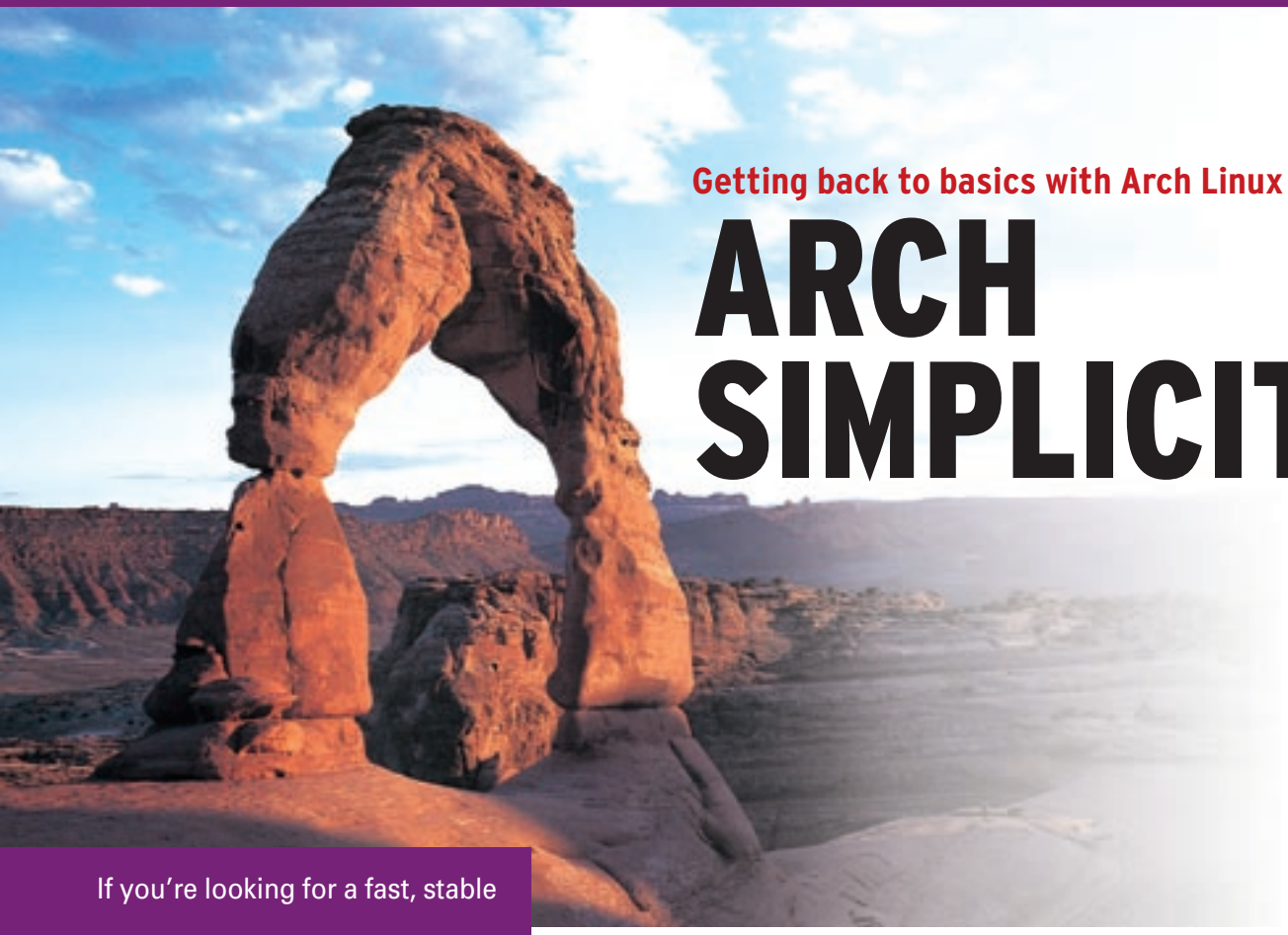

Getting back to basics with Arch Linux

ARCH SIMPLICITY

If you're looking for a fast, stable system without the GUI goo, try Arch Linux.

BY JON KENT

The recent emphasis of the Linux community has been on desktop distros that make it easy to install and configure the system without venturing beyond the GUI. Despite the success of these beginner-friendly systems, a significant segment of the Linux population prefers a simpler approach. These back-to-basics users want clarity, stability, and speed, and they do not care about the proliferation of redundant tools and glossy configuration helpers that populate the GUI-based systems. In the past, no-frills Linux users gravitated to systems such as Slackware, Gentoo, or Debian, but another back-to-basics distro is gaining favor among the Linux faithful: Arch Linux.

Arch Linux [1] was started by Judd Vinet in 2001 when he discovered that he couldn't find any other distribution that met his ideals. Arch has taken ideas from Debian, Gentoo and Slackware, and has gradually evolved into a simple, powerful, and stable distribution with an active user and developer population.

Arch provides few configuration tools and is not designed for users who are new to Linux. The Arch philosophy is to keep the user close to the underlying system. Users are expected to tinker directly with configuration files (like in the old days). But Arch also provides some advantages over other simplicity distros such as Slackware, including hotplug innovations and a more versatile package management system. Also, because Arch is specifically optimized for the i686 chip, it offers performance benefits over distros designed for a broader range of architectures. Fans of Arch say it provides "…the stability and simplicity of Slackware and the speed of Gentoo [2]." The box titled "Arch on Arch" gives an indication of how the Arch developers compare their own product with other distributions.

Although it may seem that a distribution like Arch would be more difficult for a beginning user to learn to use, the advantage of a system like Arch is that, once you learn it, you really know something about Linux. This article helps you get started with Arch Linux and describes a few of Arch's more interesting features.

Installation

Unlike most major distributions, Arch's installation program is text based. If you have installed Slackware or Debian, this text-based installation will be very famil-

Figure 1: Arch expects you to configure your system the old fashioned way: through text-based configuration files.

iar. Arch will happily install alongside any other operating systems you have installed already, although you should always back up any important data. It is worth having access to another PC to view, or print out, the well written installation documentation [4] on Arch's web site.

As with most installations, you first need to partition your disk, which is performed using cfdisk, or you can let Arch take over the entire disk if you wish. Once you have created the partitions and set the mount points, the next task is to select the packages to install. It is recommended that you install the base packages only at this stage and install any remaining packages once the system is running correctly.

Once the base system is installed, select a kernel configured for IDE or SCSI (which you'll need if you have any SATA devices), or you can even build your own kernel if you wish. However, at this stage, it is probably more sensible to select a kernel rather than build you own, as it is best to have a fully operating system before making a fundamental change. Make sure you select a udev kernel, not a devfs kernel, as Arch has now moved from devfs to udev. The last stage in the installation process is to configure the system, and this is where access to the documentation will come in handy.

Listing 1: Arch init Start Scripts

```
01 /etc/rc.sysinit - takes care
   of loading and setting up the
   system.
02 /etc/rc.single - script file
   for single user system level
03 /etc/rc.multi - script file
   for multiple users system
   level
04 /etc/rc.local - script file
   for local-multi users system
   level
05 /etc/rc.shutdown - script file
   for shutdown system level
06 /etc/rc.d/* - configured
   deamons for the system.
```

You are presented with a list of configuration files that need to be edited. There are useful comments inside these files, but it does help to have an understanding of their function. These files will be familiar to users who have configured Linux systems without the help of a GUI installer. For instance, you'll use the *rc.conf* file to configure the network, hostname, kernel modules to load, and services to start. You will need to be able to use a text editor such as vi or nano to edit these files.

The order of the services in *rc.conf* is important, as they are started in the exact order entered. There is no checking to verify any dependences, so if a service does not start up as expected, it is worth checking that the order is correct. Lastly, it is worth double checking the configuration changes you have made, and that you have made all the required modifications, as the installer will allow you to carry on even if you have not edited all the files.

When you are finished with the installation, reboot the system, and you should have a minimal Arch system. The next step is to update your system via the pacman tool (outlined later) to ensure the system is up to date before adding additional packages. If you selected udev over devfs, this upgrade will be straight forward, but if you used devfs, you'll find you need to perform additional steps to convert from devfs to udev.

System Startup

The main philosophy of Arch is to give you complete control over the configuration. As you will have seen during installation, nothing is configured automatically for you and no services are switched on unless you have configured them to be. This approach means that you come to understand the Arch system, and Linux itself, very quickly.

Arch on Arch

A page at the Arch Linux website compares Arch with other popular Linux distributions [3]. The comparison, of course, is courtesy of Arch, and the other distros may see it differently, but this summary (which is excerpted below) provides some insights on the goals and context for Arch Linux.

Arch vs. Gentoo

Gentoo has more packages. Arch allows both binary and source based distribution. PKGBUILDs are easier to create than ebuilds. Gentoo is more portable out of the box, as packages will get compiled to your specific architecture, whereas as Arch is i686 only (although i586 and x64 user-based spinoff projects are underway). There is no documented proof that Gentoo is any faster than Arch.

Arch vs. Crux

Arch Linux is descended from Crux. Judd once summarized the differences: "I used Crux before starting Arch. Arch started out as Crux, pretty much. Then I wrote pacman and makepkg to replace my bash pseudo packaging scripts. (I built Arch as an LFS system.) So the two are completely separate distros, but technically, they're almost the same. We have dependency support (officially) for example, although Crux has a community that provides other features. CLC's prt-get will do rudimentary dependency logic. Crux gets to ignore lots of problems we have too, since it's a very minimalistic package set, basically what Per uses and nothing else."

Arch vs. Slackware

Slackware and Arch are both "simple" distributions. Both use BSD-style init scripts. Arch supplies a much more robust package management system in pacman, which, unlike Slackware's standard tools, allows simple automatic system upgrades. Slackware is seen as more conservative in its release cycle, preferring proven stable packages. Arch is much more "bleeding edge" in this respect. Arch is i686 only, whereas Slackware can run on i486 systems. Arch is a very good system for Slack users who want more robust package management or more current packages.

Arch vs. Debian

Arch is simpler than Debian. Arch has fewer packages. Arch provides better support for building your own packages than Debian does. Arch is more lenient when it comes to "non-free" packages, as defined by GNU. Arch is i686 optimized. Arch packages are more bleeding edge than Debian packages.

Arch vs. Graphical Distros

The graphical distros have a lot of similarities, and Arch is very different from any of them. Arch is text based and command-line oriented. Arch is a better distro if you want to truly learn Linux. Graphical-based distros tend to ship with GUI installers (like Fedora's Anaconda) and GUI system configuration tools (like Suse's Yast).

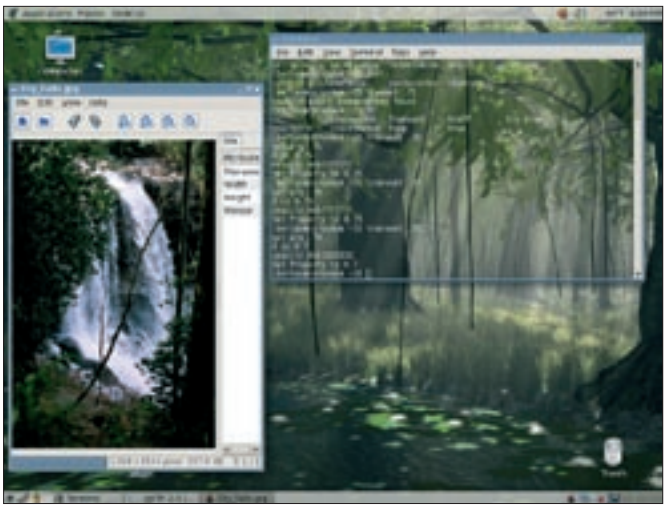

Figure 2: Although Arch emphasizes text file-based configuration, you can still use it with a desktop like Gnome or KDE.

Arch has chosen to use a BSD style init system, which is also used with Slackware. For some, this is the true init system, although others prefer the System V init approach used by a high proportion of Linux distributions. That aside, you can still use System V init scripts, which are located in */etc/rc.d*.

Binary Package Management

Arch provides the binary package management tool pacman, which is analogous to Debian's apt-get tool. pacman uses *tar.gz* files as a package format and maintains a text-based package database.

As with apt-get, pacman provides you with the ability to install and remove packages, query installation status of a package, update the package database, and so on. pacman provides an easy way to manage and install packages from either the official Arch repositories or from user repositories.

User repositories, referred to as AUR (Arch User Repositories) are a very useful feature of Arch, allowing a user to supply a package not found in the standard Arch repositories. A good example of this is the fouiny_repo repository, which provides packages for the E17 version of Enlightenment. In addition, you can create your own local repository for any packages you want to control.

Like any good package manager, pacman allows you to easily update your system. You simply synchronize the package list held on your system, and if you ask pacman to synchronize and update, pacman will then update all packages to the latest version. Installing packages is equally simple; pacman will download any dependencies that the package requires.

Pacman is configured via the configuration file */etc/pacman.conf*. Within this configuration file, you specify the repositories you wish to synchronize against. You can also specify the configuration files that are *not* to be modified by a package installation. Additionally if you wish to freeze a package at the current version, you can configure pacman to put a package on hold within this file.

```
NoUpgrade   = ➥
etc/passwd etc/group ➥
etc/shadow etc/sudoers
HoldPkg     = pacman glibc
```

In the repository section, you define which repositories to use. You can define these repositories directly or include them from another file. The latter option is useful for the official repositories, which have a lot of mirrors.

Source Package Management

Arch also provides a source management tool ABS (Arch Build System), which is almost like Gentoo's emerge. ABS is designed to package new software that is not yet available elsewhere, customize existing packages to your requirements, or even re-build the entire system using your compiler flags.

When you use ABS, it builds a package that can be installed via the pacman tool. You do not need to use ABS to have a fully functional system, but it does give you the ability to tune software packages to your liking.

ABS uses the cvsup and wget packages, so these need to be installed before ABS can be used:

```
pacman -Sy cvsup wget
```

It is always sensible to use the *-Sy* flag when installing any software to ensure that you are installing the latest version, as this option updates your package list before starting any installation.

Run the abs command to synchronize the ABS tree with the arch server using CVS, which is then mirrored into */var/abs*. The directory structure is straight forward. The base level of */var/abs* represents each category, and the next level has directories for each package. Each directory contains a *PKGBUILD* file for the package.

To install software from ABS, change to the directory of the software package in the ABS tree and then execute the

Listing 2: pacman Command Options

```
01 pacman -Sy - synchronize local packages database
02 pacman -S package_name - installing, reinstalling or upgrading a
   package
03 pacman -S extra/package_name - install package from extra repository
04 pacman -Su - upgrade all packages installed if required
05 pacman -A /<path to package>/package_name-version.pkg.tar.gz  -
   installing a local package
06
07 pacman -R package_name - remove package
08 pacman -Rs package_name - remove package and its dependencies if it
   is safe to do so
09
10 pacman -Ss package - search for a package
11 pacman -Si package - display package information
12 pacman -Scc - clear all downloaded file from pacman's cache
```

makepkg command. The *makepkg* command expects the *PKGBUILD* file to be in the local directory. Once the source code is compiled, you can then install the new software tree via pacman as follows:

```
Install new new package:
pacman -A ➥
[package name].pkg.tar.gz
Upgrade an existing package:
pacman -U ➥
[package name].pkg.tar.gz
```

In addition to these directories, there is a directory called local below */var/abs*. This is for you to create your own packages, rather than modifying the *PKGBUILD* files you have synchronized. If the current *PKGBUILD* file does not have the settings you require, create a directory under local, copy the file to this directory, add or remove settings, and then run the *makepkg* command to compile and install the software.

To control which gcc optimizations you will use, *makepkg* has a configuration file called */etc/makepkg.conf*. If you have used Gentoo, the options within this file will make sense. By default, Arch uses *-march=i686 -O2 -pipe* gcc flags, but if you prefer to live dangerously, you could change this to -O3. As Arch is firmly i686 based, it makes little sense to change the *-march* option unless you are feeling very adventurous.

hwd

By default, Arch uses the hotplug hardware detection scripts used in most Linux distributions. Hotplug simplifies the module configuration, autodetects hardware, and loads necessary modules. However Arch's developers felt that hotplug was too slow and set about developing their own approach, which they call hwd. Unlike hotplug, hwd executes modprobe in child processes, so it does not have to wait for modprobe to load each module before it can continue.

hwd is for both devfs and udev device systems. Because hwd is not a configuration utility, it does not change configuration files, as that would go against Arch philosophy. Instead hwd detects hardware and provides information how to manually configure your hardware.

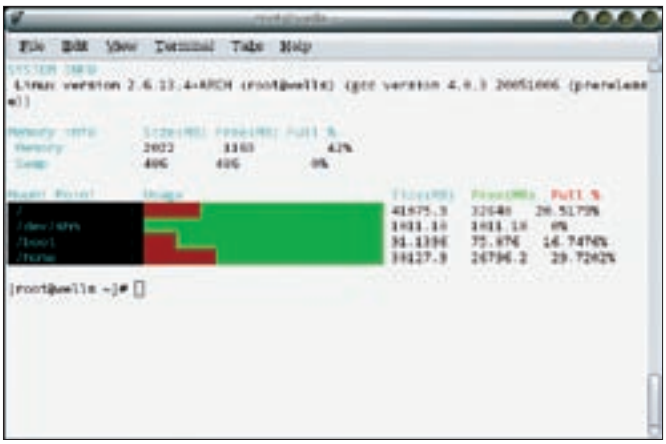

Figure 3: hwd detects hardware and displays system information.

You do not have to install hwd if you are happy with hotplug: as with all things Arch, use of hwd is completely optional. If you rarely reboot your system, or if you can live with the slight increase in boot up time, there is no reason why you cannot stay with hotplug.

Installing hwd is easy:

```
pacman -Sy hwd lshwd
```

Even though hwd is now installed, you still need to configure the hwd service to start at boot time and disable hotplug. As you'd expect, this option is controlled within the *rc.conf* file and is simply a case of adding the following line to this file:

```
!hotplug hwd
```

The *!* in front of hotplug disables the hotplug service. The next step is to download the latest pci/pcmcia tables, which hwd uses to identify the hardware, by running:

```
hwd -u
```

Now when you reboot your system, hwd will run instead of hotplug.

In addition to hardware detection, hwd and lshwd can also be used to setup an initial X configuration file or to help you set up X.

```
hwd -x
```

This command will create a sample X.org configuration file in */etc/X11*, which you can use to create your final X configuration.

Desktop

Arch uses X.org as its X-Server and has packages for all major desktops. For each desktop, you will find all of the major applications, including non-free applications such as Flash and Acrobat.

Some applications will add themselves to the desktop menu, while others will require you to manually add them. In this area there does not seem to be a standard for Arch, which can be a bit annoying. However, the speed at which GNOME or E17 run is quite impressive, and this must be due in part to the focus on i686 as the standard base for all Arch.

Conclusion

Arch Linux is a fast and lightweight distribution that takes the stripped down approach and is the better for it. Even if you have little Linux experience, you should be able to get Arch up and running, and the knowledge you gain tinkering with Arch will deepen your understanding of Linux. ■

Listing 3: /etc/pacman.conf entries

```
01 [repository-name]
02 Server = ftp://server.net/repo
03 [current]
04 # Add your preferred servers
   here, they will be used first
05 Include = /etc/pacman.d/
   current
```

INFO

[1] Arch Linux main site: Arch Linux: *http://www.archlinux.org*

[2] *http://michael-and-mary.net/intro/node/260/380*

[3] Arch vs. Others: *http://wiki.archlinux.org/index.php/Arch_vs_Others*

[4] Arch installation guide: *http://archlinux.org/docs/en/guide/install/arch-install-guide.html*

[5] Arch forums: *http://bbs.archlinux.org*

[6] Arch user contributions: *http://user-contributions.org/home/index.php*