



finstall

A GUI-based installer for FreeBSD

by Ivan Voras
<ivoras@freebsd.org>

Sponsored partially by
Google Summer of Code 2007.
Mentor: Murray Stokely



The problem with sysinstall

- Sysinstall works – undisputably
 - If you know what option you want
 - If you know when "Enter" checks the checkbox and when it confirms the dialog
 - If you know when the OK and CANCEL options work and what exactly will they work on
 - If you know when going "Back" works and when it's going to crash
 - If you don't need any new feature developed in the last 10 years



Project goals

- A modern, graphical installer for FreeBSD 7+
- Usable both by experts and novice users
- Support for several installation modes
- Support for modern FreeBSD features

~~**sysinstall**~~



Architecture

- Front-end
 - GUI using GTK, in Python
- Back-end
 - System daemon, in Python
 - Calls FreeBSD system utilities (like fdisk, disklabel)
- Miscellaneous scripts (ISO building scripts, etc.)
- Front-end and back-end communicate:
 - XML-RPC
 - UDP broadcasts for discovery



Front-end

- PyGTK
 - Not "full" Gnome, to make it smaller
- Wizard-like interface
- Modular
 - Uses Glade GUI editor for rapid development
 - Uses customizable text files for (almost all) messages
 - Important for localization
- Is mostly a standard, boring PyGTK application



Back-end

- "systoold"
- Currently written in Python
- XML-RPC server (mostly stateless)
- UDP broadcasting of "I'm here" announcements
- Invokes standard FreeBSD utilities to do the actual work
 - Newfs
 - Disklabel
 - Fdisk



Front-end and back-end

- Front-end and back-end are completely independent and replaceable
 - As far as the front-end is concerned, it might be installing Linux
 - Back-end contains generic functions – it doesn't "know" it's being used in the installer
- Example XML-RPC invocations:
 - GetDMESG()
 - GetMountPoints()
 - SetConf()



Great new possibilities

- The original Google Summer of Code project was for an installer, but...
- The back-end (systool) is usable as a system configuration tool
- Separately from the front-end!



Great new possibilities

- Automated / mass installation
 - Remote installation
- Automated administration
 - Either through GUIs or by scripting XML-RPC calls
- Remote administration of multiple systems from a single point
 - e.g. "install this package on these" machines
 - Add this line to /etc/rc.conf on these machines



What works?

- **Can install a full, working FreeBSD base system plus X.Org, Python, Ruby, Firefox...**
- Remote installation support
- Multiple file-system support (UFS, gjournal, ZFS, ext2), uses *glabel* for mount points
- Can configure NICs
- Can configure users
- Can configure basic services (sshd, ntpd, ...)
- Generates call-log of XML-RPC invocations

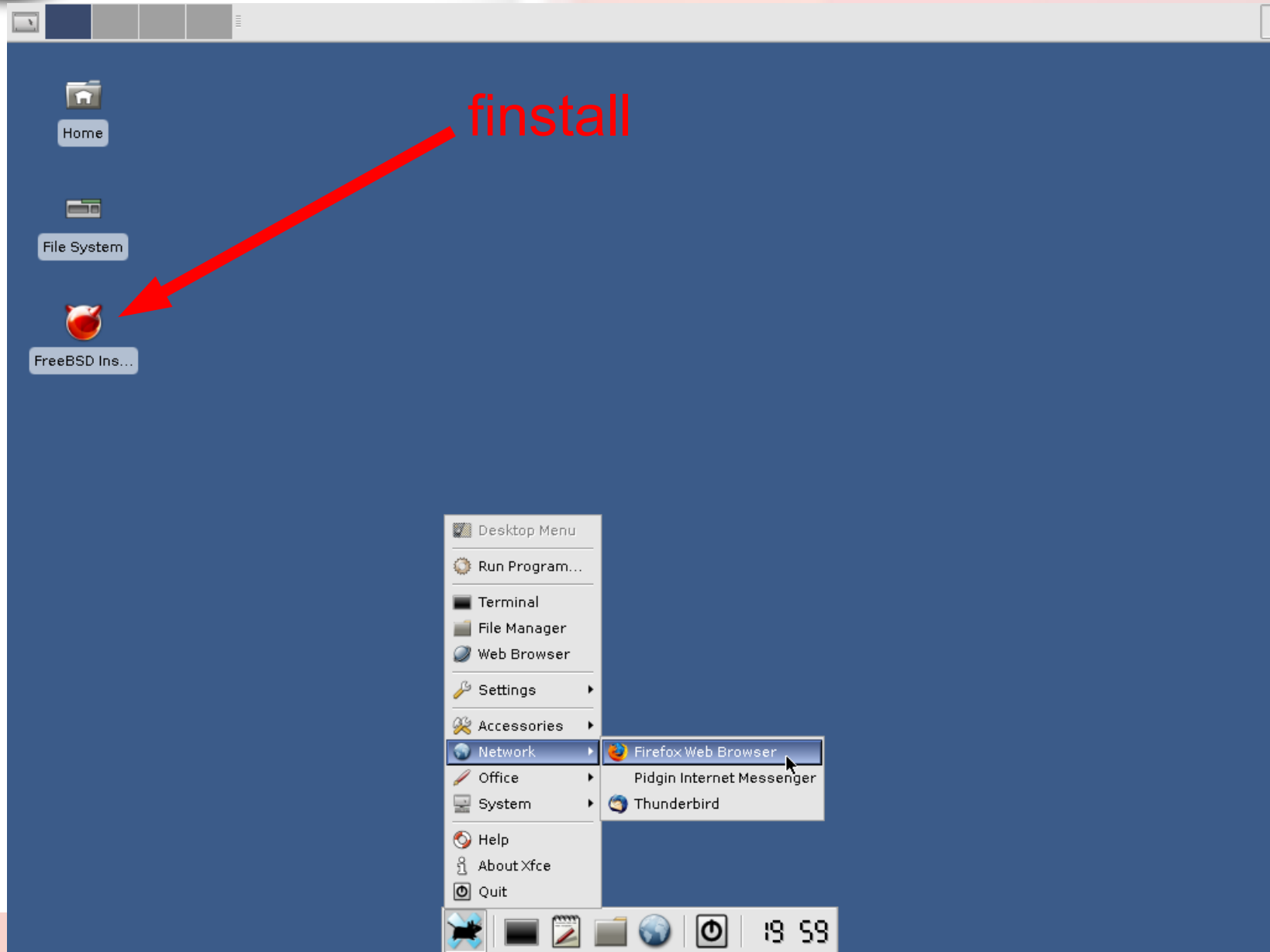


What's still missing?

- User-driven partition support (aka graphical fdisk+disklabel)
 - GPT support is also missing (implemented too late in FreeBSD)
- Software RAID support
- Package selection support
 - Currently everything from the live CD is installed
- X.Org configuration support
- Graphic card, sound card, etc.



Screenshot: Desktop





LiveCD based on 7.0-RELEASE

- Boots "normal" FreeBSD base from the CD
- Root on ISO9550 (read-only)
- Unionfs mounts memory-based UFS over significant directory trees (/var, /tmp, /etc, ...)
- X.Org 7.3 starts from this live system
- Installer application is available as a desktop shortcut
- A "normal" PyGTK application in all aspects



Technologies for LiveCD

- Almost everything is in the FreeBSD-base
- `make installworld DESTDIR=...`
- `make distribution DESTDIR=...`
- CD is booted directly, root is mounted directly
- mdconfig, memory-backed file system used with zlib-compressed images to store the /usr/local tree
- UnionFS mounted writeable memory file system over important bits (e.g. /var/log)

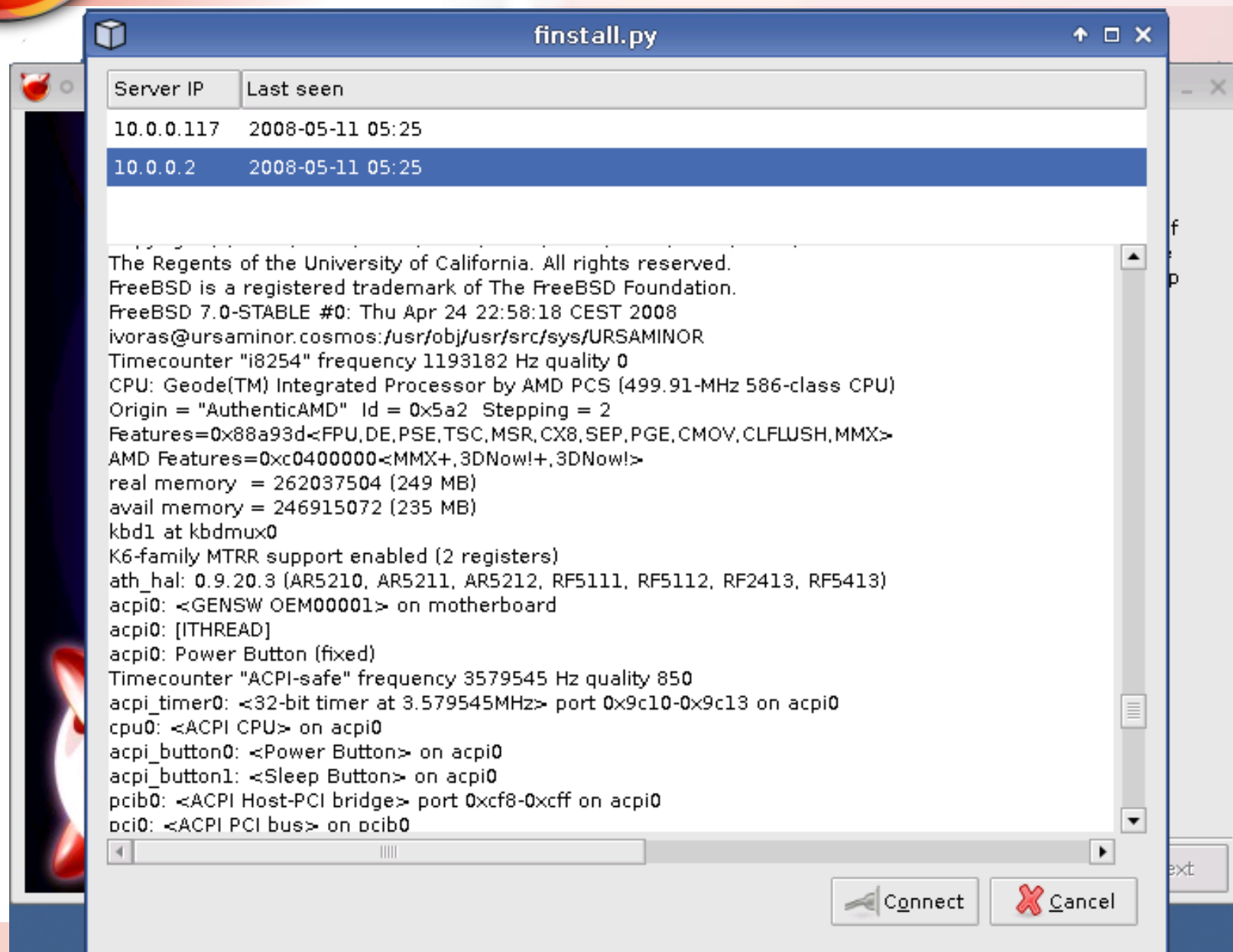


Side-effect: makeimage

- Finstall currently doesn't use an existing LiveCD system such as FreeSBIE
 - At the time of development there were features missing or broken
- Created script to make FreeBSD LiveCD ISO images: `makeimage.py`
- `./makeimage.py -d /builddcd -p pkglist -i /images/finstall.iso`
- In the future, other LiveCD systems may use finstall



Screenshots: Select remote server





Remote install

- Booting from the finstall CD by default (this can be canceled by the user) starts XML-RPC servers on available NICs (initialized by dhclient), and also a UDP broadcaster
- Front-ends can listen for UDP broadcasts to locate nodes
- Front-ends can connect to remote nodes, perform install "as usual", no difference



How does remote install work

- Systoold back-end is an XML-RPC server
- XML-RPC is a standard, widely used protocol
- It's stateless, based on HTTP transport
 - SOAP evolved from it, but it's way too complex
- The back-end doesn't care who uses it – if the GUI front-end is a local process or a remote one
- During the local installation, the front-end is run locally, for remote administration it connects over the network



"Plain" remote install

- Insert the CD with finstall in a server and connect it to the network
 - A "headless" server
- After finstall-enabled CD boots, start the front-end somewhere
- The front-end will listen for backend broadcasts and will locate it (optionally display its dmesg)
- Connect to the chosen back-end and perform the install as it were local

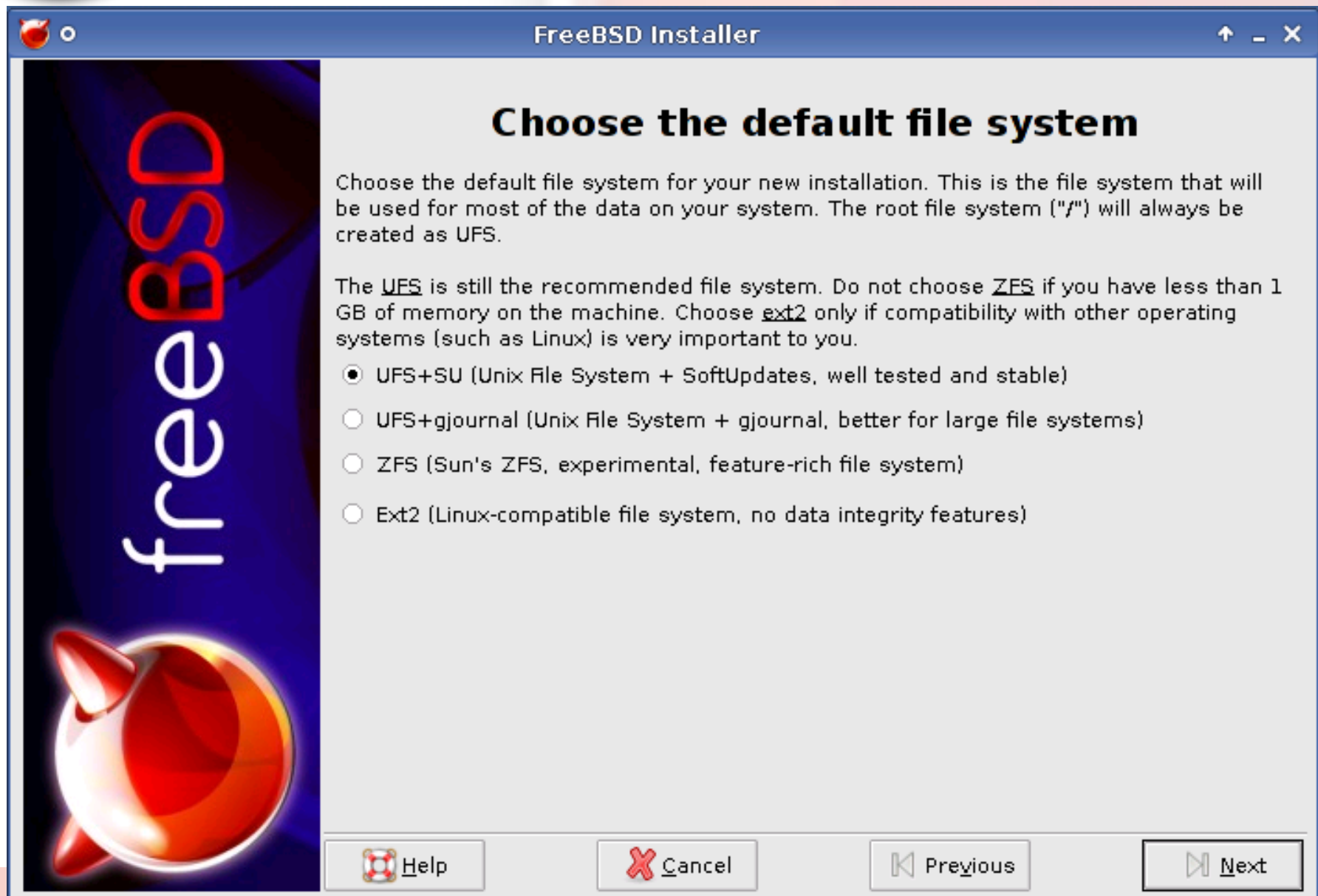


Advanced variations

- Scripted / mass installs
 - PXE boot multiple servers into finstall-enabled environment
 - Write a (Python or whatever) script containing XML-RPC calls to the remote server(s)
- Remote configuration
 - Requires systool to always be running on usable machines
 - Need to consider security implications



Screenshot: Choose file system





File system support

- Does "special" things in each case:
 - UFS+SoftUpdates
 - gjournal flags, kernel module, device label
 - ZFS cannot be used on root, needs tuning in loader.conf, enabling in rc.conf
 - Ext2 cannot be used on root
- Uses native labels (UFS, ext2) as much as possible
 - root is on `/dev/ufs/root`



File system restrictions

- Currently 7.0 can only boot from UFS
- ZFS & others may not be used on root/boot
- ZFS is sensitive, experimental
- Ext2 doesn't have journalling

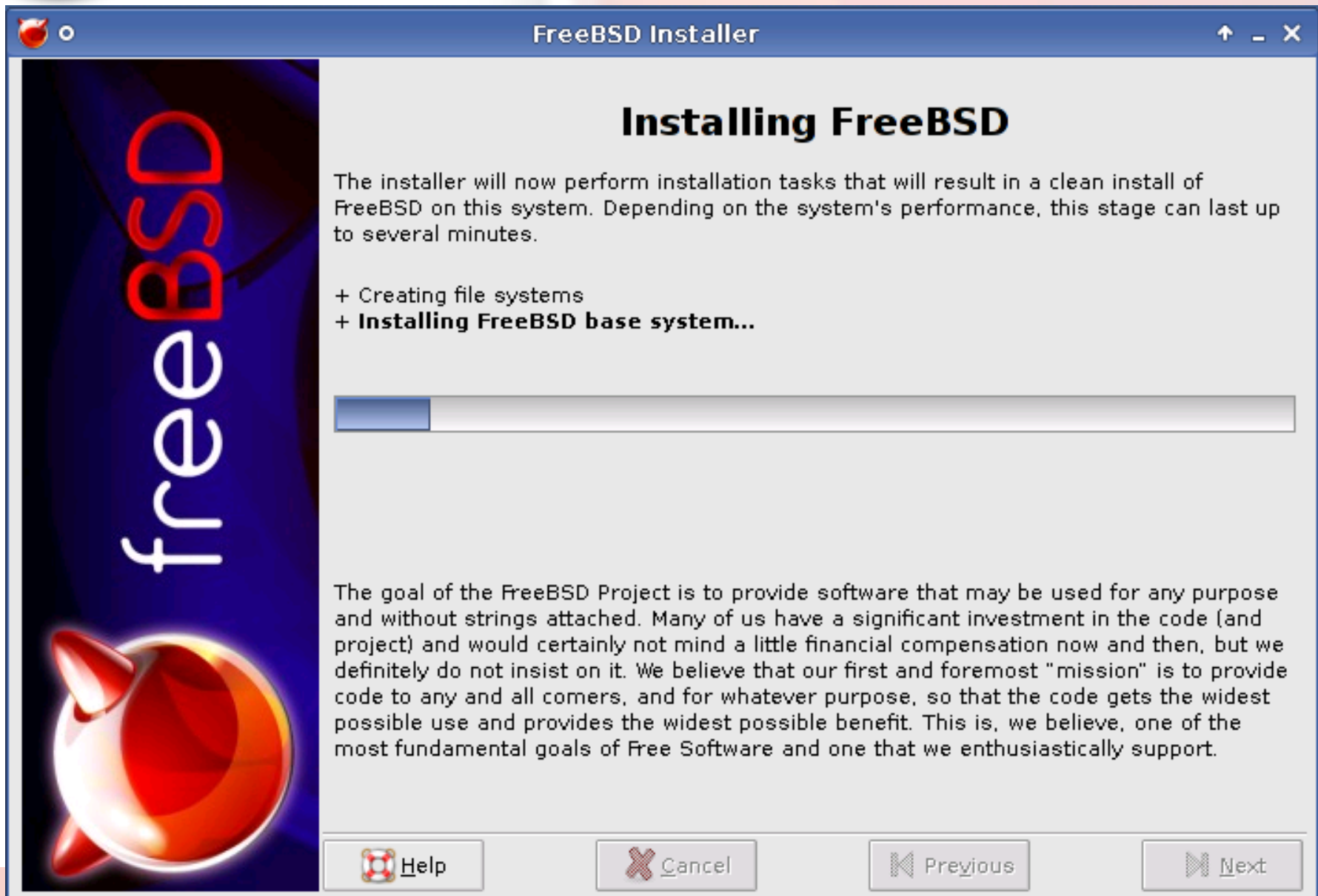


File system - overview





Screenshot: Progress





Install jobs (long-term)

- Server (systoold) spawns thread to perform long lasting jobs
- Client periodically polls for progress
- Only back-end jobs that explicitly maintain some state at the back-end
 - There are "implicit" states like: "a file system is newfs'ed"
- Advertising opportunity :)



Install progress

- Currently, entire base system and all packages are transferred
- Not particularly good in case the install footprint needs to be kept down
- Reasonably fast since there's no explicit package (de)compression phase



Configure host & root

FreeBSD Installer

Configure host name and a user

The first steps in configuring your FreeBSD system are choosing a host name for it (in the form "host.example.com") and defining a password for the root user.

If the system will be a part of a company or private network, you should choose the host name in accordance with your existing network layout. If not, it's recommended that you choose a name in the form "name.home.net".

Choose a root password that's hard to guess, or press "Generate" to generate a random hard to guess password.

Host name:

Root password:

Confirm password:

 **freebsd**

 [Help](#)  [Cancel](#)  [Previous](#)  [Next](#)



Add a user

FreeBSD Installer

Create a system user

You will need to create a regular user account for every day use. You should use only this account for your daily interaction with the system, and resort to the superuser account ("root") only for system maintenance.

Optionally, the user can be made a member of the superuser group ("wheel"), which will allow it to switch (via "su") to the superuser account for common administrative tasks.

Your name:

Account username:

Password:

Confirm password:

Shell:

☒ Member of the superuser group?

Help Cancel Previous Next



Screenshot: configure NICs

FreeBSD Installer

Configure network devices

To enable the system to participate in a network, you need to configure the network settings and the network interfaces. You can assign IP addresses to each of the network interfaces or you can enable DHCP on them. At this time you can enable the network firewall (recommended) and enable IPv6 networking.

☐ Enable IPv6 Networking Default IPv4 gateway:


☐ Enable network firewall (ipfw) DNS server:





Device	Address	Netmask
le0	DHCP	

☒ Use DHCP?

IPv4 address:

IPv4 network mask:

 freebsd

 Help  Cancel  Previous  Next

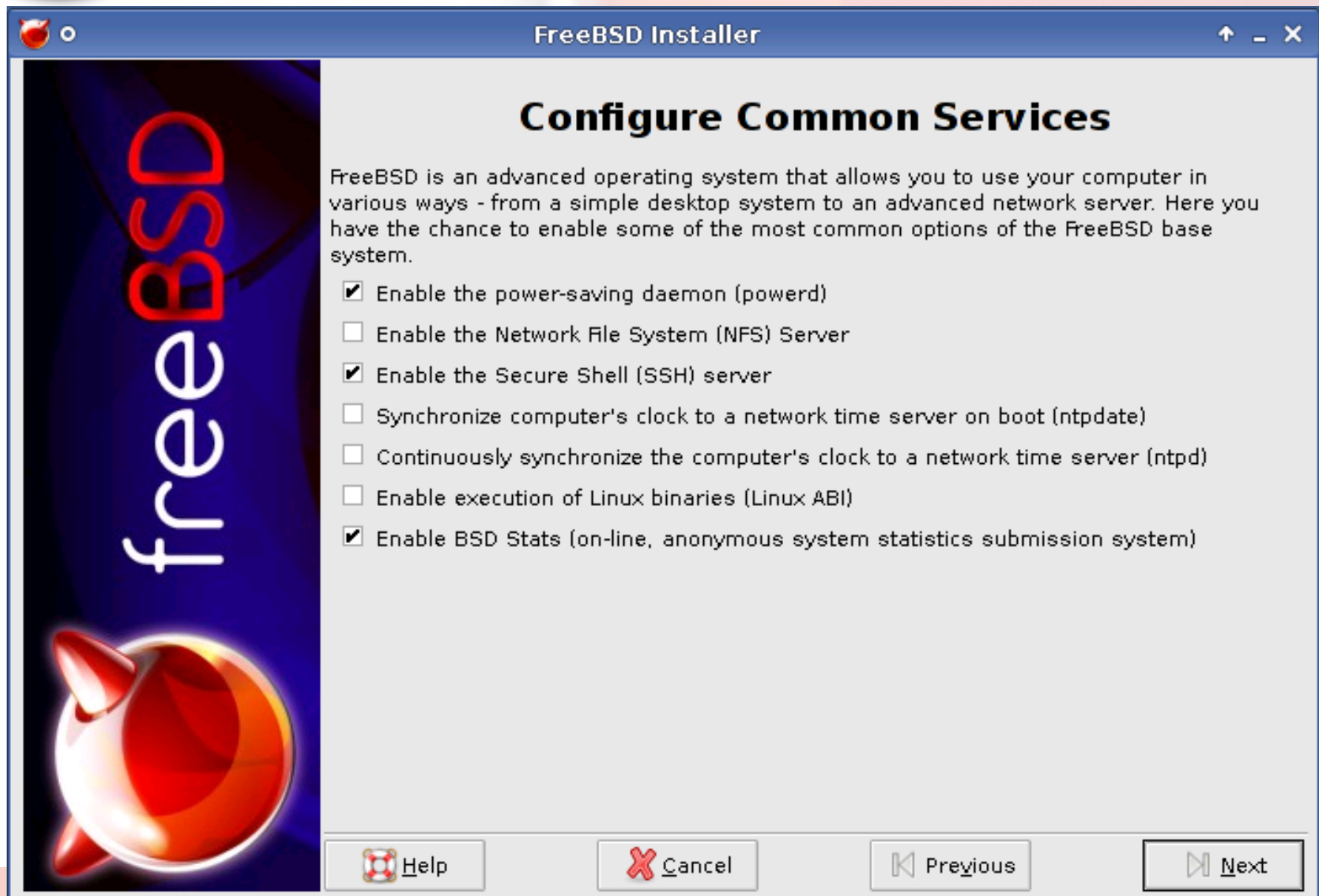


NIC configuration

- Configure each NIC separately
 - Static or DHCP
- Additional options:
 - ipfw
 - ipv6 (only enable/disable)



Screenshot: Services





Services

- For now, only simple services
 - sshd
 - ntpdate, ntpd
 - bsdstats
 - Linuxulator
 - powerd
 - Nfsd
- Editing rc.conf is cheap :)



Boot start of systool

```
Firewall rules loaded.
net.inet.ip.fw.enable: 1 -> 1
Generating host.conf.
Additional IP options:.
Mounting NFS file systems:.
ELF ldconfig path: /lib /usr/lib /usr/lib/compat /usr/local/lib /usr/local/lib/c
compat/pkg /usr/local/lib/compat/pkg /usr/local/lib/nss
a.out ldconfig path: /usr/lib/aout /usr/lib/compat/aout
Clearing /tmp (X related).
Creating and/or trimming log files:.
Starting syslogd.
/etc/rc: WARNING: Dump device does not exist.  Savecore not run.
Initial i386 initialization:.
Additional ABI support:.
Starting local daemons:
=====
=== SECURITY WARNING: At least one active Network Interface Card was ===
=== configured on this system.  Press "SPACE" or "ENTER" within 10 ===
=== seconds to disable remote install and configuration server.  If ===
=== you allow this to proceed, the machine will be opened to remote ===
=== control during the install procedure, which includes the ===
=== possibility of complete data loss. ===
=====
The configured NICs are: le0 (10.0.0.117)
Waiting: 1... 2... 3... 4... 5... 6... 7... █
```



Future possibilities

- Back-end to be rewritten in C so that it can be included in the base system
- Or, import Python in the base system?
- Make the systool the official configuration tool, usable locally or by third-party tools
- Port to other operating systems?



Future development

- Graphical partition editor
 - Software RAID (gmirror)
 - Package selection
-
- X.Org configuration
 - Multimedia device configuration



Project future

- Was sponsored by Google for SoC 2007
 - Project successfully completed
- No sponsorship or funding right now
 - Development progressing slowly, need to focus on other jobs
- Open to suggestions and future possibilities
- I rely very much on users' feedback – if you don't complain about it, it won't be done



Availability

- Will not be a part of official FreeBSD until it's finished
 - Don't look for it on www.freebsd.org
- Up-to-date information about it is announced here:
 - <http://blogs.freebsdish.org/ivoras/>
 - <http://tinyurl.com/5qpfo9>
- Major updates will be announced on `freebsd-stable` and/or `freebsd-current` mailing lists



Thank you / Questions?

- Thank you for listening to a presentation of **finstall (a graphical FreeBSD installer)**
- Ivan Voras <ivoras@freebsd.org>
- Sponsored in part by Google (Summer of Code 2007), mentor Murray Stokely