introduction
    - The Cover Page
    - The Page Title

13. **Web Group Ware** - TWIG
    - Introduction
    - Requirements
    - Extracting the Distribution Files
    - Configuring Apache
    - Configuring MySQL
    - Configuring TWIG
    - Basic configuration
    - PHPLib conflict problems

- Testing TWIG
- Related References

Works in Progress

14. **Firewalls** - Keeping the bad sorts out - ipf & ipnat
15. **Restricted SuperUser access** - sudo
16. **Secured Communications** - ssh & ssl
    - Introduction
    - Self-signed Certificates
    - Remote Access with ssh
    - Configuring ssh
    - Configuring sshd
    - Copying a file through SSH

17. **Web Caching/Proxying** - squid
    - Introduction
    - Installation
    - Starting Squid
    - Transparent Proxy
    - Access Controls (ACLS)
    - Cache Utilization Analysis Tools

Introduction

The installation instructions that comes with OpenBSD is pretty much straight forward. If you bought the CD then it will be a nicely printed CD sleeve, clear instructions. If you've downloaded the files from the Internet then read the INSTALL.architecture file (for example if you are installing it on an Intel class machine, then the file to read is INSTALL.386)

Outlined here are additional installation items that is likely to be helpful for someone new to OS installations or has come from another Unix. For those really new to Unix I suggest you read the complete section you are interested in before attempting to follow the instructions.

The initial purpose of this documentation was to record what I had to do to get OpenBSD into a workable configuration. A few of my friends wanted to try out Unix so here evolves my notes for my better understanding and for others new to OpenBSD.

Warning: If you are not familiar with using the vi text editor, or similar variants on OpenBSD (ex, view) I would suggest that it will make life much easier for you if you find a tutorial on "vi" somewhere on the 'net and get familiar. Most things in Unix requires editing text files, and it takes a while to get a graphical system up and running so editing usually requires a character based editor (like vi).

Documentation? Linux has the LDP, OpenBSD has the man pages. Although the LDP are much nicer in hand holding, OpenBSD's man pages are so convenient for us who are not 'live' on the NET. INSTALL.386 has a section "Using online OpenBSD documentation," scan through it if you are new to Unix, it has some helpful pointers on how to better make use of man pages.

There is a real nice introductory, short, tutorial for those totally new to Unix at http://www.freebsd.org/tutorials/new-users

You should at least read through the tutorial for a guide to what you will do here (and reference.)

Configuring Removable Storage Devices

(e.g. CD Drives, Zip Drives, etc.)

Configuration in /etc/fstab
[Ref: mount(8) mount file systems;
mount_msdos(8) mount an MS-DOS file system,
mount_cd9660(8) mount an ISO-9660 filesystem]
[Ref: fstab(5) - static information about the filesystems]

To simplify my installation process (low bandwidth people) I need to configure access to my CD-ROM drive.

Use **dmesg | less** to look for the device name detected as the cdrom drive. CD drives are often detected as device cd# (like cd0 or cd1). 'dmesg' is a command-line program in OpenBSD that lists boot-time information (such as what OpenBSD detects as devices on your system during startup.) less is another command-line program, this program lets you browse through a file by using space (next page) up-arrow, down-arrow, and "q" for quit.

Edit the /etc/fstab file to tell OpenBSD that I have the cdrom drive setup and this helps simplify my mounting command. If you do not yet know how to use the vi editor or other editors available during the default install, I suggest that practicing with vi will improve your enjoyment of Unix (OpenBSD.)

File: /etc/fstab

```
# "#" starts comments
#
# device        mount-point fs_type          mnt options check priority
#
/dev/wd0a       /           ffs              auto,rw      1       1

# The following is an example of what you may need to add
#
/dev/cd0a       /mnt/cdrom  cd9660           noauto,ro    0       0
/dev/fd0a       /mnt/floppy msdos            noauto,rw    0       0
```

Although the CDROM device is detected by the kernel during each boot, and during installation the device is not automatically configured for use. I have also included above how to configure floppy disk access (assuming /dev/fd0 is the controller and /dev/fd0a is the a drive.) I specify msdos file format since I mostly work with msdos floppy drives (Winx) and have no need to transfer any other format floppies.

I now create the nodes (points) for where the file systems can be mounted by issuing the following commands:

```
# mkdir /mnt
# mkdir /mnt/cdrom
# mkdir /mnt/floppy
```

I can now access the CD-ROM drive by entering the below command at the system prompt.

```
# mount /mnt/cdrom
```

Note that you will receive a read error if a CD is not in the drive. This is because mount doesn't actually configure the device, but attempts to find the file-system on the device, and mount the filesystem. To correctly mount on any device, we require a valid file-system on that device.

Similarly you can access the floppy drive. I've selected the above mounting location (/mnt) because I have a background in RedHat Linux distributions and am used to this convention where some BSD documentation's I have read prefer the /cdrom layout.

Example : *iwill motherboard with ATAPI IDE CD, SCSI CDR and SCSI Zip drive*

dmesg outputs a lot of junk with the below information included that seems valid for removable drives.

```
cd0 at scsibus0 targ 1 lun 0: <E-IDE, CD-ROM 45X, 32> SCSI0 5/cdrom
removable
cd1 at scsibus1 targ4 lun 0: <PLEXTOR, CD-R PX-R412C, 1.04> SCSI2
5/cdrom removable
sd0: 96MB, 96 cyl, 64 head, 32 sec, 512 bytes/sec, 196608 sec total
fd0 at fdc0 drive 0: 1.44MB 80 cyl, 2 head, 18 sec
```

The command "**dmesg | less**" lets us navigate up and down the list (using arrow keys) and I can quit "less" by typing in "q" to quit.

I test the ability to access the devices by first creating the 'node' or directory to mount the devices and using the mount command to check where the device 'special' is located.

/mnt/cdrom - for the E-IDE CDROM (filesystem: cd9660)
/mnt/cdr - for the CDR (filesystem: cd9660)
/mnt/floppy - for the floppy drive (filesystem: msdos)
/mnt/zip - for the Iomega SCSI ZIP drive (filesystem: msdos)

We're choosing cd9660 as the filesystem for CD drives as this is OpenBSD's name for ISO-9660 CDROM filesystem. We use msdos in this example since all other machines sharing zip drives and floppies are MSWin platforms which share MSDOS FAT filesystems (fat16, fat32) Examples for using mount (as I do below) are also listed with the mount man pages.

```
# mkdir /mnt
# mkdir /mnt/cdrom
# mkdir /mnt/cdr
# mkdir /mnt/floppy
# mkdir /mnt/zip
```

We start looking at the devices from /dev/???a ... b ... c ... until we find it. Where ??? is the device we are reviewing. Put a CD into the CD Drive and/or floppies into floppy etc. We need to make sure we have a valid media (disk) inside each drive for the mounting process to find the disk we want to mount. We use the "-v" option so we can get some debugging information from the mount command.

```
# mount -v -t cd9660 /dev/cd0a /mnt/cdrom
/dev/cd0a on /mnt/cdrom type cd9660 (local, read-only)
# mount -v -t cd9660 /dev/cd1a /mnt/cdr
/dev/cd1a on /mnt/cdr type cd9660 (local, read-only)
# mount -v -t msdos /dev/fd0a /mnt/floppy
/dev/fd0a on /mnt/floppy type msdos (rw, local, uid=0, gid=0, mask=0755)
```

The above three devices seemed to work easily with the first 'device' but the mounted zip took a little while longer to find as shown with the testing below.

```
# mount -v -t msdos /dev/sd0a /mnt/zip
mount_msdos: /dev/sd0a on /mnt/zip: Device not configured
# mount -v -t msdos /dev/sd0b /mnt/zip
mount_msdos: /dev/sd0a on /mnt/zip: Device not configured
# mount -v -t msdos /dev/sd0c /mnt/zip
/dev/sd0c on /mnt/zip type msdos (rw, local, uid=0, gid=0, mask=0755)
```

We now know where the devices can be located and can confidently specify our devices into the /etc/fstab file system table.

Edit: /etc/fstab

```
/dev/cd0a /mnt/cdrom cd9660 ro,noauto 0 0
/dev/cd1a /mnt/cdr   cd9660 rw,noauto 0 0
/dev/sd0c /mnt/zip    msdos rw,noauto 0 0
/dev/fd0a /mnt/floppy msdos rw,noauto 0 0
```

Now, all we need to do to access one of the devices above is to use "mount /mnt/????" (where ???? is the directory created above) and mount will look up the device setting/file system from the /etc/fstab file.

As an extra note for those sharing files with the FAT file system you may be interested in reading the mount_msdos man pages for more information about support for long filenames.

Adding additional 'packages'

Utility: pkg_add, pkg_info, pkg_delete
Config location: /usr/src

The **pkg_add** utility is used to install binary packages already compiled and configured for the standard OpenBSD distribution settings. pkg_add is also used with the 'ports' collection to automatically download/compile and configure source code files from CD or from the internet.

For those new to packages (like me) I change to the directory containing the packages before using pkg_add (this is not necessary and is explained later in setting environment variables for bash, my preferred shell.)

The general format for using pkg_add is:

```
# pkg_add –v /[path-to-package]/filename
# pkg_add –v ftp.site.com/[path-to-package]/filename
```

The –v option is Verbose, which is real helpful in providing visual feedback of files it is processing. After you figure out how things work, you can leave the "-v" off.

To provide an example, let's install the bash shell. We will progress here after you have inserted the OpenBSD cd into your CD drive mounted on /mnt/cdrom and you have mounted the drive.

```
# cd /mnt/cdrom/2.7/packages/i386
# ls -l bash*
bash-1.14.7-static.tgz bash-2.04-static.tgz
# pkg_add bash-2.04-static.tgz
Requested space: 4606268 bytes, free space: 7432482816 bytes in
/var/tmp/instmp.eepTB28148
Running install with PRE-INSTALL for `bash-2.04-static'
extract: Package name is bash-2.04-static
extract: CWD to /usr/local
extract: /usr/local/bin/bash
extract: /usr/local/bin/bashbug
extract: /usr/local/man/man1/bash.1
extract: /usr/local/man/man1/bashbug.1
extract: /usr/local/info/bash.info
extract: execute 'install-info /usr/local/info/bash.info /usr/local/info/dir'
extract: /usr/local/share/doc/bash/article.ps
extract: /usr/local/share/doc/bash/article.txt
extract: /usr/local/share/doc/bash/bash.html
extract: /usr/local/share/doc/bash/bash.ps
extract: /usr/local/share/doc/bash/bashbug.ps
extract: /usr/local/share/doc/bash/bashref.html
extract: /usr/local/share/doc/bash/bashref.ps
extract: /usr/local/share/doc/bash/builtins.ps
extract: /usr/local/share/doc/bash/readline.ps extract: CWD to .
Running install with POST-INSTALL for `bash-2.04-static'
Attempting to record package into `/var/db/pkg/bash-2.04-static'
Package `bash-2.04-static' registered in `/var/db/pkg/bash-2.04-static'
```

If a package (like bash) gives you further instructions for complete the installation, make sure you follow the instructions.

For those without the Official OpenBSD CDs.

Performing a pkg_add from an ftp connection is no more difficult than the above, as shown in the below example for installing the same package.

```
# pkg_add ftp://192.168.101.77/OpenBSD/2.7/packages/i386/bash-2.04-static.tgz
>>> ftp -o - ftp://192.168.101.77/OpenBSD/2.7/packages/i386/bash-2.04-static.tgz
'EPSV': command not understood
#
```

pkg_add retrieves from my internal ftp site (192.168.101.77) the requested package and then extracts the files as per the same operation above. You can replace 192.168.101.77 with any valid ftp site which holds the package you wish to install.

If you do not know what the package name is, or the specific directory the file is located, you can still perform an ftp pkg_add. Try using the get filename **"| command "** sequence as shown in the below example. (note: I am connecting here to an internal site with the OpenBSD files, connect to some other site)

```
# ftp 192.168.101.77
Connected to 192.168.101.77.
Name (192.168.101.77:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
ftp> cd pub/OpenBSD/2.7/packages/i386
250 CWD command successful.
ftp> ls bash*
227 Entering Passive Mode (192,168,101,77,4,164).
125 Data connection already open; Transfer starting.
-r-xr-xr-x   1 owner    group         261366 May 10  0:24 bash-1.14.7.tgz
-r-xr-xr-x   1 owner    group         376068 May 10  0:26 bash-1.14.7-static.tgz
-r-xr-xr-x   1 owner    group        1000070 Jun 15  3:32 bash-2.04.tgz
-r-xr-xr-x   1 owner    group        1151567 Jun 15  3:32 bash-2.04-static.tgz
226 Transfer complete.
ftp> bi
200 Type set to I.
ftp> get bash-2.04-static.tgz "| pkg_add -zxf -"
local: | pkg_add -zxf - remote: bash-2.04-static.tgz
227 Entering Passive Mode (192,168,101,77,4,166).
125 Data connection already open; Transfer starting.
226 Transfer complete.
1151567 bytes received in 1.09 seconds (1.01 MB/s)
ftp> quit
#
```

Adding a new user with root access privileges

Utility: adduser
Config info: user-name, account-type

The first thing that OpenBSD warns of when you login is, do not login as root but use su. This is saying that you should create a user who can use su (the Switch User program) to change to the "root" user when you want to perform administration tasks. The following instructions guide you through the creation of a new user with SuperUser access privileges.

OpenBSD supplies the **adduser** script to simplify adding new users. All you have to know to create a new user is the name of the person, and what you want the login account name to be.

The adduser script is started at the command prompt, and when first started, queries you to set or change the default settings. Once the standard configuration has been set, it will continue by prompting for adding new users.

```
# adduser
```

adduser support two flags -silent or -verbose. You don't really need to know these at the beginning, but you can check the details in the man pages. Read through the example below and then start adduser to create your new account with root access privileges.

```
# adduser
Enter username [a-z0-9_-]: bricker
Enter full name [ ]: Sven De La Palmer
Enter shell bash csh ksh nologin sh [bash]: <hit ENTER>
```

The shell is your command line interpreter. It reads in the commands you type and tries to decipher them. There are several different shells to choose from. If bash does not show on the screen, then review adding packages in the previous section. You can change your settings at a later time so do not worry if some settings are not as you want them right now. The documentation that comes with OpenBSD says that 'most people' use bash, strange how they don't make it the default though.

```
Enter home directory (full path) [/home/bricker]: <hit ENTER>
Uid [1002]: <hit ENTER>
```

The uid is the User ID number that the system uses to keep track of people. These should be unique on the system. Use the default values offered by the program unless you have good knowledge of previously granted ID numbers.

```
Enter login class: default []: <hit ENTER>
```

The login class allows you to set up resource limits for groups of users.

```
Login group bricker [bricker]: <hit ENTER>
Login group is "bricker". Invite bricker into other groups: guest no
[no]:  wheel
```

Important: Your administrator account should be a member of the group **wheel**. Regular users of your host should not be members of the wheel group. If this is your 1st account for the machine (and presumably your account) then I suggest you add the account to the group "**wheel**."

Login groups are used to divide security privileges by account groups. The group '**wheel**' is generally used for administrators with special privileges including the ability to su (switch user) to the SuperUser. Accounts who are not members of the group 'wheel' cannot gain root access remotely. Invite user accounts you wish to grant special security rights into the group '**wheel**,' or create a separate security group for people who need to work together.

```
Enter password []:
Enter password again []:
```

You will be asked for the user's password twice and it will not be displayed. Afterwards, it will display all of the user's information and ask if it is correct.

```
Name:     bricker
Password: ****
Fullname: Sven De La Palmer
Uid:      1000
Gid:      1000 (bricker)
Class:
Groups:   bricker wheel
HOME:     /home/bricker
Shell:    /bin/sh
OK? (y/n) [y]: <hit ENTER>
```

If you make a mistake, you can start over, or its possible to correct most of this information using the 'chpass' command (discussed below).

Changing User Information

Utility: chpass (cf chpass, vipw)

Once you've configured the base system for working, we can look at basic configuration of users. Note, for those with some previous Unix experience, Do not just edit /etc/passwd or /etc/Master.passwd

Use the chpass utility when adding or changing user information. If you try to modify the user shell selection manually (by changing /etc/passwd) it wont work, trust me I've made this mistake for weeks before I found out my erroneous ways.

Entered at the command line without a parameter (ie. typed by itself,) chpass will edit your personal information. As root, you can use it to modify any user account on the system. You can find more details on chpass in the man pages, but let's go through an example review of the account we created above.

```
# chpass bricker
```

This will brings up information about the user 'bricker' in the 'vi' editor. The password line is encrypted, so don't change it. If you want to disable the user, one method would be to add a # at the beginning of the password string, so you can easily remove it later when you want to reactivate the user. There are methods of disabling user that may be better though.

```
Login: bricker
Password:
Uid [#]: 1000
Gid [# or name]: 1000
Change [month day year]:
Expire [month day year]:
Class:
Home directory: /home/bricker
Shell: /bin/sh
Full Name: Sven De La Palmer
Office Location:
Office Phone:
Home Phone:
Other information:
~
~
~
~
~
~
~
~
/path/temp-file: unmodified: line 1
```

Remember your vi commands ? :q (colon+q) quit, :w (colon+w) write, :q! (colon+q+exlamation-mark) quit without saving. If you're still having problems, remember the tutorial http://www.freebsd.org/tutorials/new-users

Configuring bash

Files: .bash_profile, and .bashrc

Since I like using the Bash shell largely due to my ignorance about the other shells, here is an example of the files for initialisation. The two user files which contain the shell settings are ~/.bash_profile, and ~/.bashrc.

Note that these are templates and there are some things that MUST be changed. I've put **[path-to-….]** as designators of specific paths that have to be set by the user/admin.

File: ~/.bash_profile

```
# .bash_profile
#
# Things loaded once per session (by the login manager).
#
# Source of global definitions
if [ -f /etc/bashrc ]; then
   . /etc/bashrc
fi

PATH=/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/local/sbin:/usr/X11R6/bin

# Define variables useful for OpenBSD Installations
#
PKG_PATH=/[path-to-packages]/packages/i386
export PKG_PATH PATH
# Change the prompt to give current directory (\W) and
# $ if regular user -or- # if root (\$).
PS1='\W \$'
export PS1
```

File: ~/.bashrc

```
# .bashrc

# Put in here variables and stuff to be launched by subinvocations
# of bash (like /usr/local/bin/bash)
```

The tilde ~ is used here to refer to the home directory of the current user. Therefore if you are logged in as 'bricker' then typing in cd ~ should put you in the directory /home/bricker. Likewise if you edit the file ~/.bash_profile the file is actually created as /home/bricker/.bash_profile. If you were to su (switch user) to root and then type cd ~ you should be moved to /root the home directory for root.

Afterboot Settings

The afterboot man pages list a sequence of issues to review after the OpenBSD system has been configured and is up and running. For the 'expert' practitioner many of the items seem trivial, for us newbies it is a good time to review basic skills that will be re-used often and will probably minimize problems that would otherwise occur just from not checking 'basic' items.

afterboot is a serious document if you want to ensure the stability of your system. I recommend you read the document anyway and use these pages as supportive material where possible. These notes are supportive of afterboot material.

Date - Setting the Date & Time

You can check and configure the system date using the date command. Without parameters, date command will display the current system date. You can set the date by using the following template

```
date YYYYMMDDHHMM
```

Where YYYY is the four digit year, followed by MM a two digit month of the year, DD a two digit date of the month, HH a two digit (24 hour) representation of the hour, and MM for the minute in the hour.

Using the above specification, we can set (as per man afterboot example)

```
# date 199901271504
```

Set the current date to January 27th, 1999 3:04pm.

For those new to the convention used above (YYYYMMDDHHMM) it is the ANSI specified date format for SQL. I also prefer the above date formatting as it is less confusing when sharing things with the Americans 8-)

TimeZone – Setting the Time Zone

The time zone information is recorded as data files under the /usr/share/zoneinfo directory. So if I want to set the timezone to Paris, France then I can look it up using "find / -name "Paris" -print" and I can specify the zone file by typing in:

```
/root # cd /usr/share/zoneinfo
zoneinfo # find . -name "Paris" -print
./Europe/Paris
zoneinfo # ln -fs /usr/share/zoneinfo/Europe/Paris /etc/localtime
```

Of course for us people in Tonga with UTC+13 we use ln -fs /usr/share/zoneinfo/Pacific/Tongatapu /etc/localtime (I thought you might just want to know that ?)

Network – Setting the basic network services

Basic services for connecting on the network are generally covered by these three items.

- host configuration,
- network interface configuration, and
- network routing.

Host configuration details

Files: /etc/hosts, /etc/myname

For many network services to function they need to determine the name of the current host. Host Details are checked by using the **hostname** command. hostname will display what your current host name is. If you need to change the hostname more details are available in the hostname(1) man page. If you change the hostname, then you need to also make the change to /etc/myname and possibly /etc/hosts.

/etc/hosts is a text file listing IP addresses and their related hostnames. Your hostname should be in this file associated with the IP address which you assigned your host during installation.

/etc/myname is a text file with just one line containing the hostname of your machine.

Network interface configuration

Network interfaces are necessary if you wish to communicate to other computers (at least if you want to communicate using the standard tools.) In most cases the network interface device will be an Ethernet card. To list the network devices recognized by your system we use the ifconfig -a command.

```
# ifconfig -a
```

The **ifconfig -a** command will list the network interfaces currently active on the system. This will let you review what the system knows of itself during this instance. You can set the default configurations by editing the /etc/hostname.* file that corresponds to the network interface.

If the **ifconfig -a** command lists an interface **le0** than the corresponding hostname file will be **/etc/hostname.le0**

Example: ifconfig -a displays the following ethernet device on my compaq with a HP network card.

```
le1: flags=8863
<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST>
mtu 1500
inet 192.168.101.130 netmask 0xffffff00 broadcast 192.168.101.255
inet6 fe80::260:b0ff:fea4:18d3%le1 prefixlen 64 scopeid 0x1
```

The related hostname file is /etc/hostname.le1 which contains the lines

```
inet 192.168.101.130 255.255.255.0 NONE
inet alias 207.124.66.156
```

You can see that the inet line in hostname.le1 corresponds to the inet line displayed by ifconfig -a. ifconfig allows you to manually configure the network card, or at least check different configurations before you insert the details into the hostname.interface file. Details for configuring the network card are read from the /etc/hostname.interface file during the boot sequence.

An example output for the loopback device will look like:

```
lo0: flags=8009<UP,LOOPBACK,MULTICAST> mtu 32972

inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
inet6 ::1 prefixlen 128
inet 127.0.0.1 netmask 0xff000000
```

If you have other network interfaces (example a ppp connection) then these will also be listed. Check the afterboot and ifconfig pages for more details.

The inet line specifies IPv4 information whereas the inet6 line specifies IPv6 information. Since OpenBSD is an early adopter of IPv6 you will see this additional information for many network devices.

Routing Configuration

We can check the network routing using **netstat -r -n**

```
# netstat -r -n
Routing tables
Internet:
Destination     Gateway         Flags  Refs Use   Mtu   Interface
127/8           127.0.0.1       UGRS   0      0  32972    lo0
127.0.0.1       127.0.0.1       UH     4     42  32972    lo0
192.168.101/24  link#1          UC     0      0   1500    le1
192.168.101.130 127.0.0.1       UGHS   0    122  32972    lo0
192.168.101.255 link#1          UHL    3     49   1500    le1
207.124.66/24   link#1          UC     0      0   1500    le1
207.124.66.156  127.0.0.1       UGHS   0      5  32972    lo0
224/4           127.0.0.1       URS    0      0  32972    lo0
```

If you are new to Unix, then just check to make sure the IP address you specified for your host is listed and take a note that the IP range (class) is gatewayed through the interface.

In the above example all 192.168.101/24 destinations (except for my host ip address 192.168.101.130 nor the broadcast address 192.168.101.255) are sent through link#1 which is my network interface le1 *[note: I need to verify more of this detail]*

As I have an alias to the 207.124.66.156 the 207.124.66/24 destinations are also sent through link#1 (except for the host alias 207.124.66.156 *[note: I need to verify more of this detail]*

The default gateway address is stored in the /etc/mygate file. If you need to edit this file, a painless way to reconfigure the network afterwards is route flush followed by a sh -x /etc/netstart command. Or, you may prefer to manually configure using a series of route add and route delete commands (see route(8))

```
# route flush
# sh -x /etc/netstart
```

Daily, Weekly, Monthly Scripts

Actions that are scheduled to occur in a repetitive pattern such as once each day, each week, each month can be placed into the /etc/daily.local /etc/weekly.local /etc/monthly.local scripts.

The OpenBSD installation supplies a set of standard /etc/daily, /etc/weekly, and /etc/monthly scripts. The scripts will check for daily.local, weekly.local, and monthly.local so you should specify your scripts as part of one of the above *.local files.

Finding and locating files. One of the more frequently asked questions is how to find a file. The /etc/weekly script updates (on a weekly basis) the locate.db file to index files on your system. To manually execute the db update, see the notes below.

To manually execute any of the above scripts, they are sh shell scripts, then use one of the examples below

```
# sh /etc/daily
# sh /etc/weekly
# sh /etc/monthly
```

Miscellaneous

Making it easier to find files

[ref: locate(8) - find filenames quickly]
[ref: locate.updatedb(1) - update locate database]
[ref: find(1) - walk a file hierarchy]

Unix has a nice file indexing utility accessible through '**locate**.' The locate program interrogates a database created by locate.updatedb, in this manner you do not have to traverse the hard-disk each time you want to find a file. Update the file/location database by using the locate.updatedb program and then interrogate (search in) the database by using locate. Start locate.updatedb.

```
# /usr/libexec/locate.updatedb
# locate filename
```

Now you can use 'locate filename' to find exactly where that file is. As locate.updatedb updates information in the locate database dependent on the user starting the program there is a potential risk (since root has access to all files) of listing files in the database that you do not want other users to be aware of.

To be safe, you could just manually start the /etc/weekly script which is configured to execute locate.updated as user "nobody" without the access privileges available to root:

```
# sh /etc/weekly
```

Using the above weekly script is simpler than trying to figure out how su, nice interact to minimize security holes through the locate db.

Otherwise you can still use the Unix 'find / -name "filename"' command

```
# find / -name "filename"
```

Booting in Single User Mode

[ref: FAQ. 14.0 Disk Setup]

Booting the system in Single User Mode is an important option when you need to perform tasks on the machine that is sensitive to other user activities on the system. Of course, you could be just like me and have forgotten root's password or have zapped the shell you used for root and other accounts and need to dive back into root to fix the system.

When your system starts up, it momentarily offers the boot> prompt where we can force single user mode.

```
boot> boot -s
```

Assuming you performed the above steps correctly and nothing has gone wrong you should end up at a prompt asking you for a shell path or press return. Press return to use sh.

The single user mode starts with the "/" partition. This partition has been mounted as read only (precautionary procedure). It is advisable at this point to perform a file system check on the "/" partition.

```
shell # fsck /
```

After the fsck we want to remount root in r/w mode as opposed to read only. Issue the following command:

```
shell # mount -u -w /
```

The "-u" flag allows us to change the status of an already mounted file system (because "/" was previously mounted by the startup. The "-w" flag tells mount to make "/" read-write.

Once you have mounted "/" as read/write you can also mount the rest of your file system or just do what it is you want to do in single user mode and restart the system.

Moving Directories Safely

Problem: How can I safely move all files/directories under /opt to /home/opt ?

Sooner or later you'll come across the problem of running out of disk-space on your partition scheme. The following is a set of methods for 'safely' moving files from one folder to another.

For this example we will pretend that our /opt directory has just filled our / partition and we need to move files from /opt to a less congested partition (or a new drive) so we can continue developing ('acking'.) We find that /usr is getting tight on space and /home has heaps of space (cause we have no users yet,) so we will move the files to /home/opt for the time-being.

```
/opt : FULL sub-directories kde, Office51, etc.
/home/opt : FREESPACE, there's plenty of freespace here, so we'll
relocate files

option 1: cd /opt; find . -xdev -depth -print | cpio -pdmu /home/opt
option 2: cd /opt; tar cXf - . | (cd /home/opt; tar xpf - )
option 3: cd /home/opt; dump -0uaf - /opt | restore -rf -
```

Option 1: [ref: OpenBSD FAQ and e-mail by Håkan Olsson]

If the 'find' is run on the locally mounted filesystem, this is a rather efficient method to copy the data. Also, if you move lots of data and there is the chance it may change during copy/move time (say user or project data on an NFS-exported partition), you can rerun once without the 'u' flag to cpio, in which case only updated files are copied, if any. Not foolproof certainly, but often good enough if you have sane time in your network (ntp, et al).

-xdev (x: do not search directories on other file systems/devices, d: depth-first traversal; e:

Option 2: [ref: e-mail by Christopher Linn]

This would be if you have any other partitions mounted inside of /usr, you don't want tar to cross filesystem boundary

Option 3: [ref: e-mail by Dan Harnett]

It has been my experience that it is safer and more reliable.

[ref: OpenBSD FAQ. 14.0 Disk Setup -> 14.3 Adding Extra Disks in OpenBSD]
Note: the use of the above names in no way implies these people want to be associated with this information release

General Tools I install

The OpenBSD base install has a number of standard features (web server etc.) Below is just a list of tools that I used on a consistent basis to be installing with each generic install I put together.

bash-2.04	GNU Bourne Again Shell
m4-1.4	GNU m4
Autoconf	automatically configure source code
automake	GNU Makefile generator
bison	another one of those tools that seems to be needed when compiling various programs
gmake	GNU version of make
mawk	new/posix awk
samba	SMB/CIF file/print resource sharer very useful with MS Windows environments
vnc	display X & Win32 desktops on remote X/Win32/java displays

Available from ftp sites (& distfiles)

openssh	SSH1 and SSH2 binaries, clients installed by default but servers require RSA libraries available on ftp sites.
openssl	ssl27 (ssl26) contains RSA code
Pgp-intl	Data Encryption package

Author and Copyright

I reserve the right to be totally incorrect even at the best advice of betters. In other words, I'm probably wrong in enough places for you to call me an idiot, but don't 'cause you'll hurt my sensibilities, just tell me where I went wrong and I'll try again.

You are permitted and encouraged to use this guide for fun or for profit as you see fit. If you republish this work in what-ever form, it would be nice (though not enforceable) to be credited.