FreeBSD Quickstart Guide for Linux® Users

John Ferrell

$FreeBSD: release/9.2.0/en_US.ISO8859-1/articles/linux-users/article.xml 41645 2013-05-17 18:49:52Z gabor $

$FreeBSD: release/9.2.0/en_US.ISO8859-1/articles/linux-users/article.xml 41645 2013-05-17 18:49:52Z gabor $

This document is intended to quickly familiarize intermediate to advanced Linux® users with the basics of FreeBSD.

Table of Contents

1 Introduction

This document will highlight the differences between FreeBSD and Linux so that intermediate to advanced Linux users can quickly familiarize themselves with the basics of FreeBSD. This is just a technical quickstart, it does not attempt to design "philosophical" differences between the two operating systems.

This document assumes that you have already installed FreeBSD. If you have not installed FreeBSD or need help with the installation process please refer to the Installing FreeBSD (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/install.html) chapter of the FreeBSD Handbook.

2 Shells: No Bash?

Those coming from Linux are often surprised to find that **Bash** is not the default shell in FreeBSD. In fact, **Bash** is not even in the default installation. Instead, FreeBSD uses tcsh(1) as the default shell. Although, **Bash** and your other favorite shells are available in FreeBSD's Packages and Ports Collection (article.html#SOFTWARE).

If you do install other shells you can use chsh(1) to set a user's default shell. It is, however, recommended that the `root`'s default shell remain unchanged. The reason for this is that shells not included in the base distribution are normally installed in `/usr/local/bin` or `/usr/bin`. In the event of a problem the file systems where `/usr/local/bin` and `/usr/bin` are located may not be mounted. In this case `root` would not have access to its default shell, preventing `root` from logging in. For this reason a second `root` account, the `toor` account, was created for use with non-default shells. See the security FAQ for information regarding the toor account (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/faq/security.html#TOOR-ACCOUNT).

3 Packages and Ports: Adding software in FreeBSD

In addition to the traditional UNIX® method of installing software (download source, extract, edit source code, and compile), FreeBSD offers two other methods for installing applications: packages and ports. A complete list of of all available ports and packages can be found here (http://www.freebsd.org/ports/master-index.html).

3.1 Packages

Packages are pre-compiled applications, the FreeBSD equivalents of `.deb` files on Debian/Ubuntu based systems and `.rpm` files on Red Hat/Fedora based systems. Packages are installed using pkg_add(1). For example, the following command installs **Apache 2.2**:

```
# pkg_add /tmp/apache-2.2.6_2.tbz
```

Using the `-r` switch will tell pkg_add(1) to automatically fetch a package and install it, as well as any dependencies:

```
# pkg_add -r apache22
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6.2-release/Latest/apache22.tbz...
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6.2-release/All/expat-2.0.0_1.tbz.
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6.2-release/All/perl-5.8.8_1.tbz..
[snip]

To run apache www server from startup, add apache22_enable="YES"
in your /etc/rc.conf. Extra options can be found in startup script.
```

> **Note:** If you are running a release version of FreeBSD (6.2, 6.3, 7.0, etc., generally installed from CD-ROM) `pkg_add -r` will download packages built for that specific release. These packages *may not* be the most up-to-date version of the application. You can use the `PACKAGESITE` variable to override this default behavior. For example, set `PACKAGESITE` to ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-6-stable/Latest/ to download the most recent packages built for the 6.X series.

For more information on packages please refer to section 4.4 of the FreeBSD Handbook: Using the Packages System (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/packages-using.html).

3.2 Ports

FreeBSD's second method for installing applications is the Ports Collection. The Ports Collection is a framework of `Makefiles` and patches specifically customized for installing various software applications from source on FreeBSD. When installing a port the system will fetch the source code, apply any required patches, compile the code, and install the application (and do the same for any dependencies).

The Ports Collection, sometimes referred to as the ports tree, can be found in `/usr/ports`. That is assuming the Ports Collection was installed during the FreeBSD installation process. If the Ports Collection has not been installed it can be added from the installation discs using sysinstall(8), or pulled from the FreeBSD servers using csup(1) or portsnap(8). Detailed instructions for installing the Ports Collection can be found in section 4.5.1 (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/ports-using.html) of the handbook.

Installing a port is as simple (generally) as changing in to the port's directory and starting the build process. The following example installs **Apache 2.2** from the Ports Collection:

```
# cd /usr/ports/www/apache22
# make install clean
```

A major benefit of using ports to install software is the ability to customize the installation options. For example, when installing **Apache 2.2** from ports you can enable **mod_ldap** by setting the `WITH_LDAP` make(1) variable:

```
# cd /usr/ports/www/apache22
# make WITH_LDAP="YES" install clean
```

Please see section 4.5 of the FreeBSD Handbook, Using the Ports Collection (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/ports-using.html), for more information about the Ports Collection.

3.3 Ports or packages, which one should I use?

Packages are just pre-compiled ports, so it is really a matter of installing from source (ports) versus installing from binary packages. Each method has its own benefits:

Packages (binary)

- Faster installation (compiling large applications can take quite a while).
- You do not need to understand how to compile software.
- No need to install compilers on your system.

Ports (source)

- Ability to customize installation options. (Packages are normally built with standard options. With ports you can customize various options, such as building additional modules or changing the default path.)
- You can apply your own patches if you are so inclined.

If you do not have any special requirements, packages will probably suit your situation just fine. If you may ever need to customize, ports are the way to go. (And remember, if you need to customize but prefer packages, you can build a custom package from ports using `make package` and then copy the package to other servers.)

4 System Startup: Where are the run-levels?

Linux uses the SysV init system, whereas FreeBSD uses the traditional BSD-style init(8). Under the BSD-style init(8) there are no run-levels and no `/etc/inittab`, instead startup is controlled by the rc(8) utility. The `/etc/rc` script reads `/etc/defaults/rc.conf` and `/etc/rc.conf` to determine which services are to be started. The specified services are then started by running the corresponding service initialization scripts located in `/etc/rc.d/` and `/usr/local/etc/rc.d/`. These scripts are similar to the scripts located in `/etc/init.d/` on Linux systems.

> *Why are there two locations for service initialization scripts?* The scripts found in `/etc/rc.d/` are for applications that are part of the "base" system. (cron(8), sshd(8), syslog(3), and others.) The scripts in `/usr/local/etc/rc.d/` are for user-installed applications such as **Apache**, **Squid**, etc.
>
> *What is the difference between the "base" system and user-installed applications?* FreeBSD is developed as a complete operating system. In other words, the kernel, system libraries, and userland utilities (such as ls(1), cat(1), cp(1), etc.) are developed and released together as one. This is what is referred to as the "base" system. The user-installed applications are applications that are not part of the "base" system, such as **Apache**, **X11**, **Mozilla Firefox**, etc. These user-installed applications are generally installed using FreeBSD's Packages and Ports Collection (article.html#SOFTWARE). In order to keep them separate from the "base" system, user-installed applications are normally installed under `/usr/local/`. Therefore the user-installed binaries reside in `/usr/local/bin/`, configuration files are in `/usr/local/etc/`, and so on.

Services are enabled by specifying *`ServiceName`*`_enable="YES"` in `/etc/rc.conf` (rc.conf(5)). Take a look at `/etc/defaults/rc.conf` for the system defaults, these default settings are overridden by settings in `/etc/rc.conf`. Also, when installing additional applications be sure to review the documentation to determine how to enable any associated services.

The following snippet from `/etc/rc.conf` enables sshd(8) and **Apache 2.2**. It also specifies that **Apache** should be started with SSL.

```
# enable SSHD
sshd_enable="YES"
# enable Apache with SSL
apache22_enable="YES"
apache22_flags="-DSSL"
```

Once a service has been enabled in `/etc/rc.conf`, the service can be started from the command line (without rebooting the system):

```
# /etc/rc.d/sshd start
```

If a service has not been enabled it can be started from the command line using `forcestart`:

```
# /etc/rc.d/sshd forcestart
```

5 Network configuration

5.1 Network Interfaces

Instead of a generic *ethX* identifier that Linux uses to identify a network interface, FreeBSD uses the driver name followed by a number as the identifier. The following output from ifconfig(8) shows two Intel® Pro 1000 network interfaces (`em0` and `em1`):

```
% ifconfig
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=b<RXCSUM,TXCSUM,VLAN_MTU>
        inet 10.10.10.100 netmask 0xffffff00 broadcast 10.10.10.255
        ether 00:50:56:a7:70:b2
        media: Ethernet autoselect (1000baseTX <full-duplex>)
        status: active
em1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=b<RXCSUM,TXCSUM,VLAN_MTU>
        inet 192.168.10.222 netmask 0xffffff00 broadcast 192.168.10.255
        ether 00:50:56:a7:03:2b
        media: Ethernet autoselect (1000baseTX <full-duplex>)
        status: active
```

5.2 IP Configuration

An IP address can be assigned to an interface using ifconfig(8). However, to remain persistent across reboots the IP configuration must be included in `/etc/rc.conf`. The following example specifies the hostname, IP address, and default gateway:

```
hostname="server1.example.com"
ifconfig_em0="inet 10.10.10.100  netmask 255.255.255.0"
defaultrouter="10.10.10.1"
```

Use the following to configure an interface for DHCP:

```
hostname="server1.example.com"
ifconfig_em0="DHCP"
```

6 Firewall

Like **IPTABLES** in Linux, FreeBSD also offers a kernel level firewall; actually FreeBSD offers three firewalls:

- IPFIREWALL (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html)
- IPFILTER (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipf.html)
- PF (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/firewalls-pf.html)

IPFIREWALL or **IPFW** (the command to manage an **IPFW** ruleset is ipfw(8)) is the firewall developed and maintained by the FreeBSD developers. **IPFW** can be paired with dummynet(4) to provide traffic shaping capabilities and simulate different types of network connections.

Sample **IPFW** rule to allow **SSH** in:

```
ipfw add allow tcp from any to me 22 in via $ext_if
```

IPFILTER is the firewall application developed by Darren Reed. It is not specific to FreeBSD, and has been ported to several operating systems including NetBSD, OpenBSD, SunOS, HP/UX, and Solaris.

Sample **IPFILTER** command to allow **SSH** in:

```
pass in on $ext_if proto tcp from any to any port = 22
```

The last firewall application, **PF**, is developed by the OpenBSD project. **PF** was created as a replacement for **IPFILTER**. As such, the **PF** syntax is very similar to that of **IPFILTER**. **PF** can be paired with altq(4) to provide QoS features.

Sample **PF** command to allow **SSH** in:

```
pass in on $ext_if inet proto tcp from any to ($ext_if) port 22
```

7 Updating FreeBSD

There are three methods for updating a FreeBSD system: from source, binary updates, and the installation discs.

Updating from source is the most involved update method, but offers the greatest amount of flexibility. The process involves synchronizing a local copy of the FreeBSD source code with the FreeBSD **Subversion** servers. Once the local source code is up to date you can build new versions of the kernel and userland. For more information on source updates see the chapter on updating (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/updating-upgrading.html) in the FreeBSD Handbook.

Binary updates are similar to using `yum` or `apt-get` to update a Linux system. The command freebsd-update(8) will fetch new updates and install them. The updates can be scheduled using cron(8).

> **Note:** If you do use cron(8) to schedule the updates, please be sure to use `freebsd-update cron` in your crontab(1) to reduce the possibility of a large number of machines all pulling updates at the same time.

```
0 3 * * * root /usr/sbin/freebsd-update cron
```

The last update method, updating from the installation discs, is a straight-forward process. Boot from the installation discs and select the option to upgrade.

8 procfs: Gone But Not Forgotten

In Linux, you may have looked at `/proc/sys/net/ipv4/ip_forward` to determine if IP forwarding was enabled. Under FreeBSD you should use sysctl(8) to view this and other system settings, as procfs(5) has been deprecated in current versions of FreeBSD. (Although `sysctl` is available in Linux as well.)

In the IP forwarding example, you would use the following to determine if IP forwarding is enabled on your FreeBSD system:

```
% sysctl net.inet.ip.forwarding
net.inet.ip.forwarding: 0
```

The `-a` flag is used to list all the system settings:

```
% sysctl -a
kern.ostype: FreeBSD
kern.osrelease: 6.2-RELEASE-p9
kern.osrevision: 199506
kern.version: FreeBSD 6.2-RELEASE-p9 #0: Thu Nov 29 04:07:33 UTC 2007
    root@i386-builder.daemonology.net:/usr/obj/usr/src/sys/GENERIC

kern.maxvnodes: 17517
kern.maxproc: 1988
kern.maxfiles: 3976
kern.argmax: 262144
kern.securelevel: -1
kern.hostname: server1
kern.hostid: 0
kern.clockrate: { hz = 1000, tick = 1000, profhz = 666, stathz = 133 }
kern.posix1version: 200112
...
```

> **Note:** Some of these `sysctl` values are read-only.

There are occasions where procfs is required, such as running older software, using truss(1) to trace system calls, and Linux Binary Compatibility (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/linuxemu.html). (Although, Linux Binary Compatibility uses its own procfs, linprocfs(5).) If you need to mount procfs you can add the following to `/etc/fstab`:

```
proc                /proc           procfs  rw,noauto       0       0
```

> **Note:** `noauto` will prevent `/proc` from being automatically mounted at boot.

And then mount procfs with:

```
# mount /proc
```

9 Common Commands

9.1 Package Management

Linux command (Red Hat/Debian)	FreeBSD equivalent	Purpose
`yum install` *package* / `apt-get install` *package*	`pkg_add -r` *package*	Install *package* from remote repository
`rpm -ivh` *package* / `dpkg -i` *package*	`pkg_add -v` *package*	Install package
`rpm -qa` / `dpkg -l`	`pkg_info`	List installed packages

9.2 System Management

Linux command	FreeBSD equivalent	Purpose
`lspci`	`pciconf`	List PCI devices
`lsmod`	`kldstat`	List loaded kernel modules
`modprobe`	`kldload` / `kldunload`	Load/Unload kernel modules
`strace`	`truss`	Trace system calls

10 Conclusion

Hopefully this document has provided you with enough to get started with FreeBSD. Be sure to take a look at the FreeBSD Handbook (http://www.FreeBSD.org/doc/en_US.ISO8859-1/books/handbook/index.html) for more in depth coverage of the topics touched on as well as the many topics not covered in this document.