NAME

pkg-config − Return metainformation about installed libraries

SYNOPSIS

pkg-config [−−modversion] [−−version] [−−help] [−−atleast-pkgconfig-version=VERSION] [−−print-errors] [−−short-errors] [−−silence-errors] [−−errors-to-stdout] [−−debug] [−−cflags] [−−libs] [−−libs-only-L] [−−libs-only-l] [−−cflags-only-I] [−−libs-only-other] [−−cflags-only-other] [−−variable=VARIABLENAME] [−−define-variable=VARIABLENAME=VARIABLEVALUE] [−−print-variables] [−−uninstalled] [−−exists] [−−atleast-version=VERSION] [−−exact-version=VERSION] [−−max-version=VERSION] [−−validate] [−−list−all] [−−print-provides] [−−print-requires] [−−print-requires-private] [LIBRARIES...]

DESCRIPTION

The *pkg-config* program is used to retrieve information about installed libraries in the system. It is typically used to compile and link against one or more libraries. Here is a typical usage scenario in a Makefile:

program: program.c
    cc program.c $(pkg-config --cflags --libs gnomeui)

pkg-config retrieves information about packages from special metadata files. These files are named after the package, and has a *.pc* extension. On most systems, *pkg-config* looks in */usr/lib/pkgconfig, /usr/share/pkgconfig, /usr/local/lib/pkgconfig* and */usr/local/share/pkgconfig* for these files. It will additionally look in the colon-separated (on Windows, semicolon-separated) list of directories specified by the PKG_CONFIG_PATH environment variable.

The package name specified on the *pkg-config* command line is defined to be the name of the metadata file, minus the *.pc* extension. If a library can install multiple versions simultaneously, it must give each version its own name (for example, GTK 1.2 might have the package name "gtk+" while GTK 2.0 has "gtk+-2.0").

In addition to specifying a package name on the command line, the full path to a given *.pc* file may be given instead. This allows a user to directly query a particular *.pc* file.

OPTIONS

The following options are supported:

--modversion
: Requests that the version information of the libraries specified on the command line be displayed. If *pkg-config* can find all the libraries on the command line, each library's version string is printed to stdout, one version per line. In this case *pkg-config* exits successfully. If one or more libraries is unknown, *pkg-config* exits with a nonzero code, and the contents of stdout are undefined.

--version
: Displays the version of *pkg-config* and terminates.

--atleast-pkgconfig-version=VERSION
: Requires at least the given version of pkg-config.

--help
: Displays a help message and terminates.

--print-errors
: If one or more of the modules on the command line, or their dependencies, are not found, or if an error occurs in parsing a *.pc* file, then this option will cause errors explaining the problem to be printed. With "predicate" options such as "--exists" *pkg-config* runs silently by default, because it's usually used in scripts that want to control what's output. This option can be used alone (to just print errors encountered locating modules on the command line) or with other options. The PKG_CONFIG_DEBUG_SPEW environment variable overrides this option.

--short-errors
: Print short error messages.

--silence-errors
: If one or more of the modules on the command line, or their dependencies, are not found, or if an error occurs in parsing a a *.pc* file, then this option will keep errors explaining the problem from being printed. With "predicate" options such as "--exists" *pkg-config* runs silently by default,

because it's usually used in scripts that want to control what's output. So this option is only useful with options such as "--cflags" or "--modversion" that print errors by default. The PKG_CONFIG_DEBUG_SPEW environment variable overrides this option.

--errors-to-stdout
If printing errors, print them to stdout rather than the default stderr

--debug
Print debugging information. This is slightly different than the PKG_CONFIG_DEBUG_SPEW environment variable, which also enable "--print-errors".

The following options are used to compile and link programs:

--cflags This prints pre-processor and compile flags required to compile the packages on the command line, including flags for all their dependencies. Flags are "compressed" so that each identical flag appears only once. *pkg-config* exits with a nonzero code if it can't find metadata for one or more of the packages on the command line.

--cflags-only-I
This prints the -I part of "--cflags". That is, it defines the header search path but doesn't specify anything else.

--cflags-only-other
This prints parts of "--cflags" not covered by "--cflags-only-I".

--libs This option is identical to "--cflags", only it prints the link flags. As with "--cflags", duplicate flags are merged (maintaining proper ordering), and flags for dependencies are included in the output.

--libs-only-L
This prints the -L/-R part of "--libs". That is, it defines the library search path but doesn't specify which libraries to link with.

--libs-only-l
This prints the -l part of "--libs" for the libraries specified on the command line. Note that the union of "--libs-only-l" and "--libs-only-L" may be smaller than "--libs", due to flags such as -rdynamic.

--libs-only-other
This prints the parts of "--libs" not covered by "--libs-only-L" and "--libs-only-l", such as "--pthread".

--variable=VARIABLENAME
This returns the value of a variable defined in a package's *.pc* file. Most packages define the variable "prefix", for example, so you can say:

```
$ pkg-config --variable=prefix glib-2.0
/usr/
```

--define-variable=VARIABLENAME=VARIABLEVALUE
This sets a global value for a variable, overriding the value in any *.pc* files. Most packages define the variable "prefix", for example, so you can say:

```
$ pkg-config --print-errors --define-variable=prefix=/foo \
        --variable=prefix glib-2.0
/foo
```

--print-variables
Returns a list of all variables defined in the package.

--uninstalled
Normally if you request the package "foo" and the package "foo-uninstalled" exists, *pkg-config* will prefer the "-uninstalled" variant. This allows compilation/linking against uninstalled packages. If you specify the "--uninstalled" option, *pkg-config* will return successfully if any "-uninstalled"

packages are being used, and return failure (false) otherwise. (The PKG_CONFIG_DISABLE_UNINSTALLED environment variable keeps *pkg-config* from implicitly choosing "-uninstalled" packages, so if that variable is set, they will only have been used if you pass a name like "foo-uninstalled" on the command line explicitly.)

--exists

--atleast-version=VERSION

--exact-version=VERSION

--max-version=VERSION

These options test whether the package or list of packages on the command line are known to *pkg-config*, and optionally whether the version number of a package meets certain constraints. If all packages exist and meet the specified version constraints, *pkg-config* exits successfully. Otherwise it exits unsuccessfully. Only the first VERSION comparing option will be honored. Subsequent options of this type will be ignored.

Rather than using the version-test options, you can simply give a version constraint after each package name, for example:

$ pkg-config --exists 'glib-2.0 >= 1.3.4 libxml = 1.8.3'

Remember to use −−print-errors if you want error messages. When no output options are supplied to *pkg-config*, −−exists is implied.

--validate

Checks the syntax of a package's *.pc* file for validity. This is the same as −−exists except that dependencies are not verified. This can be useful for package developers to test their *.pc* file prior to release:

$ pkg-config --validate ./my-package.pc

--msvc-syntax

This option is available only on Windows. It causes *pkg-config* to output -l and -L flags in the form recognized by the Microsoft Visual C++ command-line compiler, *cl*. Specifically, instead of *-Lx:/some/path* it prints */libpath:x/some/path*, and instead of *-lfoo* it prints *foo.lib*. Note that the --libs output consists of flags for the linker, and should be placed on the cl command line after a /link switch.

--define-prefix

--dont-define-prefix

These options control whether *pkg-config* overrides the value of the variable *prefix* in each .pc file. With −−define-prefix, *pkg-config* uses the installed location of the .pc file to determine the prefix. −−dont-define-prefix prevents this behavior. The default is usually −−define-prefix.

When this feature is enabled and a .pc file is found in a directory named *pkgconfig*, the prefix for that package is assumed to be the grandparent of the directory where the file was found, and the *prefix* variable is overridden for that file accordingly.

If the value of a variable in a .pc file begins with the original, non-overridden, value of the *prefix* variable, then the overridden value of *prefix* is used instead. This allows the feature to work even when the variables have been expanded in the .pc file.

--prefix-variable=PREFIX

Set the name of the variable that *pkg-config* overrides instead of *prefix* when using the −−define-prefix feature.

--static Output libraries suitable for static linking. That means including any private libraries in the output. This relies on proper tagging in the .pc files, else a too large number of libraries will ordinarily be output.

--list-all
List all modules found in the *pkg-config* path.

--print-provides
List all modules the given packages provides.

--print-requires
List all modules the given packages requires.

--print-requires-private
List all modules the given packages requires for static linking (see --static).

ENVIRONMENT VARIABLES

PKG_CONFIG_PATH
A colon-separated (on Windows, semicolon-separated) list of directories to search for .pc files. The default directory will always be searched after searching the path; the default is *libdir*/pkgconfig:*datadir*/pkgconfig where *libdir* is the libdir for *pkg-config* and *datadir* is the datadir for *pkg-config* when it was installed.

PKG_CONFIG_DEBUG_SPEW
If set, causes *pkg-config* to print all kinds of debugging information and report all errors.

PKG_CONFIG_TOP_BUILD_DIR
A value to set for the magic variable *pc_top_builddir* which may appear in *.pc* files. If the environment variable is not set, the default value '$(top_builddir)' will be used. This variable should refer to the top builddir of the Makefile where the compile/link flags reported by *pkg-config* will be used. This only matters when compiling/linking against a package that hasn't yet been installed.

PKG_CONFIG_DISABLE_UNINSTALLED
Normally if you request the package "foo" and the package "foo-uninstalled" exists, *pkg-config* will prefer the "-uninstalled" variant. This allows compilation/linking against uninstalled packages. If this environment variable is set, it disables said behavior.

PKG_CONFIG_ALLOW_SYSTEM_CFLAGS
Don't strip -I/usr/include out of cflags.

PKG_CONFIG_ALLOW_SYSTEM_LIBS
Don't strip -L/usr/lib or -L/lib out of libs.

PKG_CONFIG_SYSROOT_DIR
Modify -I and -L to use the directories located in target sysroot. this option is useful when cross-compiling packages that use pkg-config to determine CFLAGS and LDFLAGS. -I and -L are modified to point to the new system root. this means that a -I/usr/include/libfoo will become -I/var/target/usr/include/libfoo with a PKG_CONFIG_SYSROOT_DIR equal to /var/target (same rule apply to -L)

PKG_CONFIG_LIBDIR
Replaces the default *pkg-config* search directory, usually */usr/lib/pkgconfig*:*/usr/share/pkgconfig*.

PKG_CONFIG_$PACKAGE_$VARIABLE
Overrides the variable VARIABLE in the package PACKAGE. The environment variable should have the package name and package variable upper cased with non-alphanumeric characters converted to underscores. For example, setting PKG_CONFIG_GLADEUI_2_0_CATALOGDIR will override the variable "catalogdir" in the "gladeui-2.0" package.

PKG-CONFIG DERIVED VARIABLES

pkg-config sets a few metadata variables that can be used in .pc files or queried at runtime.

pc_path
The default search path used by *pkg-config* when searching for .pc files. This can be used in a query for the *pkg-config* module itself itself:
$ pkg-config --variable pc_path pkg-config

pcfiledir
: The installed location of the .pc file. This can be used to query the location of the .pc file for a particular module, but it can also be used to make .pc files relocatable. For instance:
prefix=${pcfiledir}/../..
exec_prefix=${prefix}
libdir=${exec_prefix}/lib
includedir=${prefix}/include

pc_sysrootdir
: The sysroot directory set by the user. When the sysroot directory has not been set, this value is `/`. See the *PKG_CONFIG_SYSROOT_DIR* environment variable for more details.

pc_top_builddir
: Location of the user's top build directory when calling *pkg-config*. This is useful to dynamically set paths in uninstalled .pc files. See the *PKG_CONFIG_TOP_BUILD_DIR* environment variable for more details.

WINDOWS SPECIALITIES

The *pkg-config* default search path is ignored on Windows. Instead, the search path is constructed by using the installed directory of *pkg-config* and then appending *lib\pkgconfig* and *share\pkgconfig*. This can be augmented or replaced using the standard environment variables described above.

AUTOCONF MACROS

PKG_CHECK_MODULES(VARIABLE-PREFIX, MODULES [,ACTION-IF-FOUND [,ACTION-IF-NOT-FOUND]])

The macro PKG_CHECK_MODULES can be used in *configure.ac* to check whether modules exist. A typical usage would be:
PKG_CHECK_MODULES([MYSTUFF], [gtk+-2.0 >= 1.3.5 libxml = 1.8.4])

This would result in MYSTUFF_LIBS and MYSTUFF_CFLAGS substitution variables, set to the libs and cflags for the given module list. If a module is missing or has the wrong version, by default configure will abort with a message. To replace the default action, specify an ACTION-IF-NOT-FOUND. PKG_CHECK_MODULES will not print any error messages if you specify your own ACTION-IF-NOT-FOUND. However, it will set the variable MYSTUFF_PKG_ERRORS, which you can use to display what went wrong.

Note that if there is a possibility the first call to PKG_CHECK_MODULES might not happen, you should be sure to include an explicit call to PKG_PROG_PKG_CONFIG in your configure.ac.

Also note that repeated usage of VARIABLE-PREFIX is not recommended. After the first successful usage, subsequent calls with the same VARIABLE-PREFIX will simply use the _LIBS and _CFLAGS variables set from the previous usage without calling *pkg-config* again.

PKG_PREREQ(MIN-VERSION)
: Checks that the version of the pkg-config autoconf macros in use is at least MIN-VERSION. This can be used to ensure a particular pkg-config macro will be available.

PKG_PROG_PKG_CONFIG([MIN-VERSION])

Defines the PKG_CONFIG variable to the best pkg-config available, useful if you need pkg-config but don't want to use PKG_CHECK_MODULES.

PKG_CHECK_MODULES_STATIC(VARIABLE-PREFIX, MODULES [,ACTION-IF-FOUND [,ACTION-IF-NOT-FOUND]])
: Enables static linking through --static prior to calling PKG_CHECK_MODULES.

PKG_CHECK_EXISTS(MODULES, [ACTION-IF-FOUND], [ACTION-IF-NOT-FOUND])

Check to see whether a particular set of modules exists. Similar to PKG_CHECK_MODULES(), but does not set variables or print errors.

Similar to PKG_CHECK_MODULES, make sure that the first instance of this or PKG_CHECK_MODULES is called, or make sure to call PKG_CHECK_EXISTS manually.

PKG_INSTALLDIR(DIRECTORY)

Substitutes the variable pkgconfigdir as the location where a module should install pkg-config .pc files. By default the directory is $libdir/pkgconfig, but the default can be changed by passing DIRECTORY. The user can override through the --with-pkgconfigdir parameter.

PKG_NOARCH_INSTALLDIR(DIRECTORY)

Substitutes the variable noarch_pkgconfigdir as the location where a module should install arch-independent pkg-config .pc files. By default the directory is $datadir/pkgconfig, but the default can be changed by passing DIRECTORY. The user can override through the --with-noarch-pkgconfigdir parameter.

PKG_CHECK_VAR(VARIABLE, MODULE, CONFIG-VARIABLE, [ACTION-IF-FOUND], [ACTION-IF-NOT-FOUND])

Retrieves the value of the pkg-config variable CONFIG-VARIABLE from MODULE and stores it in VARIABLE. Note that repeated usage of VARIABLE is not recommended as the check will be skipped if the variable is already set.

METADATA FILE SYNTAX

To add a library to the set of packages *pkg-config* knows about, simply install a *.pc* file. You should install this file to *libdir*/pkgconfig.

Here is an example file:

```
# This is a comment
prefix=/home/hp/unst   # this defines a variable
exec_prefix=${prefix}  # defining another variable in terms of the first
libdir=${exec_prefix}/lib
includedir=${prefix}/include

Name: GObject                    # human-readable name
Description: Object/type system for GLib # human-readable description
Version: 1.3.1
URL: http://www.gtk.org
Requires: glib-2.0 = 1.3.1
Conflicts: foobar <= 4.5
Libs: -L${libdir} -lgobject-1.3
Libs.private: -lm
Cflags: -I${includedir}/glib-2.0 -I${libdir}/glib/include
```

You would normally generate the file using configure, so that the prefix, etc. are set to the proper values. The GNU Autoconf manual recommends generating files like .pc files at build time rather than configure time, so when you build the .pc file is a matter of taste and preference.

Files have two kinds of line: keyword lines start with a keyword plus a colon, and variable definitions start with an alphanumeric string plus an equals sign. Keywords are defined in advance and have special meaning to *pkg-config*; variables do not, you can have any variables that you wish (however, users may expect to retrieve the usual directory name variables).

Note that variable references are written "${foo}"; you can escape literal "${" as "$${".

Name: This field should be a human-readable name for the package. Note that it is not the name passed as an argument to *pkg-config*.

Description:
This should be a brief description of the package

URL: An URL where people can get more information about and download the package

Version:
This should be the most-specific-possible package version string.

Requires:
This is a comma-separated list of packages that are required by your package. Flags from dependent packages will be merged in to the flags reported for your package. Optionally, you can specify the version of the required package (using the operators =, <, >, >=, <=); specifying a version allows *pkg-config* to perform extra sanity checks. You may only mention the same package one time on the *Requires:* line. If the version of a package is unspecified, any version will be used with no checking.

Requires.private:
A list of packages required by this package. The difference from *Requires* is that the packages listed under *Requires.private* are not taken into account when a flag list is computed for dynamically linked executable (i.e., when −−static was not specified). In the situation where each .pc file corresponds to a library, *Requires.private* shall be used exclusively to specify the dependencies between the libraries.

Conflicts:
This optional line allows *pkg-config* to perform additional sanity checks, primarily to detect broken user installations. The syntax is the same as *Requires:* except that you can list the same package more than once here, for example "foobar = 1.2.3, foobar = 1.2.5, foobar >= 1.3", if you have reason to do so. If a version isn't specified, then your package conflicts with all versions of the mentioned package. If a user tries to use your package and a conflicting package at the same time, then *pkg-config* will complain.

Libs: This line should give the link flags specific to your package. Don't add any flags for required packages; *pkg-config* will add those automatically.

Libs.private:
This line should list any private libraries in use. Private libraries are libraries which are not exposed through your library, but are needed in the case of static linking. This differs from *Requires.private* in that it references libraries that do not have package files installed.

Cflags: This line should list the compile flags specific to your package. Don't add any flags for required packages; *pkg-config* will add those automatically.

AUTHOR

pkg-config was written by James Henstridge, rewritten by Martijn van Beers, and rewritten again by Havoc Pennington. Tim Janik, Owen Taylor, and Raja Harinath submitted suggestions and some code. *gnome-config* was written by Miguel de Icaza, Raja Harinath and various hackers in the GNOME team. It was inspired by Owen Taylor's *gtk-config* program.

BUGS

pkg-config does not handle mixing of parameters with and without = well. Stick with one.

Bugs can be reported at http://bugs.freedesktop.org/ under the *pkg-config* component.