FreeBSD Quickstart Guide for Linux® Users

Abstract

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

Table of Contents

1. Introduction

This document highlights some of the technical differences between FreeBSD and Linux® so that intermediate to advanced Linux® users can quickly familiarize themselves with the basics of FreeBSD.

This document assumes that FreeBSD is already installed. Refer to the Installing FreeBSD chapter of the FreeBSD Handbook for help with the installation process.

2. Default Shell

Linux® users are often surprised to find that Bash is not the default shell in FreeBSD. In fact, Bash is not included in the default installation. Instead, the Bourne shell-compatible sh(1) as the default user shell. The root shell is tcsh(1) by default on FreeBSD 13 and earlier and sh(1) on FreeBSD 14 and later. sh(1) is very similar to Bash but with a much smaller feature-set. Generally shell scripts written for sh(1) will run in Bash, but the reverse is not always true.

However, Bash and other shells are available for installation using the FreeBSD Packages and Ports Collection.

After installing another shell, use chsh(1) to change a user's default shell. It is recommended that the `root` user's default shell remain unchanged since shells which are not included in the base distribution are installed to /usr/local/bin. In the event of a problem, the file system where /usr/local/bin is located may not be mounted. In this case, `root` would not have access to its default shell, preventing `root` from logging in and fixing the problem.

3. Packages and Ports: Adding Software in FreeBSD

FreeBSD provides two methods for installing applications: binary packages and compiled ports. Each method has its own benefits:

Binary Packages

- Faster installation as compared to compiling large applications.
- Does not require an understanding of how to compile software.
- No need to install a compiler.

Ports

- Ability to customize installation options.
- Custom patches can be applied.

If an application installation does not require any customization, installing the package is sufficient. Compile the port instead whenever an application requires customization of the default options. If needed, a custom package can be compiled from ports using `make package`.

A complete list of all available ports and packages can be found here.

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`. For example, the following command installs Apache 2.4:

```
# pkg install apache24
```

For more information on packages refer to section 5.4 of the FreeBSD Handbook: Using pkgng for Binary Package Management.

3.2. Ports

The FreeBSD Ports Collection is a framework of Makefiles and patches specifically customized for installing 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 any required dependencies.

The Ports Collection, sometimes referred to as the ports tree, can be installed to /usr/ports using Git. Detailed instructions for installing the Ports Collection can be found in section 4.5.1 of the FreeBSD Handbook.

To compile a port, change to the port's directory and start the build process. The following example installs Apache 2.4 from the Ports Collection:

```
# cd /usr/ports/www/apache24
# make install clean
```

A benefit of using ports to install software is the ability to customize the installation options. This example specifies that the mod_ldap module should also be installed:

```
# cd /usr/ports/www/apache24
# make WITH_LDAP="YES" install clean
```

Refer to Using the Ports Collection for more information.

4. System Startup

Many Linux® distributions use 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 /etc/inittab does not exist. Instead, startup is controlled by rc(8) scripts. At system boot, /etc/rc reads /etc/rc.conf and /etc/defaults/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.

The scripts found in /etc/rc.d/ are for applications that are part of the "base" system, such as cron(8), sshd(8), and syslog(3). The scripts in /usr/local/etc/rc.d/ are for user-installed applications such as Apache and Squid.

Since FreeBSD is developed as a complete operating system, user-installed applications are not considered to be part of the "base" system. User-installed applications are generally installed using Packages or Ports. In order to keep them separate from the base system, user-installed applications are installed under /usr/local/. Therefore, user-installed binaries reside in /usr/local/bin/, configuration files are in /usr/local/etc/, and so on.

Services are enabled by adding an entry for the service in /etc/rc.conf. The system defaults are found in /etc/defaults/rc.conf and these default settings are overridden by settings in /etc/rc.conf. Refer to rc.conf(5) for more information about the available entries. When installing additional applications, review the application's install message to determine how to enable any associated services.

The following entries in /etc/rc.conf enable sshd(8), enable Apache 2.4, and specify that Apache should be started with SSL.

```
# enable SSHD
sshd_enable="YES"
# enable Apache with SSL
apache24_enable="YES"
apache24_flags="-DSSL"
```

Once a service has been enabled in /etc/rc.conf, it can be started without rebooting the system:

```
# service sshd start
# service apache24 start
```

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

```
# service sshd onestart
```

5. Network Configuration

Instead of a generic *ethX* identifier that Linux® uses to identify a network interface, FreeBSD uses the driver name followed by a number. 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
```

An IP address can be assigned to an interface using ifconfig(8). To remain persistent across reboots, the IP configuration must be included in /etc/rc.conf. The following /etc/rc.conf entries specify 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 entries to instead configure an interface for DHCP:

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

6. Firewall

FreeBSD does not use Linux® IPTABLES for its firewall. Instead, FreeBSD offers a choice of three kernel level firewalls:

- PF
- IPFILTER
- IPFW

PF is developed by the OpenBSD project and ported to FreeBSD. PF was created as a replacement for IPFILTER and its syntax is similar to that of IPFILTER. PF can be paired with altq(4) to provide QoS features.

This sample PF entry allows inbound SSH:

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

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.

The IPFILTER syntax to allow inbound SSH is:

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

IPFW is the firewall developed and maintained by FreeBSD. It can be paired with dummynet(4) to provide traffic shaping capabilities and simulate different types of network connections.

The IPFW syntax to allow inbound SSH would be:

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

7. Updating FreeBSD

There are two methods for updating a FreeBSD system: from source or binary updates.

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 Git repository. Once the local source code is up-to-date, a new version of the kernel and

userland can be compiled.

Binary updates are similar to using `yum` or `apt-get` to update a Linux® system. In FreeBSD, freebsd-update(8) can be used fetch new binary updates and install them. These updates can be scheduled using cron(8).

When using cron(8) to schedule updates, use `freebsd-update cron` in the 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
```

For more information on source and binary updates, refer to the chapter on updating in the FreeBSD Handbook.

8. procfs: Gone But Not Forgotten

In some Linux® distributions, one could look at /proc/sys/net/ipv4/ip_forward to determine if IP forwarding is enabled. In FreeBSD, sysctl(8) is instead used to view this and other system settings.

For example, use the following to determine if IP forwarding is enabled on a FreeBSD system:

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

Use `-a` to list all the system settings:

```
% sysctl -a | more
```

If an application requires procfs, add the following entry to /etc/fstab:

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

Including `noauto` will prevent /proc from being automatically mounted at boot.

To mount the file system without rebooting:

```
# mount /proc
```

9. Common Commands

Some common command equivalents are as follows:

Linux® command (Red Hat/Debian)	FreeBSD equivalent	Purpose
`yum install package` / `apt-get install package`	`pkg install package`	Install package from remote repository
`rpm -ivh package` / `dpkg -i package`	`pkg add package`	Install local package
`rpm -qa` / `dpkg -l`	`pkg info`	List installed packages
`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

This document has provided an overview of FreeBSD. Refer to the FreeBSD Handbook for more in-depth coverage of these topics as well as the many topics not covered by this document.