

---

# PRT-GET - USER MANUAL

*An advanced package management tool for CRUX*

## Author

Johannes Winkelmann, [jw@tk6.net](mailto:jw@tk6.net)

## Table of Contents

1. Introduction .....	2
1.1. Description .....	2
1.2. Recommendation .....	2
1.3. General usage .....	3
1.4. Wildcards in prt-get .....	3
1.5. Remember the test mode .....	3
2. Configuration .....	3
2.1. Some words about prt-get's internals .....	3
2.2. Configuration file .....	4
2.3. The prtdir Option .....	4
2.4. The cachefile Option .....	4
2.5. The logging Options .....	4
2.6. The readme Option .....	5
2.7. The depfile Option .....	6
2.8. Execution of install scripts .....	6
2.9. Order of directories .....	6
2.10. Listing a directory multiple times .....	6
2.11. Example prt-get.conf file .....	6
2.12. Changing the configuration settings via the command line .....	7
2.13. Checking your configuration values .....	7
3. Installation, Removal and Update of Packages .....	8
3.1. Install .....	8
3.2. Update .....	9
3.3. Group install .....	9
3.4. Install with dependencies .....	10
3.5. Removing packages .....	10
3.6. Update all outdated packages .....	10
3.7. Locking packages .....	11
3.8. Pre- and Post-Install scripts .....	11
4. Searching and getting information about ports .....	11
4.1. List available ports .....	11
4.2. Formatted list of available ports .....	12
4.3. List installed port .....	13
4.4. Searching the port names .....	13
4.5. Description Search .....	14
4.6. Searching the files of the ports .....	14
4.7. Information .....	15
4.8. Printing the path .....	15
4.9. Printing the README file of a port .....	15
5. Dependencies .....	16
5.1. Dependencies as supported by prt-get .....	16
5.2. Listing dependencies .....	16
5.3. Listing dependent packages .....	17
5.4. Tree representation of dependencies .....	17
6. Package status information .....	18
6.1. Differences between installation and the ports tree .....	18
6.2. Check if a package is installed .....	18
6.3. Check the version of an installed package .....	18

6.4. List duplicate packages in ports tree .....	18
7. Using a log file .....	19
7.1. Description .....	19
8. Using a cache file .....	19
8.1. Using a cache file .....	19
8.2. Creating a cache file .....	19
8.3. Creating a cache file .....	19
8.4. Using the cache explicitly: prt-cache .....	19
9. Advanced usage .....	20
9.1. Installation using dependencies .....	20
9.2. Update all outdated ports .....	20
9.3. Package clusters .....	20
9.4. Accessing files of a port .....	20
9.5. Sharing a Ports Tree .....	20
9.6. All your base... ..	21
9.7. Recompile all installed ports .....	21
10. Copyright and licensing .....	21
10.1. Copyright .....	21
10.2. License .....	21

# 1. Introduction

## 1.1. Description

**prt-get** is a package management tool for CRUX<sup>[1]</sup> which provides additional functionality to crux' package management system. It works with the local ports tree and is therefore fully compatible with **ports(8)** and **pkgmk(8)/pkgadd(8)**. It offers the following features:

- abstract port installation/update from file system
- install/update a list of packages with one command
- list dependencies for a list of packages
- show information about ports
- search within the ports
- advanced logging for builds

What **prt-get** basically does is installing and upgrading packages, using **pkgmk** and **pkgadd**. Additionally, you don't have to be in the port's directory to call **prt-get**. **prt-get** will search for the respective port itself in a list of directories specified in `/etc/prt-get.conf`. This allows you to just install or update a package, without caring where it actually is located on your file system.

**prt-get** also offers some features like searching ports by name, showing information about ports (without installing them of course) and can list the dependencies listed in the `Pkgfile`, and provide a complete dependency list for a port. Note that dependencies are no requirement for crux packages and therefore not always accurate

**prt-get** has a test mode so you can see what effect an install/update operation would have. Use the `--test` switch for this (see Section 1.5, "Remember the test mode" [3] to learn more about the test mode).

## 1.2. Recommendation

---

<sup>[1]</sup> CRUX is a lightweight Linux Distribution, check <http://crux.nu> for more information

**prt-get** is a front end to **pkgmk** and **pkgadd** and requires a basic knowledge how these tools work. Make sure you understand the package management of CRUX and the tools used *before* you start using **prt-get**.

## 1.3. General usage

The general usage for **prt-get** is similar to **cvs**: the first argument passed is always a so called command. In general the syntax looks like this:

```
prt-get command arguments
```

## 1.4. Wildcards in prt-get

Some commands support wildcard patterns as arguments; in general, those which produce a listing have this option.

- list, e.g. **list kde\***
- listinst, e.g. **listinst ???**
- diff, e.g. **diff a\***
- printf, e.g. **printf "%n\n" --filter=gnome\***

The shell tries to expand the wildcards, therefore you might run into problems sometimes: if you have files in your current working directory matching the pattern, the shell will expand it and **prt-get** will receive the expanded filename as argument instead of the pattern. In this case, you can simply escape the command (e.g. **list kde\\***) or pass in surrounded by quotation marks (**list "kde"**).

## 1.5. Remember the test mode

**prt-get** has a test mode where no actual installation or update happens, but only the respective output shown. It can be used by adding the **--test** switch to a command, e.g.

```
prt-get install gimp --test
```

You might want to try out the commands you see here, and you can use the test mode to make sure you don't do anything before you're perfectly sure what a command means.

# 2. Configuration

## 2.1. Some words about prt-get's internals

One of **prt-get**'s targets is to abstract the ports from the file system. This means that a user doesn't have to care about the actual directory where a port is located and she/he can just install a package (e.g. **mutt**) instead of a port (e.g. `/usr/ports/contrib/mutt`). The key used to identify is only the name of the port (Sidenote: **prt-get** uses the name of the port's directory as name for the port, not the name variable from the `Pkgfile`<sup>[2]</sup>. They should always be the same.)

Still the main idea behind this abstraction is not comfort, but simplicity for dependency specification. Having any information about the file system in `Pkgfiles` (even if it's only

---

<sup>[2]</sup> If you don't know what `Pkgfiles` are, please refer to the CRUX user manual

"base", "opt", "clc/stable" or "clc/unstable") is something to avoid by any means.

This concept requires the user to specify the base directories where prt-get looks for ports. This can be done in the configuration file `/etc/prt-get.conf`.

## 2.2. Configuration file

The configuration file `/etc/prt-get.conf` has a very simple format. You specify one option per line. All options are key-value pairs, separated by whitespace.

## 2.3. The prtdir Option

Specify here where to look for ports. These directories are not the port directories itself, but the directory where prt-get looks for ports; that means you don't specify `/usr/ports/contrib/mutt` (a port directory), but `/usr/ports/contrib`. **prt-get** will then search for all ports in this directory. Recursive directories are not supported.

### 2.3.1. Example

```
prtdir /path/to/ports
```

The more, the user can restrict which ports should be used per directory by providing a list of those after the directories path. The allowed packages have to be separated by a comma character (","), the list itself must be separated with a colon char (":") from the path. The format of a line therefore looks like this:

```
prtdir /path/to/ports:port1, port2, ...
```

## 2.4. The cachefile Option

By default, the cache file is `/var/lib/pkg/prt-get.conf`. You can change this by using `cachefile /path/file`. This is especially nice if you want to place it on an NFS share without exporting the complete `/var/lib/pkg` tree.

## 2.5. The logging Options

prt-get has a mechanism to log the output of builds into a file. There are three options in the config file:

- `writelog (enabled|disabled):` enable logging <sup>[3]</sup>
- `logmode (append|overwrite):` always append to the logfile
- `logfile filename:` write log to *filename*. *filename* can contain the `%n` character, which is replaced by the port name, and the `%p` character, which is replaced with the path to the port's *directory*.

Example: one log file for each port

```
logmode overwrite  
logfile /tmp/log/%n.log
```

or: one directory per port, same file name for all:

```
logmode overwrite
```

```
logfile /tmp/log/%n/build.log
```

or: one log file for everything.

```
logmode append  
logfile /tmp/log/build.log
```

or: log file in port directory:

```
logmode append  
logfile %p/build.log
```

## 2.6. The readme Option

### 2.6.1. Idea

From version 0.3.5, **prt-get** checks for a file called *README* in the port's directory. Those files usually contain important information and it is highly recommended to read them. **prt-get** has a command to print out readme files which is called **prt-get readme** (who would have guessed). The information is also shown when you issue **prt-get info**.

Now often when you install software you don't want to check for a readme file, that's why **prt-get** offers a way to automatically display this information for install and update actions. There are two ways of displaying this information: either integrated in the result of an install/update operation.

```
$ prt-get update gtk2  
...  
-- Packages updated  
gtk2 (README)  
  
prt-get: updated successfully
```

or appended as a separate block:

```
$ prt-get update gtk2  
...  
-- Packages updated  
gtk2  
  
-- updated packages with README files:  
gtk2  
  
prt-get: updated successfully
```

### 2.6.2. Configuration in prt-get.conf

To configure this, there is a configuration option called `readme` which can be set to `compact` (integrated in the result), `verbose` (separate block) or `disabled` (don't show information about readme files).

### 2.6.3. Example

compact readme information

```
###  
# prt-get.conf  
...  
readme compact
```

## 2.7. The depfile Option

Starting with prt-get 0.5.11, the depfile option is not used anymore; a dependency list is shipped by default.

## 2.8. Execution of install scripts

If you want to execute pre- and post-install scripts automatically, you can set the **runscripts** variable to 'yes'.

### 2.8.1. Example

```
###
# prt-get conf
runscripts yes
```

## 2.9. Order of directories

Note that the order of the directories (prtdir) in `prt-get.conf` is very important. As **prt-get** doesn't care about the directory where a port is, it can distinguish `/usr/ports/clc/unstable/yafc` and `/home/build/jw/yafc`. In such a situation, **prt-get** uses the port it finds first. So if you want a port to have precedence over another, make sure that its base directory appears earlier in `prt-get.conf`. You can use the **prt-get dup** command to see which ports are multiple times in the ports tree, and which one has precedence.

## 2.10. Listing a directory multiple times

A directory can appear twice in `prt-get.conf`. Usually this means that it's just ignored the second time as there are already ports of this name (see above). Still it can be interesting, if you want to let a single port having precedence over all the others:

### 2.10.1. Example

```
###
# prt-get conf

# use bash from /home/jw/build
prtdir /home/jw/build:bash

prtdir /usr/ports/base
prtdir /usr/ports/opt
prtdir /usr/ports/clc/stable
prtdir /usr/ports/clc/unstable

# use the rest from /home/jw/build
prtdir /home/jw/build
```

## 2.11. Example prt-get.conf file

```
###
# prt-get.conf

prtdir /usr/ports/base
prtdir /usr/ports/opt
prtdir /usr/ports/clc/stable
```

```
prtdir /usr/ports/clc/unstable

# use ONLY prt-get and yafc from /home/jw/build
prtdir /home/jw/build:prt-get, yafc

# use alternate cache file
cachefile /tmp/cache

# log
logmode overwrite
logfile /tmp/log/%n.log

# show readme files in compact mode
readme compact

# use an external dependency listing
depfile /var/lib/pkg/opt.dependencies

runscripts yes
```

## 2.12. Changing the configuration settings via the command line

Sometimes it's handy to add or override configuration settings for a single installation. This is possible using the following four command line options:

- `--config-set="single config line"`
- `--config-append="single config line"`
- `--config-prepend="single config line"`
- `--no-std-config`

The `append` and `prepend` options just act like you'd append or prepend the text to the configuration file. The **config-set** option tries to replace the key included there. This is mostly useful for the **prtdir** setting. The **--no-std-config** option finally makes `prt-get` simply ignore `/etc/prt-get.conf`.

```
$ prt-get update myport --config-append="readme compact" # 1
$ prt-get update myport --config-prepend="prtdir /tmp/tempports" # 2
$ prt-get update myport --no-std-config --config-set="prtdir /mnt/ports"# 3
```

- 1) set `readme` to `compact`
- 2) prefer ports from `/tmp/tempports`
- 3) only look at `/mnt/ports`; ignore `/etc/prt-get.conf`

## 2.13. Checking your configuration values

You can check your current settings using the **dumpconfig** command:

```
$ prt-get dumpconfig
Configuration file: /etc/prt-get.conf
Dependency file:   /var/lib/pkg/external-pkg-dependencies
Log file:         /tmp/log/%n.log
```

```
Write log:      no
Append log:    no

Port directories:
/usr/ports/base
/usr/ports/opt
/usr/ports/jw
/usr/ports/contrib
/usr/ports/unmaintained
```

## 3. Installation, Removal and Update of Packages

### 3.1. Install

To install a software package from the ports tree, use **prt-get install**. You can call it from any directory, so you don't have to change to the port's directory first. If you are not sure about the package name, you can use the search function of **prt-get** (see Section 4.4, "Searching the port names" [13]).

You can install more than one package with the same command. If installing of one package fails, **prt-get** will continue with the next package. (If you don't want this, use **prt-get gr-pinst**, which stops when building of a package fails)

```
prt-get install package [package2 ...]
```

#### 3.1.1. Arguments

A list of arbitrary length of packages to be installed (at least one). The order is relevant, packages are installed in the order they are passed to **prt-get**. Calls **pkgmk** and **pkgadd**. **prt-get install** does all the downloading, building and installing for you.

#### 3.1.2. Options

- **--margs=...:** arguments to be passed to **pkgmk**
- **--args=...:** arguments to be passed to **pkgadd**
- **--pre-install:** execute pre install scripts if available
- **--post-install:** execute post install scripts if available
- **--install-scripts:** execute pre and post install scripts if available
- **--cache:** use cache
- **--log:** write a log file
- **--ignore=<package1,package2>:** ignore those packages

#### 3.1.3. Examples

- **prt-get install mutt**

- **prt-get install procmail fetchmail**
- **prt-get install --margs='-im -f' --aargs='-f' exim**

## 3.2. Update

To update a package which is already installed, use **prt-get update**. You can for example update packages listed in **ports -d** or **prt-get diff**.

```
prt-get update package [package2 ...]
```

### 3.2.1. Arguments

A list of arbitrary length of packages to be updated (at least one). The order is relevant, packages are installed in the order they are passed to **prt-get**. Calls **pkgmk** and **pkgadd**. **prt-get update** does all the downloading, building and installing for you.

### 3.2.2. Options

- **--margs=...:** arguments to be passed to **pkgmk**
- **--aargs=...:** arguments to be passed to **pkgadd**
- **--cache:** use cache
- **--pre-install:** execute pre install scripts if available
- **--post-install:** execute post install scripts if available
- **--install-scripts:** execute pre and post install scripts if available
- **--cache:** use cache
- **--log:** write a log file

### 3.2.3. Examples

- **prt-get update openssh**
- **prt-get update procmail fetchmail**

## 3.3. Group install

When you install a list of packages, it is sometimes desirable to stop after installation of a package fails. This is mostly the case if packages depend on the ones installed before. For example, the port of **distcc** requires **popt** to be installed in order to compile. If installing **popt** fails, it makes no sense in trying to install **distcc**.

```
prt-get grpinst package [package2 ...]
```

### 3.3.1. Options

- **--margs=...**: arguments to be passed to pkgmk
- **--aargs=...**: arguments to be passed to pkgadd
- **--pre-install**: execute pre install scripts if available
- **--post-install**: execute post install scripts if available
- **--install-scripts**: execute pre and post install scripts if available
- **--cache**: use cache
- **--cache**: use cache
- **--log**: write a log file

### 3.3.2. Example

- `prt-get grpinst popt distcc`

## 3.4. Install with dependencies

To install one package or more packages and all packages that are listed as dependencies use the **depinst** command. Note that some dependency listings are not complete, so a failure here is most certainly not a bug in prt-get but in the packages.

Besides handling dependencies **prt-get depinst** behaves the same as **prt-get grpinst**

```
prt-get depinst package [package2 ...]
```

## 3.5. Removing packages

To remove an arbitrary number of packages use the **remove** command. This will remove the packages using **pkgrm**. It is possible to remove a list of packages with one command.

```
prt-get remove package [package2 ...]
```

## 3.6. Update all outdated packages

To update all outdated packages automatically, you can use the **sysup** command. It will sort the packages by dependency if possible (e.g. if both `apache` and `libmm` are out of date, **prt-get** will update `libmm` first as `apache` depends on it).

```
prt-get sysup
```

### 3.6.1. Options

- **--margs=...**: arguments to be passed to pkgmk
- **--aargs=...**: arguments to be passed to pkgadd
- **--pre-install**: execute pre install scripts if available

- **--post-install**: execute post install scripts if available
- **--install-scripts**: execute pre and post install scripts if available
- **--cache**: use cache
- **--log**: write a log file

## 3.7. Locking packages

If you don't want **prt-get sysup** to update some ports, you can explicitly lock them. You will still be able to update them manually (using **prt-get update**). They will get a label "locked" in **prt-get diff**, and they won't appear in **prt-get quickdiff**

There are three commands for locking:

```
prt-get lock package1 [package2 ...]
```

```
prt-get unlock package1 [package2 ...]
```

```
prt-get listlocked [-v|-vv]
```

### 3.7.1. Options

- **-v**: Show version
- **-vv**: show version and description; slows the command a bit down

## 3.8. Pre- and Post-Install scripts

**prt-get** can execute scripts before and after a package is build and installed. Note that a failing script won't stop **prt-get**'s current install transaction. The list of installed/updated packages contains an information whether execution succeeded.

It is recommended to use the pre- and post-install options only if it is suggested by the packager of the software you want to install.

# 4. Searching and getting information about ports

## 4.1. List available ports

It can be interesting to list the ports available in your ports tree. **prt-get** provides a command for this:

```
prt-get list [-v|-vv] [filter]
```

Without arguments, **prt-get list** prints out a all ports by name. *filter* can contain a wild-card pattern which is used to limit the resulting output. Make sure you escape characters where needed.

### 4.1.1. Options:

- **-v**: adds the version and release of the ports to the output
- **-vv**: adds version, release and description<sup>[4]</sup> of the ports to the output
- **--path**: prepend path to port in output

The **-v** and **-vv** options are mutually exclusive.

### 4.1.2. Example

```
prt-get list # default, list all
prt-get list q\* # list packages starting with q"
```

## 4.2. Formatted list of available ports

```
prt-get printf format_string1
                [--sort=format_string2]
                [--filter=filter]
```

where `format_string{1,2}` is a string representing the output format or the string used for sorting. Note that these two strings are independent. You can use a pattern including wild-cards to limit the number of ports listed.

### 4.2.1. Format specifiers:

- `%n`: name
- `%p`: path
- `%v`: version
- `%r`: release
- `%u`: URL
- `%d`: Description
- `%e`: Dependencies
- `%P`: Packager
- `%M`: Maintainer
- `%R`: Has Readme ("yes", "no")
- `%i`: "no" if a package is not installed, "yes" if it's installed and up to date, "diff" if it's installed but there's a newer version in the ports tree
- `%E`: "yes" if a package has a pre-install script, "no" otherwise
- `%O`: "yes" if a package has a post-install script, "no" otherwise

Use `\n` and `\t` to format your output (no additional format specified supported)

---

<sup>[4]</sup> note that descriptions are not a requirement for a `Pkgfile` and therefore sometimes missing

## 4.2.2. Example

print all port names and versions, sort by installation status and name

```
prt-get printf "%n %v\n" --sort="%i%n"
```

of the ports starting with "q" print names and versions, sort by installation status and name

```
prt-get printf "%n %v\n" --sort="%i%n" --filter=q\*
```

## 4.3. List installed port

As **pkginfo -i**, **prt-get listinst** prints out a list of installed ports. Differences are:

- Without verbose switch, it omits the version and release
- Faster than **pkginfo** (at least on my system. Reason is most certainly the slow STL iostream implementation of gcc 2.95.3)

```
prt-get listinst [-v|-vv] [filter]
```

Without arguments, **prt-get listinst** prints out a all ports by name. Optionally a wildcard pattern can be passed as filter.

### 4.3.1. Options:

- **-v**: adds the version and release of the ports to the output
- **-vv**: adds the description of the ports to the output. *Note: will slow down the process as prt-get has to parse both the ports database and the ports tree*

## 4.4. Searching the port names

Often one would like to search the ports tree for a certain name, and fortunately this is exactly what **prt-get search** does: it searches all ports in the ports tree where a certain expression is contained in the name.

```
prt-get search [-v|-vv] expr
```

Without arguments, **prt-get search** prints out a all matching port names

### 4.4.1. Options:

- **-v**: adds the version and release of the ports to the output
- **-vv**: adds version, release and description of the ports to the output
- **--path**: prepend path to port in output
- **--regex**: interpret pattern as regular expression

The **-v** and **-vv** options are mutually exclusive.

### 4.4.2. Example:

- `prt-get search -vv icq`

## 4.5. Description Search

As the `search` command only searches in the ports names for a certain expression, there is also a command to search in both names and descriptions:

```
prt-get dsearch [-v|-vv] expr
```

Without arguments, `prt-get search` prints out a all matching port names

### 4.5.1. Options:

- `-v`: adds the version and release of the ports to the output
- `-vv`: adds version, release and description of the ports to the output
- `--path`: prepend path to port in output
- `--regex`: interpret pattern as regular expression

The `-v` and `-vv` options are mutually exclusive.

### 4.5.2. Example:

- `prt-get dsearch -vv irc`

Note that this operation requires `prt-get` to access all `Pkgfiles` which makes it rather slow (depending on the number of ports in your ports tree). Consider using caching (see section Section 8, "Using a cache file" [19]) to avoid unnecessary waiting times.

## 4.6. Searching the files of the ports

If you are looking for an program by name, you can use this function to search the ports tree for a file name in their footprint. Strips the directories from the file names before matching, so look for `grep`, not `/usr/bin/grep`.

```
prt-get fsearch pattern
```

### 4.6.1. Example:

- `prt-get fsearch "*.h"`
- `prt-get fsearch "conve*"`
- `--full`: include path in search
- `--regex`: interpret pattern as regular expression

## 4.7. Information

Once you found a port by its name, you can query all the information provided by its **Pkg-file** and collected by **prt-get**. This always includes name, version, release and physical location of the port. Many packages from the "clc/stable" and "clc/unstable" repositories also include a description, a URL where to find more information and a dependency listing. These values are only provided if the packager added them, and they are no requirement for CRUX packages (yet). If a readme file is present, it will also append a line saying "Readme: Yes" (it is omitted if there is no README file).

```
prt-get info port
```

### 4.7.1. Example:

- `prt-get info kdelibs`

### 4.7.2. Output (yours might differ):

```
Name:          kdelibs
Path:          /usr/ports/unofficial
Version:       3.0.1
Release:       1
Description:   KDE Libraries.
URL:           http://www.kde.org
Dependencies:  qt3,arts,libxml2,libxslt,libjpeg,libpng
```

## 4.8. Printing the path

It is sometimes interesting to have a look at some files of a port. To make this easier, **prt-get** has a command to print out the path of a port.

```
prt-get path port
```

### 4.8.1. Examples:

- `prt-get path lilo`
- `more `prt-get path lilo`/Pkgfile`

## 4.9. Printing the README file of a port

README files contain important information. **prt-get** therefore offers a direct way to show them.

```
prt-get readme port
```

This operation doesn't do anything if the port has no README file.

### 4.9.1. Example:

```
prt-get readme gtk2
```

## 5. Dependencies

### 5.1. Dependencies as supported by prt-get

Some packages contain dependency information, but this is optional. The dependency support **prt-get** tries to provide is to install a list of packages which are all made for the actual version of a system. It doesn't care about versions. It does not support dependencies like "package XY requires at least version n of libYZ" (breaking all program relying on the old version of libYZ), therefore avoiding one of the larger regions of dependency hell while providing a convenient help to install packages without doing a lot of research.

### 5.2. Listing dependencies

There are two commands to list dependencies

```
prt-get depends package [package2 ...]
prt-get quickdep package[package2 ...]
```

They provide a listing of packages which are required to install the 1..n packages passed as an argument. Note that those packages don't need to have any relation between each other. The difference is the output:

#### 5.2.1. Output of prt-get depends kdenetwork

```
-- dependencies ([i] = installed)
[i] zlib
[i] xfree86
[i] libjpeg
[i] libpng
[ ] freetype
[i] audiofile
[i] libxml2
[i] qt3
[i] arts
[i] libxslt
[i] kdelibs
[i] kdbase
[ ] kdenetwork

-- missing packages
qt from arts
[esd] from arts
```

You can see here all additional packages needed to build and install the package, those already installed marked with an [i]. If there are dependencies to packages which are not in the ports tree (which means that the packager made an error while writing the dependency list) they are listed under "missing packages", along with the package which requires them. Unfortunately, there is no standard how to specify those dependencies right now. **prt-get** understands whitespace and comma separated lists. The packages listed must be the name of the port in the ports tree.

#### 5.2.2. Output of prt-get quickdep kdenetwork

```
zlib xfree86 libjpeg libpng freetype \
audiofile libxml2 qt3 arts \
libxslt kdelibs kdbase kdenetwork
```

Here, the missing packages have been omitted, and it's in a much simpler format (note that they are all printed on one line). Like this, the output of **prt-get quickdep *package*** can be

used as an input to **prt-get install** to install using dependencies (See Section 9.1, "Installation using dependencies" [20]).

## 5.3. Listing dependent packages

To see the impact of a certain update command it's nice to see which packages depend on the package to be updated. To do this, you can use the **prt-get dependent** command. Note that it usually lists dependent packages which are installed. If you want to see all dependent packages, you can use the **--all** switch.

### 5.3.1. Notes:

- dependency listing is optional and therefore the output of this command is only available for those packages with accurate dependency listing.
- Only direct dependencies are shown

### 5.3.2. Options:

- **--all**: list all dependent packages, not only installed ones
- **-v**: adds the version and release of the ports to the output
- **-vv**: adds version, release and description of the ports to the output

### 5.3.3. Output of prt-get dependent

```
$ prt-get dependent qt3
arts
kdelibs
sim-icq
```

## 5.4. Tree representation of dependencies

To get a complete idea of a port's dependencies, **prt-get deptree** shows the dependencies of a port in a tree where every dependency of a port is shown as a child node. To save space, repeated subtrees are collapsed and marked with **-->**, but they can be expanded by using the **--all** switch.

### 5.4.1. Options:

- **--all**: expand all subtrees, even those already shown

### 5.4.2. Output of prt-get deptree

```
$ prt-get deptree apache
-- dependencies ([i] = installed, '-->' = seen before)
[i] apache
[i]   apr
[i]   expat
[i]   openssl
```

## 6. Package status information

### 6.1. Differences between installation and the ports tree

**prt-get** provides two commands to show differences between the packages installed and those available in the ports tree:

```
prt-get diff [--all|-v|-vv] [package1 package2...]  
prt-get quickdiff
```

#### 6.1.1. Arguments for prt-get diff

You can pass a arbitrary number of packages to check for differences. If you pass no packages, **prt-get** lists all differences. You can also use a wildcard pattern to describe which packages you want to check.

#### 6.1.2. Options for prt-get diff

- **-v**: adds the version and release of the ports to the output
- **-vv**: adds version, release and description of the ports to the output

If you locked some packages using **prt-get lock** they won't be displayed by default. To see them just add the **--all** option to the command

The **-v** and **-vv** options are mutually exclusive.

Again, the commands differ mostly by the format of their output. **prt-get quickdiff** prints a single line with only the port name of those packages to be updated. This is meant to be used as an input to **prt-get update** (see subsection Section 9.2, "Update all outdated ports" [20])

## 6.2. Check if a package is installed

A shorthand to **pkginfo -l|grep package** to print out whether a package is installed.

```
prt-get isinst package
```

## 6.3. Check the version of an installed package

To get the current version of an installed package, you can use this command:

```
prt-get current package
```

## 6.4. List duplicate packages in ports tree

Prints out ports which appear multiple times in the ports tree (means: same port name can be found in multiple directories).

```
prt-get dup
```

## 7. Using a log file

### 7.1. Description

Sometimes it's desirable to log your build results into a log file. You can enable logging both via command line arguments and the configuration file (see Section 2.5, "The logging Options" [4]) for details on how to configure it. You can also use the **--log** command line switch to enable logging.

## 8. Using a cache file

### 8.1. Using a cache file

As some operations require accessing all `Pkgfiles`, it can make sense to create a cache file containing all the information. This is no requirement, and you can decide for yourself whether you want this or not. The advantage is that most operations are a bit faster, but **prt-get dsearch** and others become *really* fast compared to non-cached operation mode. The disadvantage is that you have to update your cache file each time you change something in the ports tree (a save way to do this is calling **prt-get cache** each time **ports -u** is run).

### 8.2. Creating a cache file

Creating a cache file is very easy, it's just a matter of invoking

```
prt-get cache
```

The default location to the cache file is `/var/lib/pkg/prt-get.cache`, so make sure you have write permissions to this directory (if the directory does not exist, `prt-get` will try to create it). You can change the location of the cache file by specifying the `cachefile path` option in `/etc/prt-get.conf`

### 8.3. Creating a cache file

To use the cache, one has to invoke **prt-get** with the **--cache** switch (note the difference between the *cache command* and the *--cache switch*).

```
prt-get command --cache [OPTIONS] arguments
```

#### 8.3.1. Example:

- `prt-get dsearch --cache irc`

### 8.4. Using the cache explicitly: prt-cache

To avoid typing the **--cache** switch all the time, the **prt-get** distribution creates a symlink called **prt-cache**. The program itself detects that it's been called using this command and uses the cache automatically.

#### 8.4.1. Example:

- `prt-cache dsearch irc`

## 9. Advanced usage

### 9.1. Installation using dependencies

To install a package with all its dependencies available in the ports tree, you can just use the **depinst** command, introduced in prt-get 0.5.4.

As always, remember the test mode to see what actually would happen if you feel insecure.

### 9.2. Update all outdated ports

From version 0.4.0 of **prt-get** this can be done using the **sysup** command (See Section 3.6, "Update all outdated packages" [10]).

### 9.3. Package clusters

Sometimes multiple packages provide some sort of functionality together without having a dependency. It can be handy to create such package clusters to simplify common setup tasks. As **prt-get install** just want a list of packages, The easiest way is to create a text file listing all packages wanted and use it as an input source.

Hopefully there will be a place where everyone can publish such package clusters, for example to build office servers, development workstations, web servers, KDE desktop, GNOME desktop etc.

### 9.4. Accessing files of a port

As mentioned in subsection Section 4.8, "Printing the path" [15], it's possible to print the path where a port is located. This can be used to edit/view for example the `Pkgfile` of a `Pkgfile` of a port:

```
ls `prt-get path lilo`  
more `prt-get path lilo`/Pkgfile
```

Like this, you don't have to know the pathes of hundreds of ports in your ports tree, but you can still keep the control over the various `Pkgfiles`.

*Note:* Recent versions of **prt-get** contain a utility called **prt-edit** to edit the `Pkgfile` directly.

### 9.5. Sharing a Ports Tree

If you have multiple machines with the same CPU architecture and software installed, there is no gain in compiling the package on each machine. You can therefore share the ports tree using NFS (read/write), and instruct prt-get to look in those directories for ports. As **prt-get install/update** will not rebuild an already built packages (unless you specify the **-margs=-f** option), you can run it on all machines, and the package will only be built once.

#### 9.5.1. Example:

Imagine you have three machines, `magellan` (1.2GHz), `hoc` (1GHz) and `cogito` (700MHz). They have all i686 CPUs, and the same software installed up to now. One machine has to provide the ports tree to the others. Make sure this machine has network access, and can use **cvsup** (**cvsup** not blocked by a firewall). You don't really have to use the fastest machine for this, as you can build it on any machine you want<sup>[5]</sup>. In my ex-

---

<sup>[5]</sup> I strongly suggest to try out **distcc** to use all three machines in a compile farm

ample, I choose `hoc` as the ports tree (NFS) server.

```
hoc> ports -u
...
magellan> cat /etc/prt-get.conf
prtdir /mnt/nfs/ports/base
prtdir /mnt/nfs/ports/opt
prtdir /mnt/nfs/ports/contrib
prtdir /mnt/nfs/ports/unofficial

magellan> prt-get install irssi
[build and install irssi on the fastest machine]
...
hoc> prt-get install irssi
[install the previously built package]
...
cogito> prt-get install irssi
[install the previously built package]
```

You maybe also want to share the cache then.

## 9.6. All your base...

Sometimes, it is fun to check whether all files from `/usr/ports/base` are installed on your system. This is as easy as<sup>[6]</sup>

```
for i in `find /usr/ports/base/* -maxdepth 0`; do
    prt-get isinst `basename $i`;
done
```

You might want to add `|grep not` to see which packages are not installed

## 9.7. Recompile all installed ports

If for what reason ever you'd like to recompile your base system (e.g. after a compiler update), you can do this using the following command:

```
prt-get listinst|xargs prt-get update --margs="-f" --log
```

# 10. Copyright and licensing

## 10.1. Copyright

Copyright (c) 2002-2003 by Johannes Winkelmann, [jw@tk6.net](mailto:jw@tk6.net)

## 10.2. License

**prt-get** is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

---

<sup>[6]</sup> I'm not a bash guy, so there might be shorter commands to accomplish the same (I'm not using `ls` because its color codes - if used - can confuse `prt-get`)