Tracking FreeBSD in a Commercial Environment

Warner Losh
imp@FreeBSD.org

The FreeBSD Project

BSDCan 2009 — Ottawa, Canada
8 May 2009
Outline

1. Background and Context
2. FreeBSD Development Model
   - Theory
   - Reality
3. Product Life Cycle
   - Product Life Cycle
   - Upgrading
   - Bad FreeBSD Experience
4. Tracking Options
   - Grab and go / upgrade
   - Track Stable Branches
   - Mirror FreeBSD’s development process
   - Major porting to a new release
5. SVK Hints
Outline

1. **Background and Context**
2. **FreeBSD Development Model**
   - Theory
   - Reality
3. **Product Life Cycle**
   - Product Life Cycle
   - Upgrading
   - Bad FreeBSD Experience
4. **Tracking Options**
   - Grab and go / upgrade
   - Track Stable Branches
   - Mirror FreeBSD’s development process
   - Major porting to a new release
5. **SVK Hints**
FreeBSD Based Development

- Develop an initial product based on FreeBSD
- Development on product continues
- Development on FreeBSD continues
- Product needs a newer FreeBSD
- Now what?
FreeBSD Based Products

- Product includes BSD
- May be used unmodified
- May have extensive local changes
- May include custom software
- May have custom settings
Outline

1. Background and Context
2. FreeBSD Development Model
   - Theory
   - Reality
3. Product Life Cycle
   - Product Life Cycle
   - Upgrading
   - Bad FreeBSD Experience
4. Tracking Options
   - Grab and go / upgrade
   - Track Stable Branches
   - Mirror FreeBSD’s development process
   - Major porting to a new release
5. SVK Hints
FreeBSD Branching Model

- Main development branch “Current”
- Periodic major releases
- Major release creates new stable branch (aka RELENG_X)
- Minor releases done from stable branch
- Security/Errata branches
FreeBSD Code Work Flow

- Patches Submitted to the Project
- Code reviewed
- Code goes into “Current”
- Code refined, if necessary, based on testing
- Code merged to RELENG_X branch
- New release off RELENG_X called X.Y
Theoretical Release Schedule

- New major release every 18-24 months
- Branch active 24 months
- Branches terminate after 3 years
- Gradual reduction in activity
- Usually one stable branch active, plus “current”
- Worst case, two branches are active
Theoretical Release Schedule

Idealized 2 year release cycle

- HEAD
- RELENG_8
- RELENG_7
- RELENG_6
- RELENG_5
- RELENG_4
- RELENG_3
- RELENG_2
- RELENG_1
- RELENG_0

Release versions are plotted against time, with the theoretical end of life for each version marked.

Warner Losh
Tracking FreeBSD in a Commercial Environment
Actual Release Schedule

- Major releases not evenly spaced
- Branches can live for a long time
- Some branches get more attention
- Pent up demand and code freezes distort graph
- On the average, especially lately, we match theory
- Graphs can be misleading

Warner Losh

Tracking FreeBSD in a Commercial Environment
Actual Release Schedule

Cumulative Commits For FreeBSD Branches

- HERD
- RELENG_7
- RELENG_6
- RELENG_5
- RELENG_4
- RELENG_3
- RELENG_2_2
- RELENG_2_1

Warner Losh
Tracking FreeBSD in a Commercial Environment
Outline

1 Background and Context
2 FreeBSD Development Model
   - Theory
   - Reality
3 Product Life Cycle
   - Product Life Cycle
   - Upgrading
   - Bad FreeBSD Experience
4 Tracking Options
   - Grab and go / upgrade
   - Track Stable Branches
   - Mirror FreeBSD’s development process
   - Major porting to a new release
5 SVK Hints
Product Life Cycle

- Import FreeBSD and other software
- Make modifications and customizations
- Maybe develop applications
- Release the product
- What happens next?
  - Profit!
  - New Release?
  - Bubble Brust?
Product Life Cycle

The daydreams of cat herders

Source: Doctor Fun by David Farley
Problems Upgrading

- First version is easy, later versions hard
- Forward porting local modification
- Forward porting applications
- Bug fixes
- Managing change, both upstream and local
- Conflicts between FreeBSD bug fixes and local bug fixes
Why Upgrade?

- New hardware support
- New features (SMP, threads, devices, gcc, etc)
- Better performance
- Bug fixes
- Easier integration into FreeBSD community
A Bad FreeBSD Experience

- Import FreeBSD code into a product
- Modify FreeBSD heavily
- Limited community involvement
- Release products, make money, celebrate
- Time passes
- Pent up demand forces FreeBSD upgrade
- Major porting effort
- Few community ties to ease effort
What to do?

- How can the pain be avoided
- Where to find advice on best practices
- Learn from other’s misfortune
- Leverage the community
- Plan for upgrades
- Bug fixes aren’t a competitive advantage
Outline

1. Background and Context

2. FreeBSD Development Model
   - Theory
   - Reality

3. Product Life Cycle
   - Product Life Cycle
   - Upgrading
   - Bad FreeBSD Experience

4. Tracking Options
   - Grab and go / upgrade
   - Track Stable Branches
   - Mirror FreeBSD’s development process
   - Major porting to a new release

5. SVK Hints

Warner Losh  Tracking FreeBSD in a Commercial Environment
Tracking Options

- Grab and go / upgrade
- Track Stable Branches
- Mirror FreeBSD’s development process
- Major porting to a new release
Grab and Go

- Grab a version of FreeBSD
- Make changes to FreeBSD
- Never upgrade or participate in Community
Grab and Go, Pros

- Easy
- Simple
- Management understands
- No interaction with community
Grab and Go, Cons

- Upgrades Hard
- Difficult to add local changes
- New features of FreeBSD not reflected in products
- New hardware often needs new OS support
- No interaction with community
Grab and Upgrade

- Grab a version of FreeBSD
- Use unmodified
- Upgrade as needed
Grab and Upgrade, Pros

- Easy
- Simple
- Management understands
Grab and Upgrade, Cons

- Difficult to add local changes
- New hardware can be slow to appear in a release
Tracking Major Branches

- Import major release sources into local SCM
- Make local changes to SCM
- Merge minor releases into SCM
- Each major branch has its own SCM model
Tracking Major Branches

- Current
- 7.x
  - Company 7.x
- 6.x
  - Company 6.x
- 5.x
  - Company 5.x
Tracking Major Branches, Pros

- Get bug fixes from FreeBSD
- Local bug fixes tracked
- Local bug fixes easy to push upstream
- Stable branches best place to base release
Tracking Major Branches, Cons

- Multiple Branches
- Local changes not automatically included on major upgrades
- Current/stable divergence makes some fixes hard to push upstream
- Major releases not completely predictable
Mirroring FreeBSD’s Development Process

- Import FreeBSD-current into SCM
- Maintain branch of current + local changes
- Make own stable branches
- Push changes to FreeBSD early and often
Mirroring FreeBSD’s Development Process

- Grab and go / upgrade
- Track Stable Branches
- Mirror FreeBSD’s development process
- Major porting to a new release

Warner Losh
Tracking FreeBSD in a Commercial Environment
Mirroring FreeBSD’s Development Process, Pros

- Changes typically easy to merge both directions
- Continuous porting amortizes upgrade pain
- Much community involvement
- Choice of time to cut stable branch
Mirroring FreeBSD’s Development Process, Cons

- Duplicating effort done by FreeBSD’s release engineering
- Internal stable branch gets less testing
- Management views work as being done twice
- Harder to get help from community on problems on private branch
Major Porting Effort

- Start with Grab and Go, no plans to upgrade
- Lots of time passes
- Upgrade required for new features/devices/etc
- Major efforts, much pain, desire to do it better
Outline

1. Background and Context
2. FreeBSD Development Model
   - Theory
   - Reality
3. Product Life Cycle
   - Product Life Cycle
   - Upgrading
   - Bad FreeBSD Experience
4. Tracking Options
   - Grab and go / upgrade
   - Track Stable Branches
   - Mirror FreeBSD’s development process
   - Major porting to a new release
5. SVK Hints
Creating a SVK Repository

- svk depotmap yoyobsd /path/to/repo
- svk mirror svn://svn.freebsd.org/base/yoyobsd/mirror/FreeBSD
- svk sync /yoyobsd/mirror/FreeBSD
Creating a company branch

- svk cp /yoyobsd/mirror/FreeBSD/stable/7
  /yoyobsd/yoyodyne/7
- use svk to manage merges, or interacting with mirror etc
Checking out a YoyoBSD branch

- `svn co file:///path/to/repo/yoyodyne/7`
Merging Changes from FreeBSD

- `svk sync /yoyobsd/mirror/FreeBSD`
- `svk smerge /yoyobsd/mirror/FreeBSD/stable/7 /yoyobsd/yoyodyne/7`
- Note: you can pass -C to check before committing
Optional: Tag it

- `svn cp`
  
  file:///path/to/repo/mirror/FreeBSD/stable/7
  
  file:///path/to/repo/freebsd-sync-7/YYYYMMDD
Optional: Tag it

- Real svn repo
- history works
- merging works
- NB: change numbers differ
Questions? Comments?
Warner Losh
imp@FreeBSD.org