# How to perform common FreeBSD tasks on OpenBSD

binsh.dev/how-to-perform-common-freebsd-tasks-on-openbsd

binshdev April 10, 2025

Introduction

How to perform common FreeBSD tasks on OpenBSD

I'm a bit of a ThinkPad freak, and since i386 will no longer be supported in the next 15.x release of FreeBSD, I've decided to move my 32-bit machines to OpenBSD. I have quite a few of those laying around, including the X41, X41 tablet, X60, T60, R50e, and G40. They are still quite usable for lightweight tasks like email, IRC, document editing, configuring networking equipment with the built-in serial port, remotely managing other machines, etc.

This post will serve as reminder for myself on some of the most common tasks I do on FreeBSD, with the equivalent OpenBSD commands. What I consider to be "common tasks" are things like updating/upgrading the operating system, port/package management, configuring wired/wireless networking, setting up a graphical environment, mounting external drives, etc.

## OpenBSD installation tips

The OpenBSD installer, like FreeBSD, is pretty straight forward. Just grab the latest installer image from the OpenBSD website (7.6 at the time of this writing), boot it from a flash drive, and follow the instructions. I chose the (I)nstall option on the first prompt, as opposed to (A)utoinstall. Set the keyboard layout and hostname.

Next is network configuration. My ThinkPad X60 has an Atheros-based wifi card (AR5418), which is supported on OpenBSD out of the box, so I chose the athn0 interface and proceeded.

When asked if you want to start X with xenodm, choose 'yes'. xenodm is OpenBSD's display manager. Although I prefer to be dropped to a console login on boot and then use startx, I found this approach problematic in the past, so I'll stick with xenodm.

Make sure you choose the right disk when prompted (because you'll also get an entry for the flash drive you booted from). Type '?' to list additional information about the available disks. Use the (W)hole disk when prompted, followed by the (A)uto layout.

Next is file sets. 'file sets' are basically software collections for different components of your OpenBSD system (e.g. base packages, OpenBSD kernel, X-related packages, etc).

The sets are located on the installer drive; choose 'disk' when prompted then give the name of your flash drive dev node. Choose 'all' sets then proceed. In my case, it complained that SHA256.sig didn't exist, so I answered 'yes' to 'Continue without verification'. You can choose to install over http if you want to verify the checksums.

## Patching the system

To get the latest minor system updates/patches on FreeBSD, you run:

```
# freebsd-update fetch install
```

On OpenBSD, the equivalent command is:

```
# syspatch
```

If you get no output from that, then you have the latest patches.

## Major system upgrades

On FreeBSD, to do a major version upgrade, you use **freebsd-update** with the **upgrade** sub-command:

```
# freebsd-update -r 14.2-RELEASE upgrade
```

On OpenBSD, it's **sysupgrade**, a la syspatch. How easy is that?!

```
# sysupgrade
```

sysupgrade will try to install the next incremental update for you. If you want to jump directly to a specific version, use the **-R** flag. So if you want to upgrade from 7.4 directly to 7.6, you can run:

```
# sysupgrade -R 7.6
```

However, from what I've read, it's recommended to upgrade incrementally. That is, if the latest version is 7.6 and you're on 7.3, it's best to run **sysupgrade** three times: once for 7.3->7.4, once for 7.4->7.5, and once for 7.5->7.6. I suppose you'll need to reboot after each of those.

## Updating (binary) software packages

On FreeBSD, one runs the following:

```
# pkg update
# pkg upgrade
```

The equivalent utility on OpenBSD is **pkg_add**:

```
# pkg_add -uU
```

**-u** is the flag for updating packages. **-U** (in captial) tells pkg_add to also update package dependencies first.

You can also update/upgrade a specific package by supplying its name:

```
# pkg_add -uU qutebrowser
```

Note that the **-u** flag expects the package to be already installed, otherwise it will complain.

## Searching, adding, and removing packages

FreeBSD:

```
# pkg search <pkg_name>      # search for a package
# pkg install <pkg_name>     # install a package
# pkg remove <pkg_name>      # remove a package
# pkg autoremove <pkg_name>  # remove a package and its unused dependencies
```

OpenBSD:

```
# pkg_info -aQ <pkg_name>     # search for a package
# pkg_add <pkg_name>          # install a package
# pkg_delete <pkg_name>       # remove a package
# pkg_delete -a               # remove unused dependencies after deleting a
package
```

## Power/battery status

OpenBSD uses the same utility for checking battery status: **apm**.

## Connecting to a wireless network

The common approach to connect to a WPA(2) wireless network on FreeBSD is to add an entry to wpa_supplicant.conf:

```
network={
    ssid="My WiFi Network"
    scan_ssid=1
    psk="SuperSecretPassword"
    priority=10
    proto=RSN
    key_mgmt=WPA-PSK
}
```

The netif (network interface) service will pick that up and connect to your network, assuming it's available and has the highest priority in wpa_supplicant.conf (you can also run **service netif restart** to speed things up a bit).

On OpenBSD, you can connect directly with ifconfig:

```
# ifconfig <interface> nwid <ssid> wpakey <password>
```

dhclient was recently replaced with the dhcpleased daemon on OpenBSD. It should automatically negotiate a new DHCP lease for you. You could also call **dhcpleasectl** manually:

```
# dhcpleasectl <interface>
```

You may or may not need to call the netstart script afterwards:

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

Finally, you can persist this over reboots in the hostname.<interface> file. For example, my atheros card is identified as athn0, so I add the following to /etc/hostname.athn0:

```
join 'My WiFi Network' wpakey 'SuperSecretPassword'
inet autoconf
```

If you connected to a wireless network during installation, then it should have already saved that information for you.

## Connecting to a wired network

All you need to do to connect to a wired network (via DHCP) is to run dhcpleasectl against your ethernet interface. In my case:

```
# dhcpleasectl em0
```

On FreeBSD, the netif service will normally take care of that for you. I suppose on OpenBSD you can also automate this with the hostname.<interface> file. I have not tested this, but you would need to add the following to hostname.<interface>:

```
inet autoconf
```

## Mounting FAT32 flash drives

Plug your USB drive in, and check the output from dmesg for your drive's device name (should also show on the xconsole in the lower right corner of your X session).

In most cases, you'll simply want to append 'i' to your device name (e.g. /dev/sd1i) and mount that:

```
# mount -t msdos /dev/sd1i /mnt
```

The following explanation is most likely inaccurate, but the BSDs use disklabels aka bsdlabels (up to 16 partitions on a disk named 'a' through 'p', with 'c' being a special case to represent the entire disk, e.g. /dev/sd1c refers to your entire flash drive). I said 'in most cases' above because FAT32 partitions start at the ith partition, but if you have multiple partitions on your drive, then you could also use /dev/sd1j, /dev/sd1k, etc. FreeBSD is said to be dropping disklabels in 15.x in favor of GPT. I will explore the differences between MBR, GPT, disklabel, slices, partitions, etc. in a different article.

## Changing the default window manager

For an OpenBSD installation with xenodm, you can change the default (fvwm) window manager through the ~/.xsession file. I like awesome wm. You can install it with **pkg_add awesome**. Read the post-install message (you need to copy a config file), then add the following to ~/.xsession:

```
exec awesome
```

Exit the current fvwm session, re-enter your login credentials, and you should be inside awesome wm!

Happy weekend!