Published on **MacDevCenter** (http://www.macdevcenter.com/)
http://www.macdevcenter.com/pub/a/mac/2004/04/09/darwinports.html
See this if you're having trouble printing code examples

Managing Packages on Panther with DarwinPorts

by Ernest E. Rothman, coauthor of Mac OS X Panther for Unix Geeks
04/09/2004

One of the first things I started doing almost immediately after installing Mac OS X when it first came out in March 2001, was to begin downloading and building Unix-based open source applications. I was able to build quite a few applications by performing the usual configure/make/make install sequence.

As experienced Unix users know, there are problems with this approach. In particular, it's difficult to maintain software installed in this manner, track dependencies, and uninstall software. Additionally, if you build applications from source, you might want to package the resulting binaries for distribution so others can install the package, or you can reinstall it at a later time without needing to rebuild it from source, or you can install it on multiple machines. Simply creating a tarball containing binaries and a readme file stating which other packages must be installed first is not an efficient package management strategy. There are more effective strategies. Quite a few package management systems are available on Unix and Linux systems. Mac OS X is no exception.

Since the release of Mac OS X, it has become quite rich in packaging options. One particularly popular option is the Debian-based Fink package management system, which was started by Christoph Pfisterer in December 2000. Pfisterer left the project in 2002, but it has been continued by many other folks. Fink is now a mature and indispensable tool for Unix geeks who use Mac OS X. You'll find good documentation of Fink on its web site and a number of books, including the recent book I coauthored with Brian Jepson, Mac OS X Panther for Unix Geeks, that provide additional coverage. We actually devoted a whole chapter to Fink in this book. In this article I will give a brief introduction to DarwinPorts, another packagement system available on Mac OS X, limiting my discussion to DarwinPorts on Mac OS X 10.3.x (Panther).

Overview of DarwinPorts

The DarwinPorts project, started in 2002 and led by Landon Fuller, Felix Kronlage, Jordan Hubbard, and Kevin Van Vechten, is a package management system, similar to Fink and the FreeBSD ports collection. It automates the installation of open source Unix- as well as Aqua-based software on Mac OS X. Written primarily in Tcl (which is bundled with Mac OS X), DarwinPorts can also be embedded in other applications.

The DP-COCOA project, led by Ernest Prabhakar, provides a Cocoa-based framework for manipulating DarwinPorts. A graphical user interface called PortsManager is also under development, according to the DarwinPorts web site. DarwinPorts, as any sophisticated package management system will do, provides a way to uninstall packages that it installs, and track dependencies of these packages. This means that if you attempt to install package A, which depends on package B, DarwinPorts will first install package B. Similarly, if you attempt to uninstall package B while you have installed another package that depends on package B, DarwinPorts will let you know and give you the option to remove other packages that depend on the one that your attempting to remove.

DarwinPorts installs Unix-based packages in */opt/local* by default so that your Mac OS X-installed system files in */usr* won't be affected. DarwinPorts also allows you to build several Aqua-based applications from source, which are installed in */Applications/Darwinports*. Additionally, various configuration files are installed in */private/etc/ports*, required libraries are installed in */Library/Tcl/darwinports1.0* (assuming Mac OSX 10.3.x), and the DarwinPorts infrastructure and descriptions of ported applications reside in a selected user's home directory, for example, */Users/ernierothman/darwinports*.

A more traditional Unix practice is to place locally installed software in */usr/local*. On some systems, for example, Solaris, */usr/local*, and */opt* are used for locally installed software and optional software, respectively. Installing software in */opt/local* takes the usual practices one step further and is regarded as a safer policy. If problems occur with DarwinPorts-installed packages, you can then delete the entire */opt/local* directory tree without affecting your system. In this case you should also delete the */private/etc/ports* and */Library/Tcl/darwinports1.0* directories.

When you build and install a package with DarwinPorts, it builds the package(s) from source in a special workspace directory called work, which you'll find within the *~/darwinports/dports/* directory. For example, if you're building an application named *greet*--categorized as a game--the application would be built in */Users/ernierothman/darwinports/dports/games/greet/work*. When you install greet, it will be installed in both the "destroot" directory */Users/ernierothman/darwinports/dports/games/greet /work/destroot/*, and in */opt/local* (or whatever you may have defined for `$prefix`) via */usr/bin/install*. A receipt is made for the installation.

As an alternative to installation via */usr/bin/install*, DarwinPorts can produce a .pkg (or .mpkg to include dependencies) package that can be subsequently installed via the Mac OS X Installer. It can also create an Internet-enabled disk image (.dmg) with a package installer. According to the DarwinPorts web site, a GUI-based Uninstaller application is under development. Design of this Uninstaller application will include the ability to uninstall Darwinports-installed packages, as well as support for the RPM package manager format.

Installing DarwinPorts

You'll find detailed documentation on the installation (and use of) DarwinPorts, written by Michael A. Maibaum, on the DarwinPorts web site. Although the DarwinPorts web site should be checked for the most up-to-date information, I'll provide a brief description of the installation and use here.

Before installing DarwinPorts, you must install the developer tools, Xcode, which ships with Mac OS X. You will also need to install X11, which is an optional installation on the Mac OS X Panther CD collection, and the X11 SDK, which is an optional installation on the Xcode CD.

Installation of DarwinPorts is built around the Concurrent Versioning System (CVS), which is installed with Xcode. A word of caution is in order before you get started. DarwinPorts and Fink can co-exist on the same system, but if you've already installed Fink (say, in its default location */sw*), there is a chance that the configure phase described below will identify the Fink-installed version of required software.

For example, if you've installed Tcl/tk with Fink, then there's a chance that DarwinPorts will use the version of Tcl in */sw*, rather than the Mac OS X bundled Tcl in */usr/bin*. If this happens and you later decide to remove Fink, you'll mess up your DarwinPorts installation. To avoid this potential problem, you may want to temporarily remove */sw/bin* from your path, (or, if you've added it to your *.bashrc* file, comment out the line `. /sw/bin/init.sh`.)

To install DarwinPorts, you should be logged in as an administrative user. In the following discussion, assume you're logged in as the administrative user ernierothman.

To download DarwinPorts and establish its infrastructure on your system, perform the following steps:

1. Change to your home directory:

   ```
   cd
   ```

2. Log into the OpenDarwin CVS server by entering the following command. When you're prompted for a password, press return:

   ```
   cvs -d :pserver:anonymous@anoncvs.opendarwin.org:/Volumes/src/cvs/od login
   ```

3. Checkout, that is, download DarwinPorts distribution files into a directory /Users/ernierothman/darwinports:

   ```
   cvs -d :pserver:anonymous@anoncvs.opendarwin.org:/Volumes/src/cvs/od co -P darwinports
   ```

The directory */Users/ernierothman/darwinports* contains the port description files (also known as portfiles), and must be kept even after you've installed DarwinPorts.

As an alternative to downloading the Darwinports files via the CVS commands given above, you can download a nightly snapshot tarball. To install DarwinPorts from the nightly snapshot, download and unpack it in your home directory. This will create the */Users/ernierothman/darwinports* directory. You can then proceed as outlined below. After you've downloaded DarwinPorts, you're ready to build and install it:

1. Change to the */Users/ernierothman/darwinports/base*, known as the DarwinPorts infrastructure.

   ```
   cd darwinports/base
   ```

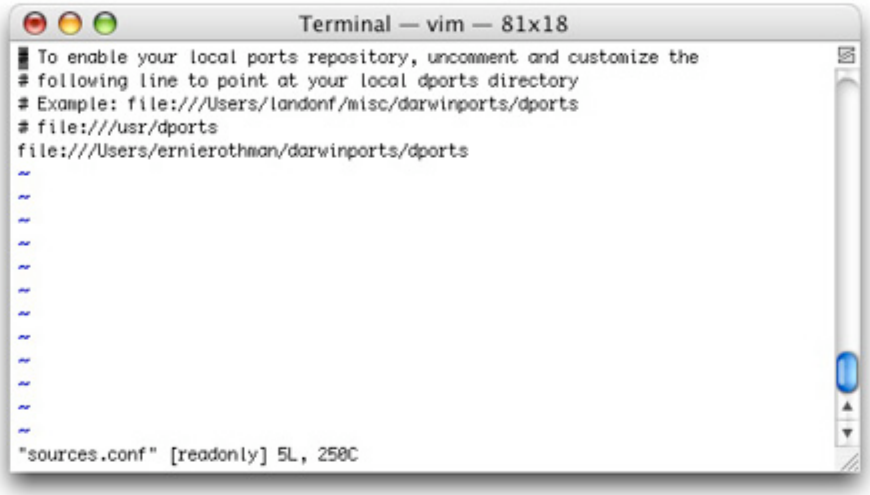
2. Perform the configure, make, make install sequence:

   ```
   ./configure
   make
   sudo make install
   ```

These commands build and install necessary files in */opt/local*, */private/etc/ports*, and */Library/Tcl/darwinports1.0*.

As part of the installation of DarwinPorts, the file */private/etc/ports/sources.conf* is created with the following line, which points to your local dports directory:

```
file:///Users/ernierothman/darwinports/dports
```

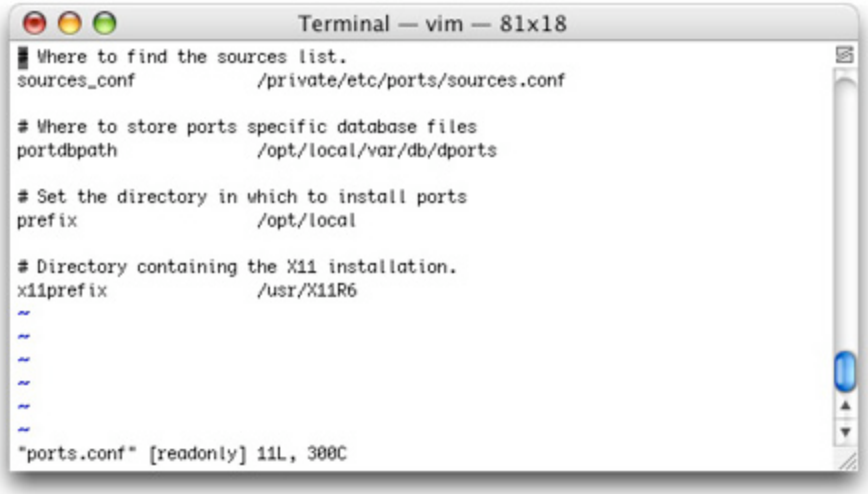


```
# To enable your local ports repository, uncomment and customize the
# following line to point at your local dports directory
# Example: file:///Users/landonf/misc/darwinports/dports
# file:///usr/dports
file:///Users/ernierothman/darwinports/dports
~
~
~
~
~
~
~
~
~
~
~
~
"sources.conf" [readonly] 5L, 250C
```

Figure 1. The sources.conf file.

The local dports directory contains the ported software descriptions and related "Portfiles," which are Tcl scripts needed to build and install each port (i.e., ported software). According to documentation on the DarwinPorts web site, the sources.conf file is used to list the locations of both the local and remote port software hierarchies, although currently there is no remote dports repository.

The installation of DarwinPorts also installs the files */private/etc/ports/ports.conf* and */private/etc/ports/prefix.mtree*.



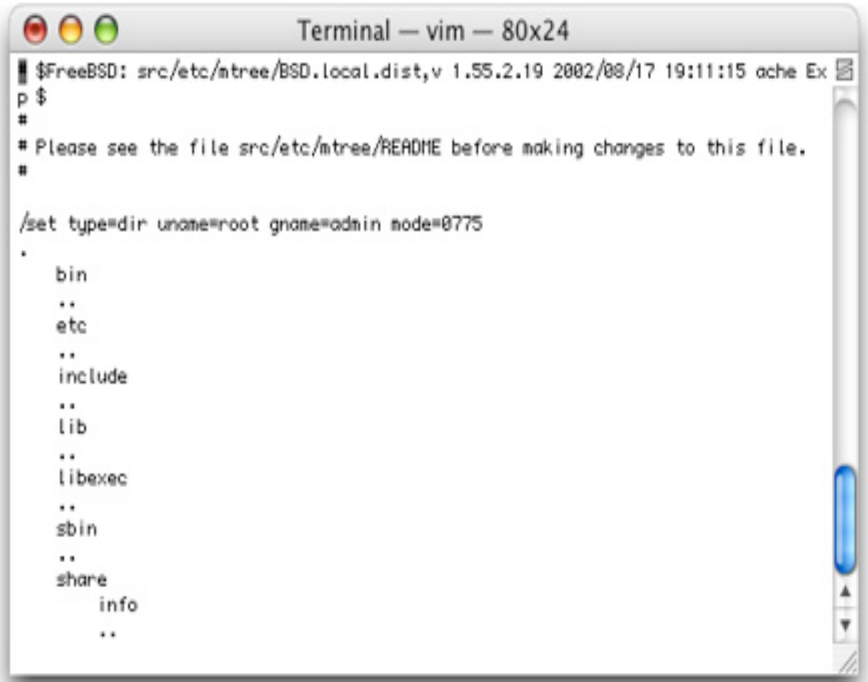
```
# Where to find the sources list.
sources_conf            /private/etc/ports/sources.conf

# Where to store ports specific database files
portdbpath              /opt/local/var/db/dports

# Set the directory in which to install ports
prefix                  /opt/local

# Directory containing the X11 installation.
x11prefix               /usr/X11R6
~
~
~
~
~
~
"ports.conf" [readonly] 11L, 300C
```

Figure 2. The ports.conf file.



```
$FreeBSD: src/etc/mtree/BSD.local.dist,v 1.55.2.19 2002/08/17 19:11:15 ache Ex
p $
#
# Please see the file src/etc/mtree/README before making changes to this file.
#

/set type=dir uname=root gname=admin mode=0775
.
    bin
    ..
    etc
    ..
    include
    ..
    lib
    ..
    libexec
    ..
    sbin
    ..
    share
        info
        ..
```

Figure 3. The prefix.mtree file.

Since DarwinPorts software is installed in */opt/local* you should add */opt/local/bin* to your path. Once you have performed these steps, you'll have a working installation of DarwinPorts. If you want DarwinPorts software to install software in a directory other than */opt/local*, you can edit the file */etc/ports/ports.conf* and change the value of prefix from /opt/local to the directory in which you want packages installed. You could alternatively, run the configure command used in the build of DarwinPort with the `--prefix` option.

Installing Packages From Source

Installation of packages with DarwinPorts is performed with the port command. Although a man page is available for port (in */opt/local/man*), I'll present a few examples of its basic use. The first thing you may want to do is to determine what software is available for installation via DarwinPorts. To view this list enter the command:

```
port list
```

or the command:

```
port search .+
```

Since the list is quite long, you may want to pipe this command through the `more` shell command. You can also use the port command to search for specific packages. For example, the command:

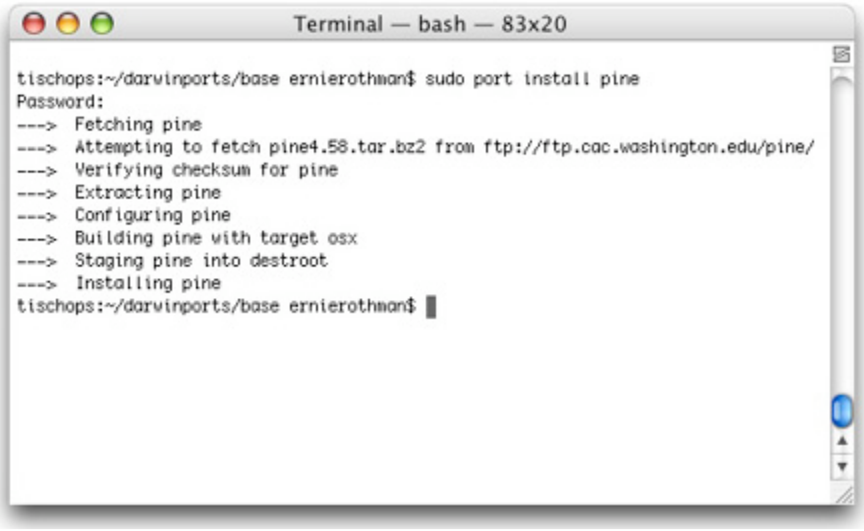
```
port search pine
```

will list the pine package, while the following command will list several packages, namely all available packages that contain the string kde.

```
port search kde
```

The port command can be used to install software. For example, to install pine, enter the following command:

```
sudo port install pine
```



```
tischops:~/darwinports/base ernierothman$ sudo port install pine
Password:
--->  Fetching pine
--->  Attempting to fetch pine4.58.tar.bz2 from ftp://ftp.cac.washington.edu/pine/
--->  Verifying checksum for pine
--->  Extracting pine
--->  Configuring pine
--->  Building pine with target osx
--->  Staging pine into destroot
--->  Installing pine
tischops:~/darwinports/base ernierothman$
```

Figure 4. Installing a port.

The installation of a software package may leave a number of intermediate files that were created during the build of the software. To delete these files, for example, in the build of pine, enter the command:

```
sudo port clean pine
```

To uninstall a particular port, use the port uninstall command. For example, to uninstall pine, enter the command:

```
sudo port uninstall  pine
```

To update a particular port it's necessary to first uninstall it, and subsequently install it.

To list your installed ports, enter the command:

```
port installed
```

Creating Packages in pkg Format

You can create a .pkg package installer using the port command with the package option, For example, to create a pkg installer for bvi enter the following command:

```
port package bvi
```

This command will download the source for bvi, build the application, and create a double-clickable package installer, *bvi-1.3.1.pkg*, in */Users/ernierothman/darwinports/dports/editors/bvi/work*. The package will not be installed in this case. To install it (in */opt/local/*), double-click *bvi-1.3.1.pkg* in the Finder, authenticate yourself as an administrative user, and install the package on your system as you would with any other .pkg package. When you install a package in this manner, the DarwinPorts database will not list it among its installed packages. That is, the 'port installed' command will not list it. If you enter the command '`port clean bvi`', the installer *bvi-1.3.1.pkg* will be deleted.

Creating Packages in rpm Format

If you are planning to create packages in rpm format, the first thing you should do is to install rpm (via the 'sudo port install rpm' command). Once you have installed rpm, you can create rpm packages using the port command with the rpmpackage option, For example, to create an rpm for pine enter the following command:

```
sudo port rpmpackage pine
```

This command will create the rpm file in *${prefix}/src/apple/RPMS/${arch}*, which in most cases will be */opt/local/src/apple/RPMS/ppc*. You can safely clean up (via '`sudo port clean pine`') after the rpm is created since the '`port clean`' command will not remove the .rpm installer. Before installing rpm packages you need to create */etc/mnttab*, which is the file that keeps track of which rpm packages are installed. This can be done in the usual way: '`touch /etc/mnttab`'.

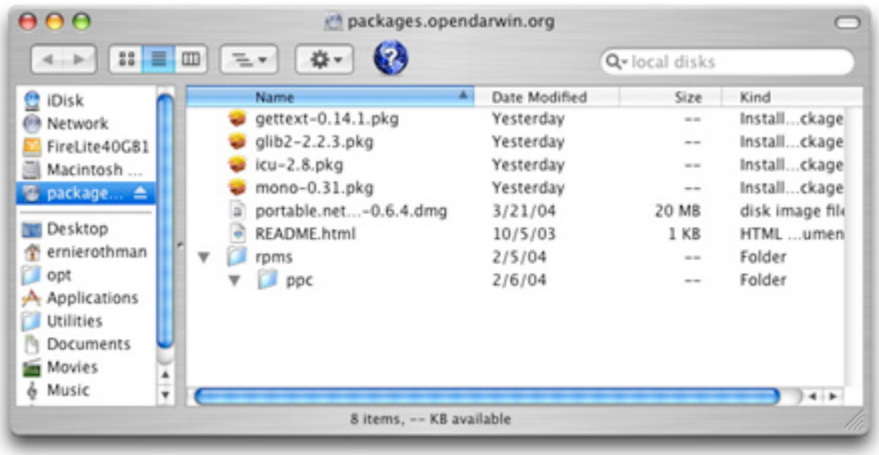
A summary of the use of the port command is given in Table 1.

Command	**Description**
`port list`	Lists available packages. Do this first.
`port install` *`foo`*	Downloads, builds, and installs package *foo*.
`port destroot` *`foo`*	Downloads, builds, and installs *foo* into an intermediate destination root, called a "destroot". This is useful in developing and testing new ports
`port uninstall` *`foo`*	Deletes package *foo*.
`port search` *`foo`*	Searches for available package *foo*.
`port installed`	Lists all the installed packages.
`port clean` *`foo`*	Cleans up intermediate files after installation of package *foo*.
`port contents` *`foo`*	Lists all files installed with installation of package *foo*.
`port deps` *`foo`*	Lists package *foo*'s dependencies.
`port variants` *`foo`*	Lists variants of package *foo*.
`port package` *`foo`*	Build .pkg package installer for *foo*. (Does not install *foo*.)
`port dmg` *`foo`*	Build an internet-enabled disk image containing an OS X .pkg package installer for *foo*. (Does not install *foo*.)
`port rpmpackage` *`foo`*	Build rpm package installer for *foo*. (Does not install *foo*.)

Table 1 - Selected port commands

Installing Binary Packages

The DarwinPorts project provides pre-built binary packages in .pkg format, that is, for use with the Mac OS X Installer application. To install one of these binary packages, you must first download its installer. Select Go -> Connect to Server from the Finder's menu bar, and enter http://packages.opendarwin.org to mount the (remote) DarwinPorts binary packages folder as a WebDAV volume on your Desktop, as shown in Figure 5.



Name	Date Modified	Size	Kind
gettext-0.14.1.pkg	Yesterday	--	Install...ckage
glib2-2.2.3.pkg	Yesterday	--	Install...ckage
icu-2.8.pkg	Yesterday	--	Install...ckage
mono-0.31.pkg	Yesterday	--	Install...ckage
portable.net...-0.6.4.dmg	3/21/04	20 MB	disk image file
README.html	10/5/03	1 KB	HTML ...umen
rpms	2/5/04	--	Folder
ppc	2/6/04	--	Folder

Figure 5: The DarwinPorts Binary Applications Volume.

Binaries may be provided as both individual packages (.pkg) and meta-packages (.mpkg). It is usually preferable to install a meta-package, if it is available, to ensure that dependencies are satisfied. These installers install Unix based packages into */opt/local*. Currently, very few packages are available from http://packages.opendarwin.org.

Updating DarwinPorts

To take advantage of the new and updated ports that are frequently added to the DarwinPorts distribution, you need to update your DarwinPorts installation. There are essentially two ways to update your DarwinPorts installation. One way is to change to the */Users /ernierothman/darwinports/dports* directory and enter the the command:

```
cvs -z3 update -dP
```

After this command has completed its work, you should enter the the command:

```
portindex
```

to rebuild the index of available ports.

The other way to update your DarwinPorts installation is to reinstall the DarwinPorts infrastructure in /Users/ernierothman/darwinports. To accomplish this, change to the /Users/ernierothman/darwinports/base directory and enter the following commands:

```
cvs -z3 update -dP
./configure
make clean && make
sudo make install
```

Creating DarwinPorts Packages

You can create your own DarwinPorts packages (i.e., ports) by identifying a source archive and creating a Portfile file in the appropriate subdirectory of the dports directory. For example, the Portfile for foo would be placed in */Users/ernierothman/darwinports/dports /games/foo*.

To illustrate this process suppose you want to create a port named greet for the following program, which prints a greeting.

```
/*
*   greet.c  - prints a friendly greeting
*/
#include <stdio.h>
int main()
{
  printf("Greetings, Earthlings!\n");
  return 0;
}
```

Creating and Publishing the Tarball

The DarwinPorts package system needs a tarball that can be downloaded with the curl utility, so you should put the required files into a directory, such as *greet-1.0*. In this simple example, the directory greet-1.0 will contain *greet.c*, a man page *greet.6*, and the following makefile.

```
# Makefile for greet
all:
        cc -o greet greet.c
```

Create a tarball *greet-1.0.tar.gz* containing these files and the top-level directory greet-1.0, obtain the md5 checksum of *greet-1.0.tar.gz* (you'll need this later), and put it somewhere where you can get it. In this example, you can move *greet-1.0.tar.gz* to the local */Users/Shared/greet/src* directory. The curl utility can download this file with the following URL: file:///Users/Shared/greet/src/greet-1.0.tar.gz. (You could alternatively put *greet-1.0.tar.gz* on a public web server or FTP server.)

Creating the Portfile

Once the tarball has been placed in file:///Users/Shared/greet/src/, you need to create a file named Portfile in */Users/ernierothman/darwinports/dports/games/greet*. A portfile lists the attributes of the package needed by DarwinPorts, for example, name, version, maintainer(s), where to download the package and how to install it. DarwinPorts uses this information to download, extract, and compile the source code. Information on patchfiles, special configure or compilation flags, and/or installation or post installation configuration instructions could be included in a portfile.

The following portfile can be used in the 'greet' example.

```
# $Id: $
PortSystem 1.0
name            greet
version         1.0
categories      games
maintainers     myemail@mac.com
description     "Twist on hello program"
long_description "Twist on hello program.  \
                                Prints: Greetings,\
                  Earthlings."

master_sites    file:///Users/Shared/greet/src
homepage                file:///Users/Shared/greet
distname        ${portname}-${portversion}
platforms       darwin
checksums       md5 732bf0585410cbebe6de7f57c8de88d8
configure {}

set instprog    "/usr/bin/install -m 755"
set instman     "/usr/bin/install -m 644"
```

```
destroot        {
    system "${instprog} -d ${destroot}${prefix}/bin"
        system "${instprog} -d ${destroot}${prefix}/man/man6"
        system "${instprog}  \
          ${worksrcpath}/greet ${destroot}${prefix}/bin"
        system "${instprog}  \
          ${worksrcpath}/*.6 ${destroot}${prefix}/man/man6" }
```

This Portfile file includes several entries, described in the following list.

`# $Id: $`
: All Portfiles begin with this string, which is a commented out RCS Id tag.

`Portsystem 1.0`
: This second line is the Portsystem version declaration.

`name`
: The package name.

`version`
: The package version.

`categories`
: Used for organization of packages into categories, e.g., mail, editors, games, etc.

`maintainers`
: Email addresses of folks maintaining the port.

`description`
: One line description of the package.

`long_description`
: More detailed description of the package.

`master_sites`
: The URL of the source distribution.

`homepage`
: The URL of the software's web site.

`distname`
: Name of the distribution, e.g., greet-1.0. (Note: portname gets the value of name and portversion gets the value of version.)

`platform`
: Specifies the platform on which the port is to be built.

`checksums`

This command will verify the md5 checksum. (Required.)

`configure{}`

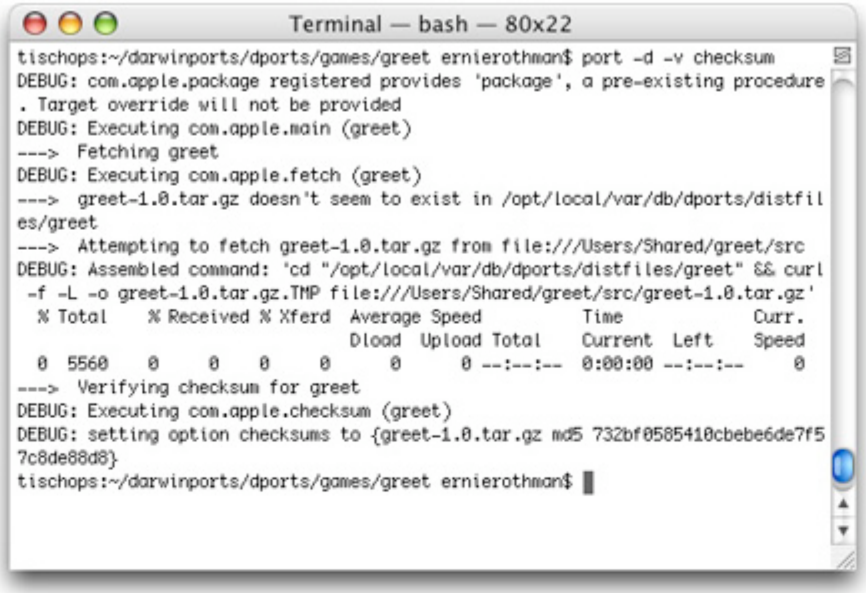
The brackets are left empty if there is no autoconf configure script to run, as in this simple example. If there is a configure script, the argument `-prefix=${prefix}` is passed to it by DarwinPorts. After the `configure{}` line in the above Portfile, there are installation instructions to ensure that the program and its man page get installed into the correct directory.

The variables `instprog` and `instman`, are used to specify exactly which commands are to be used to install the binary and man page, respectively. The `destroot` key is included to specify exactly what the system should do when the `destroot` option is used with the `port` command.

For additional details on portfiles, see Michael A. Maibaum's DarwinPorts User Guide, a sample portfile in */Users/ernierothman /darwinports/base/doc/exampleport*, and the portfile man page for more details.

Building and Installing Port

Once the Portfile file is ready, you can build the port. This involves a sequence of port commands, each invoked with the `-v` (verbose) and `-d` (debug) options. Begin this process by changing to the the directory which contains Portfile and enter the following command to verify the md5 checksum of the tarball: `port -d -v checksum`



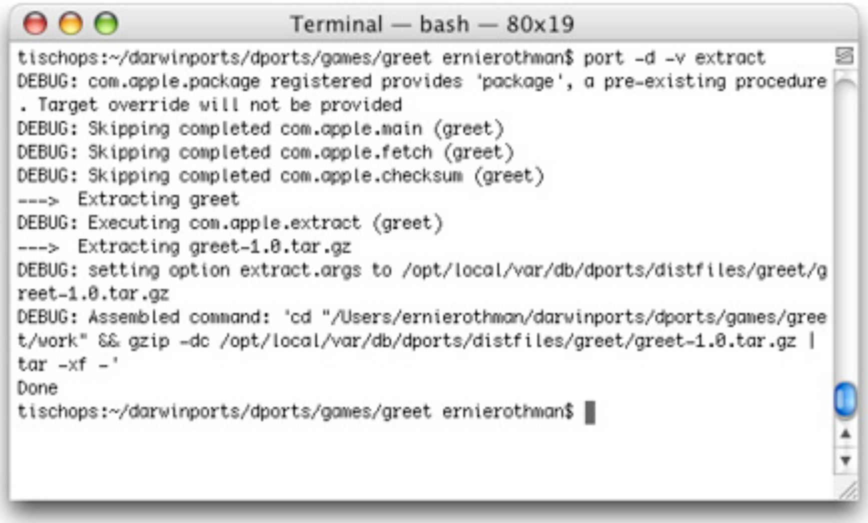
```
tischops:~/darwinports/dports/games/greet ernierothman$ port -d -v checksum
DEBUG: com.apple.package registered provides 'package', a pre-existing procedure
. Target override will not be provided
DEBUG: Executing com.apple.main (greet)
--->  Fetching greet
DEBUG: Executing com.apple.fetch (greet)
--->  greet-1.0.tar.gz doesn't seem to exist in /opt/local/var/db/dports/distfil
es/greet
--->  Attempting to fetch greet-1.0.tar.gz from file:///Users/Shared/greet/src
DEBUG: Assembled command: 'cd "/opt/local/var/db/dports/distfiles/greet" && curl
 -f -L -o greet-1.0.tar.gz.TMP file:///Users/Shared/greet/src/greet-1.0.tar.gz'
  % Total    % Received % Xferd  Average Speed          Time             Curr.
                                 Dload  Upload Total    Current  Left    Speed
  0  5560    0     0    0     0      0      0 --:--:--  0:00:00 --:--:--     0
--->  Verifying checksum for greet
DEBUG: Executing com.apple.checksum (greet)
DEBUG: setting option checksums to {greet-1.0.tar.gz md5 732bf0585410cbebe6de7f5
7c8de88d8}
tischops:~/darwinports/dports/games/greet ernierothman$ 
```

Figure 6. The required port checksum command.

Since no explicit port name was provided in the preceding command, DarwinPorts obtains from any Portfile in the current directory the information that is needed to download and verify the md5 checksum of the source file. The source tarball file *greet-1.0.tar.gz* is downloaded into */opt/local/var/db/dports/distfiles/greet*, and a work directory is created in */Users/ernierothman/darwinports/dports/games /greet/*.

Next, extract the source by entering the following command

```
port -d -v extract
```



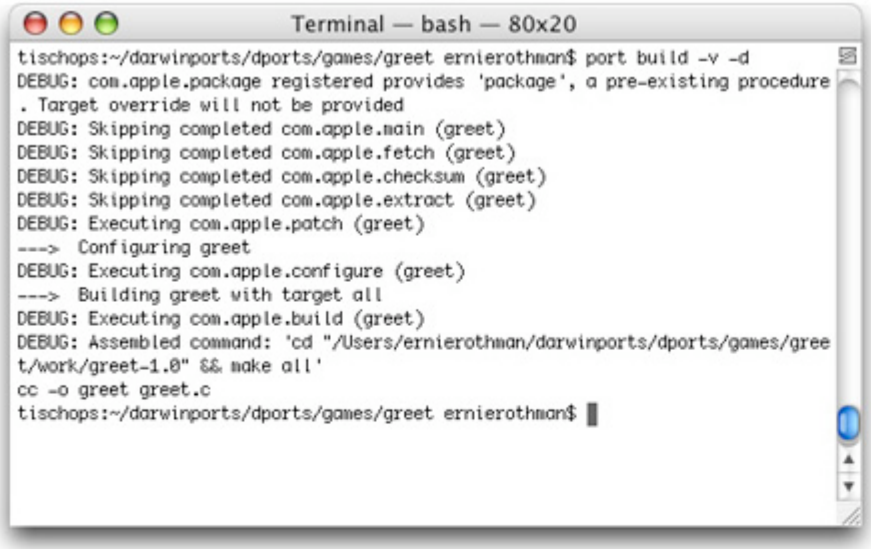
```
tischops:~/darwinports/dports/games/greet ernierothman$ port -d -v extract
DEBUG: com.apple.package registered provides 'package', a pre-existing procedure
. Target override will not be provided
DEBUG: Skipping completed com.apple.main (greet)
DEBUG: Skipping completed com.apple.fetch (greet)
DEBUG: Skipping completed com.apple.checksum (greet)
--->  Extracting greet
DEBUG: Executing com.apple.extract (greet)
--->  Extracting greet-1.0.tar.gz
DEBUG: setting option extract.args to /opt/local/var/db/dports/distfiles/greet/g
reet-1.0.tar.gz
DEBUG: Assembled command: 'cd "/Users/ernierothman/darwinports/dports/games/gree
t/work" && gzip -dc /opt/local/var/db/dports/distfiles/greet/greet-1.0.tar.gz |
tar -xf -'
Done
tischops:~/darwinports/dports/games/greet ernierothman$ 
```

Figure 7. Extracting the source code.

This command unpacks *greet-1.0.tar.gz*, creating the */Users/ernierothman/darwinports/dports/games/greet/work/greet-1.0* directory.

Once the source code has been unpacked, you can build the package with the following command:

```
port -d -v build
```



```
tischops:~/darwinports/dports/games/greet ernierothman$ port build -v -d
DEBUG: com.apple.package registered provides 'package', a pre-existing procedure
. Target override will not be provided
DEBUG: Skipping completed com.apple.main (greet)
DEBUG: Skipping completed com.apple.fetch (greet)
DEBUG: Skipping completed com.apple.checksum (greet)
DEBUG: Skipping completed com.apple.extract (greet)
DEBUG: Executing com.apple.patch (greet)
--->  Configuring greet
DEBUG: Executing com.apple.configure (greet)
--->  Building greet with target all
DEBUG: Executing com.apple.build (greet)
DEBUG: Assembled command: 'cd "/Users/ernierothman/darwinports/dports/games/gree
t/work/greet-1.0" && make all'
cc -o greet greet.c
tischops:~/darwinports/dports/games/greet ernierothman$ 
```

Figure 8. Building the application.

If the build goes well, you can test the installation by first installing the port in the destroot directory:

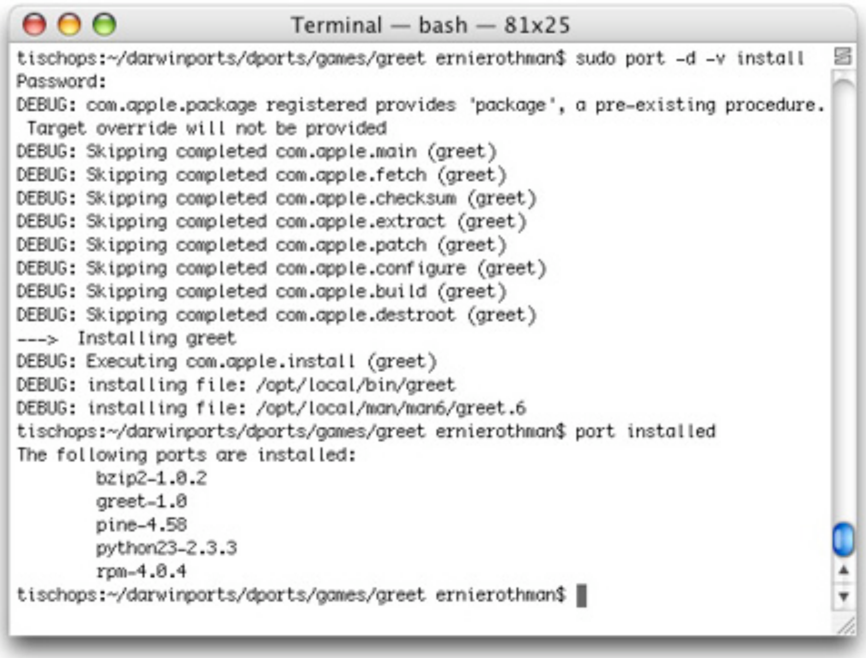
```
port -d -v destroot
```

This produces a large number of warning messages, but in the end, if all goes well, you will have installed both the binary greet and the man page greet.6 in the /Users/ernierothman/darwinports/dports/games/greet/work/destroot/opt/local directory. After you've tested the binary and man page in this destroot directory, you can install the greet port system-wide, that is, in /opt/local

To do this enter the following command:

```
sudo port -d -v install
```

You can check that greet has actually been installed properly by entering the 'port installed' command, and by trying to run greet and viewing its man page.



```
tischops:~/darwinports/dports/games/greet ernierothman$ sudo port -d -v install
Password:
DEBUG: com.apple.package registered provides 'package', a pre-existing procedure.
 Target override will not be provided
DEBUG: Skipping completed com.apple.main (greet)
DEBUG: Skipping completed com.apple.fetch (greet)
DEBUG: Skipping completed com.apple.checksum (greet)
DEBUG: Skipping completed com.apple.extract (greet)
DEBUG: Skipping completed com.apple.patch (greet)
DEBUG: Skipping completed com.apple.configure (greet)
DEBUG: Skipping completed com.apple.build (greet)
DEBUG: Skipping completed com.apple.destroot (greet)
--->  Installing greet
DEBUG: Executing com.apple.install (greet)
DEBUG: installing file: /opt/local/bin/greet
DEBUG: installing file: /opt/local/man/man6/greet.6
tischops:~/darwinports/dports/games/greet ernierothman$ port installed
The following ports are installed:
        bzip2-1.0.2
        greet-1.0
        pine-4.58
        python23-2.3.3
        rpm-4.0.4
tischops:~/darwinports/dports/games/greet ernierothman$
```

Figure 9. Installing greet system-wide and verifying its installation.

Finally, you should enter the portindex command from */Users/ernierothman/darwinport/dport* so that your DarwinPorts installation is completely aware of the newly installed greet port. Once you've done this, you can uninstall greet as you would uninstall any other port. That is, you could uninstall greet with the following command:

```
sudo port uninstall greet
```

This example shows only a small portion of DarwinPorts' capabilities. For more information, see the sources noted earlier, which contain detailed instructions on how to build and contribute a package to the DarwinPorts distribution.

Final Thoughts

DarwinPorts, along with Fink, provide Mac OS X users with the means to easily install and maintain an enormous number of open source applications. Fink has many more ported applications and is more mature than DarwinPorts. On the other hand, you may find some specific applications missing in Fink, but included in DarwinPorts. Fortunately, Fink and DarwinPorts can co-exist on the same system. Both projects seem to be under active development and improvement. Moreover, there is increasing cooperation among the Fink, DarwinPorts, and Gentoo Linux projects via the Metapkg Alliance. These efforts have made thousands of open source packages available to Mac OS X users, and have made package management on Mac OS X quite painless.

Ernest E. Rothman is a Professor of Mathematics at Salve Regina University (SRU) in Newport, Rhode Island and coauthor of Mac OS X Tiger For Unix Geeks.

Return to the Mac DevCenter