BSD from Source

NetBSD-centric, but should be generally applicable to other BSD flavors

I. Rationale for building from source.

A. Production Systems.

1. Resource usage: Custom Kernel.

a. "GENERIC" is (necessarily) very large.

b. Remove un-necessary device drivers/CPU support.

c. Enable advanced features/options.

d. Enable/disable kernel support for services.

2. Additional Software: pkgsrc/ports.

a. Pre-built binary may not be available.

b. –current pkgsrc/ports may have more recent version (security/bug fixes).

3. Critical/Security updates.

a. Usually involves patch to kernel.

b. May also involve patch to system utilities.

4. Low-resource target.

a. Native build on same type of machine (w/more resources).

b. Cross-build from different architecture and/or host OS.

B. Experimental Systems.

1. Try alternate configurations or features marked “EXPERIMENTAL”.
2. Tracking a future release or –current.
    a. “Be the first on your block to play with. . ..”
    b. Provide feedback for debugging.
    c. Add new function or device driver (out of scope for this presentation).

II. Setting it up.

A. What to get.

1. System source: Kernel and system utilities.
2. GNU source.
    a. not needed for native, same-version, kernel-only builds if `comp.tgz` set installed.
    b. required for all cross-builds, *tools* and non-kernel targets (*build*/*world*, *distribution*, *release*).
3. X source If you want to rebuild X11R6 packages.
4. Pkgsrc/ports for building additional software.

B. Where to get it.

1. Tarballs on release CDs.
2. From BSD project mirrors.
    a. Tarballs for particular releases.
    b. Periodic (weekly) tarball of –current.

3. anonymous CVS, CVSup, rsync, etc.
    a. Get source from any release tag or date.
    b. Can start with source tarballs.

C. Where to put it.

1. The ususal places, local or NFS-mounted.
    a. System source (`/usr/src/sys`).
    b. GNU source (`/usr/src/gnu`).
    c. X source (`/usr/xsrc`).
    d. Pkgsrc/ports (`/usr/pkgsrc` or `/usr/ports`).
2. But you can put it anywhere convenient.
    a. Use symlinks.
    b. Keep separate for cross-builds.

III. Putting it together

A. See `BUILDING`.

B. The Kernel configuration file.

1. Generally well-documented and self-explanatory.
    a. See `config(8)`.
    b. See `options(4)`.
2. Copy GENERIC and edit to suit.
    a. Use **dmesg** to get list of available hardware and any hardware-specific options.
    b. The phrase: *not configured* means that the listed device either has no driver available, or the driver may exist, but was disabled or not included in that kernel.

C. Pkgsrc/ports

1. building/installing

    a. Easy as **make** *install*

    b. installed under `/usr/pkg`, or `/usr/local`.

    c. Recommend building as unpriviledged user. Install step will prompt to become root.

    d. Advanced techniques include bulk builds inside a `chroot(8)` sandbox. Often used to produce distributable binary packages without actually installing them on the (cross-)build host.

2. NetBSD pkgsrc on Non-NetBSD targets.

    a. Bootstrapping pkgsrc (. . . `/pkgsrc/bootstrap/`)

    b. Easy as **bmake** *install*

    c. installed in location specified in bootstrap step

    d. Items *c.* and *d.* from previous subtopic also apply.

3. Considerations for services/daemons built from pkgsrc.

    a. config files remain in `/usr/pkg/etc`

    b. copy `/usr/pkg/etc/rc.d/`*service* to `/etc/rc.d/` if filesystem with `/usr/pkg` not yet mounted at init.

    c. *service*'s rc script usually indicates the variable to be set in `/etc/rc.conf` to cause *service* to be started at boot.