PC-SYSINSTALL
-
A new system installer backend for PC-BSD & FreeBSD

Kris Moore

kris@pcbsd.org
PC-BSD Software / iXsystems

Abstract

The sysinstall tool has been the default system installer for FreeBSD for more than a decade now. While is it has proven itself to be reliable and resilient over the years, it doesn't support many of the new features that FreeBSD offers, as well as being un-intuitive for desktop users, who expect an easy to use graphical front-end to perform their installation. To solve these two problems the “pc-sysinstall” backend was created and now is in usage for PC-BSD 8.0. This new installer backend provides much of the same functionality as sysinstall, while offering many new features such as support for ZFS, Encryption, mirroring, scriptable installs and the ability to work with different front-ends, such as a QT based GUI. The backend also supports installing regular FreeBSD, which allows server administrators to quickly perform an installation using the new disk features it offers.

Introduction

Since its very first beta many years ago, PC-BSD has been using a custom-built installer routine, which consisted of a graphical user interface, tied into some scripts which performed the actual installation process of the system. While the process worked reasonably well, it lacked many important features which would become desired and critical down the road, such as:

- automated “scriptable” installation
- an independent installer backend with interchangeable front-ends
- support for advanced custom partitioning
- full error logging
- ZFS on root
- Geli Encryption
- Gmirror support
- Remote installations
- Restore from backup
- Advanced localization options

In addition to these features, many times the idea had been brought up about enabling the new installer to also support traditional FreeBSD installations as well. This could be used as a way for users to bypass regular “sysinstall” and install using the new features that we offer, without forcing them to install a complete desktop with Xorg and KDE.

Late in the spring of 2009 as we began looking ahead to the eventual release of PC-BSD 8.0, based on FreeBSD 8.0, I decided that it was time to take these ideas and make them a reality. I evaluated several existing FreeBSD installation backends already in existence and found each of them to have strengths, but not be exactly what we had in mind for PC-BSD. Knowing this, work began shortly thereafter on “pc-sysinstall” and the new backend was merged into our subversion repository late that fall. This new backend supports a myriad of new features and abilities and will be the default installation system on PC-BSD 8.0 and future releases going forward.

Program flow design

As I began to approach the design of the new installer backend, there were several issues which needed to be resolved beforehand. In my analysis of other installer backends, I found that there were a few design options relating to how they interacted with a front-end. In the case of our existing installer, the front-end actually performed all the logic of the installation process itself, and as such was not able to be abstracted enough to allow automated installations with a supplied configuration file. This was the design we were trying to break free from, by creating a backend which could perform the entire installation independently.

On the other extreme, some of the backends I reviewed performed 100% of the logic, including program flow for the front-end during the data collection phases. The backend, once started would handle all user input, direct which dialogs to display to an end user, and decide where the user could go at any given time. This model also didn't seem to fit in with the flexibility we desired, especially when working with a powerful GUI toolkit such as QT.

After reviewing these options, I decided the approach we should take was to create a “hybrid”. A backend, which once given a valid configuration file, would be able to perform a complete system installation unattended, however the task of actual program flow for the end user would be left up to the front-end, allowing a variety of front-ends to be developed or modified without requiring extensive changes to backend functionality. To assist the front-end in their program flow, the backend would supply different commands which could be used to query specifics about the system, such as the available disks and information on them, network card availability, locale data and more. In addition to these queries, the front-end could supply the backend with arguments to take specific actions prior to an installation, such as enabling networking, setting up ssh keys, querying for available installation servers, etc. This method ensured that when designing a frontend, the developer would be given maximum flexibility in creating the user-flow best suited for his program's needs, without having to delve into updating a separate backend code base.

Development language and toolset choices

With the program flow settled, the next major decision I was faced with was the choice of programming language to use, along with any tools which the backend would rely on. While there are a variety of languages and tools readily available for such a task, my main concern was trying to keep the new backend functional with little to no additional programs or libraries required than was already available on a standard FreeBSD installation. In addition to this requirement, I wanted to keep the backend in a format which could be easily modified “on-the-fly” on a booted installation disk, without requiring re-compiling tools or libraries which may not be present on a minimal installation image. This would greatly accelerate the development process, while providing an easy way to debug, and test potential fixes in a live environment. In the end I decided that all of this could be easily accomplished by creating the backend as a shell script (/bin/sh), and using only command-line utilities which are standard to FreeBSD such as fdisk, fetch, bsdlabel, glabel, zfs and others.

pc-sysinstall features and configuration

After making the decisions on program flow and format, development began immediately on the new installer and progressed quickly. On December 2 2009 the new installer backend and frontend (Illustration 0) were both committed as the defaults for PC-BSD 8.0, and have been in usage since then. The new backend improves upon our existing installer, while also offering many new features, such as:

- Able to choose between installing vanilla FreeBSD or PC-BSD
- Support for enabling gmirror across two drives

- Complex partition layouts, using variety of file systems:
  - UFS
  - UFS with softupdate
  - UFS with Journaling
  - ZFS
- GELI-based encryption
- File system labeling with glabel
- Installation logging and debugging output
- Timezone and localization configuration
- Install using source files from the Internet or local network
- Perform upgrades of existing PC-BSD systems
- Install using custom-rolled system images or backups

From an end-user perspective using pc-sysinstall is fairly straight forward, and somewhat similar to creating a traditional FreeBSD sysinstall configuration script. A front-end is simply a tool which gathers user input on all the various installation options, then generates a working installation configuration file, which it then calls the backend to run with. By querying the backend independently, the front-end developer is able to fine-tune the workings and look their application, providing different and unique ways in which to allow the user to select their options, from disk management (Illustration 1) to user setup.

Let us take a look at some of the specifics of a pc-sysinstall configuration file, and how it can be used to install either a FreeBSD or PC-BSD system. After running the SysInstaller frontend, it will generate a pc-sysinstall based configuration file, and save it to /tmp/sys-install.cfg. Within this file will be all the options necessary to perform an installation or upgrade of a system. The beginning of the file sets some basic options common to all types of installations:

```
# Auto-Generated pc-sysinstall configuration
installInteractive=no
installMode=fresh
installType=PCBSD
packageType=uzip
```

The configuration file syntax follows some traditional standards, such as placing comments with the “#” sign, and all options are specified in the format of keyword=<setting>. Settings can be placed in almost any order, with a few exceptions for disk-layout and user information blocks. In these first few lines, we are instructing pc-sysinstall to run completely non-interactively, no prompting via the command-line for anything, specifying that this is a “fresh” install as opposed to an “upgrade”. Also, with the installType= and packageType= we are specifying which type of system is being loaded (PCBSD or FreeBSD), and what format the installation archive is in (tar or uzip). While all of the available options are documented in the examples/README file within pc-sysinstall, there are a few notable cases we will want to take a look at, specifically relating to disk management.

Disk configuration with pc-sysinstall

```
# Disk Setup for ad1
disk0=ad1
partition=ALL
bootManager=none
mirror=ad2
mirrorbal=round-robin
commitDiskPart
```

When specifying the disk(s) we want to format and install to there are a couple of specific blocks of instructions needed in order for pc-sysinstall to perform the tasks correctly. The first section is for specifying the target drive and partition (or slice). This section must begin with “disk0=”, with subsequent disks being labeled “disk1=”, “disk2=”, etc. The partition keyword indicates the where the installer will we be formatting and installing a label, either “all” for an entire disk, s1-s4 for an existing

primary partition, or “free” which takes free disk space and creates a new primary partition for it. The bootManager keyword is used to specify if the FreeBSD MBR (boot0) should be installed onto the disk drive or not, by setting it to “bsd” or “none” respectively. Also in this example we provided information on mirroring, specifying that the disk “ad2” will configured as a gmirror of “ad1”, using the balance method of “round-robin”. Lastly the commitDiskPart command instructs pc-sysinstall that we are finished specifying disk setup options for this drive/slice. If the installer is to setup multiple disks, then a similar codeblock will be required for each disk with their respective options.

```
# Partition Setup for ad1(ALL)
# All sizes are expressed in MB
# Avail FS Types, UFS, UFS+S, UFS+J, ZFS, SWAP
# UFS.eli, UFS+S.eli, UFS+J.eli, ZFS.eli, SWAP.eli
disk0-part=UFS+S 2048 /
disk0-part=SWAP.eli 14336 none
disk0-part=UFS+S 1024 /var
disk0-part=UFS+J 250000 /usr
disk0-part=ZFS 686456 /data
commitDiskLabel
```

After the specifying of the disk slice/partition information, pc-sysinstall will next require a section detailing how the individual mount-points and file systems are to be setup on this disk. In the DiskLabel section section above, we can see a complete file-system layout for the target disk., In this example **disk0** would correspond to the **disk0=ad1** specified in the previous “DiskPart” configuration block. The disk0-part= keyword is unique, in that it takes 3 arguments, separated by spaces, which are used to indicate the file-system type, size in megabytes, and mount points respectively. When specifying the filesystem, the “.eli” extension is special, indicating that geli encryption should be enabled for this partition. Partition letters will be automatically assigned and created from this configuration, such as ad0s1a for /, ad0s1b for SWAP and ad0s1d-h for additional filesystems. In addition to setting up the partitions, pc-sysinstall will also automatically generate labels for the devices using glabel, and reference those in the auto-generated /etc/fstab. A mountpoint for “/” would be given a label such as “/dev/label/root[0-9]” or swap would become /dev/label/swap[0-9]. Other filesystems would take the directory name of the mount-point and use it as a label, so that /data would become /dev/label/data[0-9]. Lastly the commitDiskLabel command must be given, instructing pc-sysinstall that we are finished specifying file-systems and ready to apply the settings to disk. As with the DiskPart configuration block, any subsequent disks will each require their own DiskLabel section.

Advanced commands for pc-sysinstall

In addition to the disk configuration options available in pc-sysinstall, it also provides some additional commands which can be used to further customize an installed system. First among these are networking configuration options, which are used to customize the systems network interfaces, so they are available at first bootup:

```
netSaveDev=AUTO-DHCP

or

netSaveDev=em0
netSaveIP=192.168.0.49
netSaveMask=255.255.255.0
netSaveNameServer=208.67.222.222
netSaveDefaultRouter=192.168.0.1
```

The netSaveDev= keyword includes a special keyword AUTO-DHCP, which instructs pc-sysinstall to automatically locate and set DHCP mode on any detected network interfaces, also creating the appropriate wlan[0-9] devices for wireless nics. Should the user wish to assign their own networking configuration, the various netSave options also can specify a specific interface, IP, netmask,

nameserver and default router.

If your system requires further customization during the installation, pc-sysinstall can also help, by offering various run commands:

runCommand=cp /root/rc.custom /etc/rc.conf
runScript=/root/post-install.sh

These options are checked for and executed after the initial extraction of the installation image, and each provide different functionality. First, the runCommand option allows the execution of the specified command within a chroot environment of the system. This can be useful when you only need to make minor adjustments to the system post-install. Should you require more advanced post-install configuration, it is also possible to supply a script you wish to run in the chroot environment using the runScript command. This command will take the specified script, copy it to the installed system, run it in chroot, and remove it afterwards.

Running pc-sysinstall and using the query interface

Once you have a complete configuration file for pc-sysinstall, starting the installation process is very straight-forward. While in the pc-sysinstall directory, we simply need to run the command with the "-c" flag to specify a working configuration:

```
# ./pc-sysinstall -c /tmp/sys-install.cfg
                  ^^^^^^^^^^^^^^^^^^
                  Path to configuration file
```

The installation process will start by doing a quick syntax check of the configuration file, in order to catch any of the more egregious configuration errors, before starting to configure the disk, extract the image, and perform post-install setup. After the installation is finished, a copy of the log file will be copied to the system disk at /pc-sysinstall.log. Should the installation fail, the installer will notify the user, and provide the location of the log file in memory for immediate inspection.

In addition to this usage, pc-sysinstall can also be run with a large variety of commands to provide information to the user, or front-end interface. Commands are available for tasks such as detecting disk drives, displaying available time-zones, testing for a working network connection and more. A full list of commands may be viewed by running ./pc-sysinstall without any arguments or by viewing the doc/help-index file.

Future enhancements & goals

While the pc-sysinstall tool is already very powerful, there are still areas of improvement I would like to see worked on. First, is improving the encryption support, such as allowing pass-phrases to be specified for a partition. Currently the installer uses randomly generated keys, which are stored in /boot/keys on the installed system. Secondly I would like to work on improving the front-end interfacing, providing more common queries, and improving upon our existing ones, making it easier for more front-ends to be developed, including some which are text-based. Lastly I would like to improve the restore functionality, by expanding it to perform complete system restores from a wider variety of backup types, such as regular rsync, tar and more.

Getting pc-sysinstall and reporting bugs

pc-sysinstall is being used on PC-BSD media starting with version 8.0 and higher, and when booted may be located in the /PCBSD/pc-sysinstall directory. The code may also be accessed directly from our subversion repository under /pcbsd/trunk/pc-sysinstall or can be checked out anonymously with the following command:

```
# svn co svn://svn.pcbsd.org/pcbsd/trunk/pc-sysinstall
```

Should you run across a bug in pc-sysinstall or have ideas for improvement, the best place to reach me is either at the dev@lists.pcbsd.org mailing lists or directly at kris@pcbsd.org / kmoore@freebsd.org

Conclusion

We've only looked at a few small examples of the usage of pc-sysinstall, and some of the features it now offers. The interface is already quite mature, and able to support a variety of different installation configurations, as well as provide a complete backend for various front-ends to be developed. Through the release of PC-BSD 8.0 and future releases the interface will continue to be enhanced and made more stable, helping to make some of the latest cutting edge FreeBSD features available during install time, for novices and advanced users alike.

Illustrations

Illustration 0: SysInstaller front-end in action

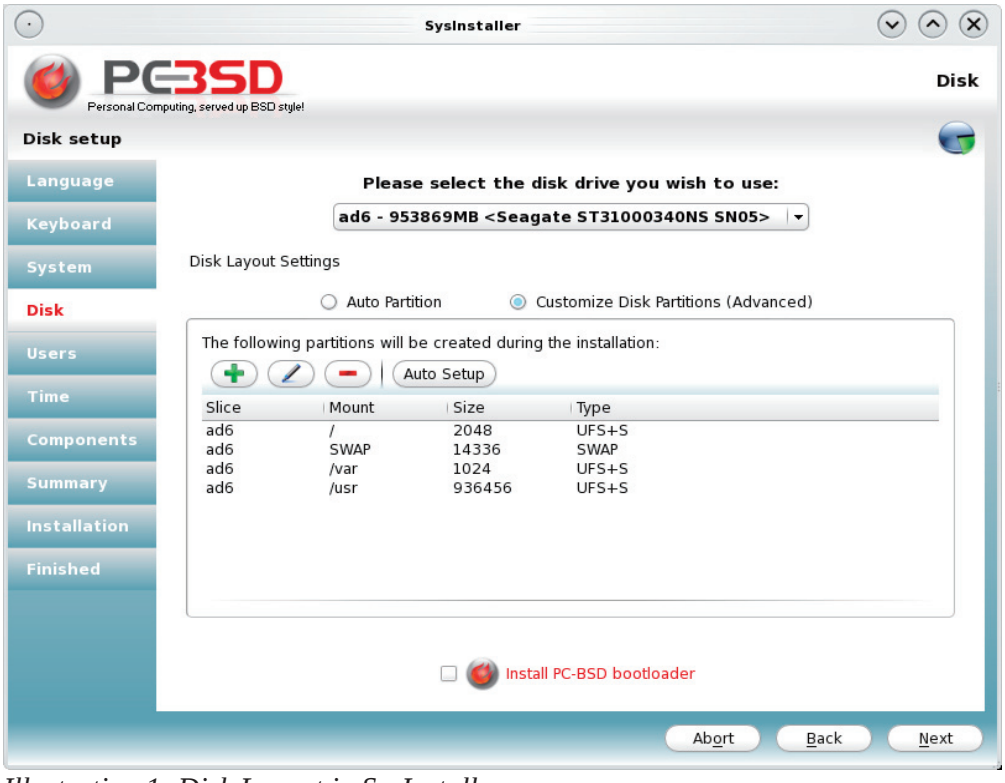

Illustration 1: Disk Layout in SysInstaller