pkgconf Documentation

Release 1.1.0

pkgconf authors

Sep 03, 2023

Contents

CHAPTER 1

libpkgconf - an API for managing *pkg-config* modules

1.1 libpkgconf *argvsplit* module

This is a lowlevel module which provides parsing of strings into argument vectors, similar to what a shell would do.

void **pkgconf_argv_free**(char ***argv*)
Frees an argument vector.

Parameters

- **argv** (*char***) – The argument vector to free.

Returns nothing

int **pkgconf_argv_split**(const char **src*, int **argc*, char ****argv*)
Splits a string into an argument vector.

Parameters

- **src** (*char**) – The string to split.
- **argc** (*int**) – A pointer to an integer to store the argument count.
- **argv** (*char****) – A pointer to a pointer for an argument vector.

Returns 0 on success, -1 on error.

Return type int

1.2 libpkgconf *audit* module

The libpkgconf *audit* module contains the functions related to attaching an audit log file to a `pkgconf_client_t` object.

The audit log format is the same as the output generated by the `PKG_CONFIG_LOG` environment variable.

void **pkgconf_audit_set_log**(pkgconf_client_t *client, FILE *auditf)
Sets the audit log file pointer on *client* to *auditf*. The callee is responsible for closing any previous log files.

Parameters

- **client** (*pkgconf_client_t**) – The client object to modify.
- **auditf** (*FILE**) – The file pointer for the already open log file.

Returns nothing

void **pkgconf_audit_log**(pkgconf_client_t *client, const char *format, ...)
Logs a message to the opened audit log (if any).

Parameters

- **client** (*pkgconf_client_t**) – The client object the log message is for.
- **format** (*char**) – The format string to use for the log messages.

Returns nothing

void **pkgconf_audit_log_dependency**(pkgconf_client_t *client, const pkgconf_pkg_t *dep, const pkgconf_dependency_t *depnode)
Convenience function which logs a dependency node to the opened audit log (if any).

Parameters

- **client** (*pkgconf_client_t**) – The client object the log message is for.
- **dep** (*pkgconf_pkg_t**) – The dependency package object being logged.
- **depnode** (*pkgconf_dependency_t**) – The dependency object itself being logged.

Returns nothing

1.3 libpkgconf *cache* module

The libpkgconf *cache* module manages a package/module object cache, allowing it to avoid loading duplicate copies of a package/module.

A cache is tied to a specific pkgconf client object, so package objects should not be shared across threads.

pkgconf_pkg_t ***pkgconf_cache_lookup**(const pkgconf_client_t *client, const char *id)
Looks up a package in the cache given an *id* atom, such as `gtk+-3.0` and returns the already loaded version if present.

Parameters

- **client** (*pkgconf_client_t**) – The client object to access.
- **id** (*char**) – The package atom to look up in the client object's cache.

Returns A package object if present, else `NULL`.

Return type pkgconf_pkg_t *

void **pkgconf_cache_add**(pkgconf_client_t *client, pkgconf_pkg_t *pkg)
Adds an entry for the package to the package cache. The cache entry must be removed if the package is freed.

Parameters

- **client** (*pkgconf_client_t**) – The client object to modify.
- **pkg** (*pkgconf_pkg_t**) – The package object to add to the client object's cache.

Returns nothing

void **pkgconf_cache_remove**(pkgconf_client_t **client*, pkgconf_pkg_t **pkg*)
Deletes a package from the client object's package cache.

Parameters

- **client** (*pkgconf_client_t **) – The client object to modify.
- **pkg** (*pkgconf_pkg_t **) – The package object to remove from the client object's cache.

Returns nothing

void **pkgconf_cache_free**(pkgconf_client_t **client*)
Releases all resources related to a client object's package cache. This function should only be called to clear a client object's package cache, as it may release any package in the cache.

Parameters

- **client** (*pkgconf_client_t **) – The client object to modify.

1.4 libpkgconf *client* module

The libpkgconf *client* module implements the *pkgconf_client_t* "client" object. Client objects store all necessary state for libpkgconf allowing for multiple instances to run in parallel.

Client objects are not thread safe, in other words, a client object should not be shared across thread boundaries.

void **pkgconf_client_dir_list_build**(pkgconf_client_t **client*)
Bootstraps the package search paths. If the `PKGCONF_PKG_PKGF_ENV_ONLY` *flag* is set on the client, then only the `PKG_CONFIG_PATH` environment variable will be used, otherwise both the `PKG_CONFIG_PATH` and `PKG_CONFIG_LIBDIR` environment variables will be used.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to bootstrap.

Returns nothing

void **pkgconf_client_init**(pkgconf_client_t **client*, pkgconf_error_handler_func_t *error_handler*, void **error_handler_data*, const pkgconf_cross_personality_t **personality*)
Initialise a pkgconf client object.

Parameters

- **client** (*pkgconf_client_t **) – The client to initialise.
- **error_handler** (*pkgconf_error_handler_func_t*) – An optional error handler to use for logging errors.
- **error_handler_data** (*void**) – user data passed to optional error handler
- **personality** (*pkgconf_cross_personality_t **) – the cross-compile personality to use for defaults

Returns nothing

pkgconf_client_t* **pkgconf_client_new**(pkgconf_error_handler_func_t *error_handler*, void **error_handler_data*, const pkgconf_cross_personality_t **personality*)
Allocate and initialise a pkgconf client object.

Parameters

- **error_handler** (*pkgconf_error_handler_func_t*) – An optional error handler to use for logging errors.
- **error_handler_data** (*void**) – user data passed to optional error handler
- **personality** (*pkgconf_cross_personality_t**) – cross-compile personality to use

Returns A pkgconf client object.

Return type pkgconf_client_t*

void **pkgconf_client_deinit** (pkgconf_client_t **client*)
Release resources belonging to a pkgconf client object.

Parameters

- **client** (*pkgconf_client_t**) – The client to deinitialise.

Returns nothing

void **pkgconf_client_free** (pkgconf_client_t **client*)
Release resources belonging to a pkgconf client object and then free the client object itself.

Parameters

- **client** (*pkgconf_client_t**) – The client to deinitialise and free.

Returns nothing

const char ***pkgconf_client_get_sysroot_dir** (const pkgconf_client_t **client*)
Retrieves the client's sysroot directory (if any).

Parameters

- **client** (*pkgconf_client_t**) – The client object being accessed.

Returns A string containing the sysroot directory or NULL.

Return type const char *

void **pkgconf_client_set_sysroot_dir** (pkgconf_client_t **client*, const char **sysroot_dir*)
Sets or clears the sysroot directory on a client object. Any previous sysroot directory setting is automatically released if one was previously set.

Additionally, the global tuple `$(pc_sysrootdir)` is set as appropriate based on the new setting.

Parameters

- **client** (*pkgconf_client_t**) – The client object being modified.
- **sysroot_dir** (*char**) – The sysroot directory to set or NULL to unset.

Returns nothing

const char ***pkgconf_client_get_buildroot_dir** (const pkgconf_client_t **client*)
Retrieves the client's buildroot directory (if any).

Parameters

- **client** (*pkgconf_client_t**) – The client object being accessed.

Returns A string containing the buildroot directory or NULL.

Return type const char *

void **pkgconf_client_set_buildroot_dir**(pkgconf_client_t **client*, const char **buildroot_dir*)
Sets or clears the buildroot directory on a client object. Any previous buildroot directory setting is automatically released if one was previously set.

Additionally, the global tuple `$(pc_top_builddir)` is set as appropriate based on the new setting.

Parameters

- **client** (*pkgconf_client_t**) – The client object being modified.
- **buildroot_dir** (*char**) – The buildroot directory to set or NULL to unset.

Returns nothing

bool **pkgconf_error**(const pkgconf_client_t **client*, const char **format*, ...)
Report an error to a client-registered error handler.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to report the error to.
- **format** (*char**) – A printf-style format string to use for formatting the error message.

Returns true if the error handler processed the message, else false.

Return type bool

bool **pkgconf_warn**(const pkgconf_client_t **client*, const char **format*, ...)
Report an error to a client-registered warn handler.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to report the error to.
- **format** (*char**) – A printf-style format string to use for formatting the warning message.

Returns true if the warn handler processed the message, else false.

Return type bool

bool **pkgconf_trace**(const pkgconf_client_t **client*, const char **filename*, size_t *len*, const char **funcname*, const char **format*, ...)
Report a message to a client-registered trace handler.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to report the trace message to.
- **filename** (*char**) – The file the function is in.
- **lineno** (*size_t*) – The line number currently being executed.
- **funcname** (*char**) – The function name to use.
- **format** (*char**) – A printf-style format string to use for formatting the trace message.

Returns true if the trace handler processed the message, else false.

Return type bool

bool **pkgconf_default_error_handler**(const char **msg*, const pkgconf_client_t **client*, const void **data*)
The default pkgconf error handler.

Parameters

- **msg** (*char**) – The error message to handle.

- **client** (*pkgconf_client_t**) – The client object the error originated from.
- **data** (*void**) – An opaque pointer to extra data associated with the client for error handling.

Returns true (the function does nothing to process the message)

Return type bool

unsigned int **pkgconf_client_get_flags** (const pkgconf_client_t **client*)
Retrieves resolver-specific flags associated with a client object.

Parameters

- **client** (*pkgconf_client_t**) – The client object to retrieve the resolver-specific flags from.

Returns a bitfield of resolver-specific flags

Return type uint

void **pkgconf_client_set_flags** (pkgconf_client_t **client*, unsigned int *flags*)
Sets resolver-specific flags associated with a client object.

Parameters

- **client** (*pkgconf_client_t**) – The client object to set the resolver-specific flags on.

Returns nothing

const char ***pkgconf_client_get_prefix_varname** (const pkgconf_client_t **client*)
Retrieves the name of the variable that should contain a module's prefix. In some cases, it is necessary to override this variable to allow proper path relocation.

Parameters

- **client** (*pkgconf_client_t**) – The client object to retrieve the prefix variable name from.

Returns the prefix variable name as a string

Return type const char *

void **pkgconf_client_set_prefix_varname** (pkgconf_client_t **client*, const char **prefix_varname*)
Sets the name of the variable that should contain a module's prefix. If the variable name is `NULL`, then the default variable name (`prefix`) is used.

Parameters

- **client** (*pkgconf_client_t**) – The client object to set the prefix variable name on.
- **prefix_varname** (*char**) – The prefix variable name to set.

Returns nothing

pkgconf_client_get_warn_handler (const pkgconf_client_t **client*)
Returns the warning handler if one is set, else `NULL`.

Parameters

- **client** (*pkgconf_client_t**) – The client object to get the warn handler from.

Returns a function pointer to the warn handler or `NULL`

pkgconf_client_set_warn_handler (pkgconf_client_t **client*, pkgconf_error_handler_func_t *warn_handler*, void **warn_handler_data*)
Sets a warn handler on a client object or uninstalls one if set to `NULL`.

Parameters

- **client** (*pkgconf_client_t**) – The client object to set the warn handler on.
- **warn_handler** (*pkgconf_error_handler_func_t*) – The warn handler to set.
- **warn_handler_data** (*void**) – Optional data to associate with the warn handler.

Returns nothing

pkgconf_client_get_error_handler (const pkgconf_client_t *client)
Returns the error handler if one is set, else NULL.

Parameters

- **client** (*pkgconf_client_t**) – The client object to get the error handler from.

Returns a function pointer to the error handler or NULL

pkgconf_client_set_error_handler (pkgconf_client_t *client, pkgconf_error_handler_func_t *error_handler*, void *error_handler_data)
Sets a warn handler on a client object or uninstalls one if set to NULL.

Parameters

- **client** (*pkgconf_client_t**) – The client object to set the error handler on.
- **error_handler** (*pkgconf_error_handler_func_t*) – The error handler to set.
- **error_handler_data** (*void**) – Optional data to associate with the error handler.

Returns nothing

pkgconf_client_get_trace_handler (const pkgconf_client_t *client)
Returns the error handler if one is set, else NULL.

Parameters

- **client** (*pkgconf_client_t**) – The client object to get the error handler from.

Returns a function pointer to the error handler or NULL

pkgconf_client_set_trace_handler (pkgconf_client_t *client, pkgconf_error_handler_func_t *trace_handler*, void *trace_handler_data)
Sets a warn handler on a client object or uninstalls one if set to NULL.

Parameters

- **client** (*pkgconf_client_t**) – The client object to set the error handler on.
- **trace_handler** (*pkgconf_error_handler_func_t*) – The error handler to set.
- **trace_handler_data** (*void**) – Optional data to associate with the error handler.

Returns nothing

1.5 libpkgconf *dependency* module

The *dependency* module provides support for building *dependency lists* (the basic component of the overall *dependency graph*) and *dependency nodes* which store dependency information.

pkgconf_dependency_t *__pkgconf_dependency_add__(pkgconf_list_t *list, const char *package, const char *version, pkgconf_pkg_comparator_t compare)

Adds a parsed dependency to a dependency list as a dependency node.

Parameters

- **client** (*pkgconf_client_t**) – The client object that owns the package this dependency list belongs to.
- **list** (*pkgconf_list_t**) – The dependency list to add a dependency node to.
- **package** (*char**) – The package *atom* to set on the dependency node.
- **version** (*char**) – The package *version* to set on the dependency node.
- **compare** (*pkgconf_pkg_comparator_t*) – The comparison operator to set on the dependency node.
- **flags** (*uint*) – Any flags to attach to the dependency node.

Returns A dependency node.

Return type pkgconf_dependency_t *

void __pkgconf_dependency_append__(pkgconf_list_t *list, pkgconf_dependency_t *tail)

Adds a dependency node to a pre-existing dependency list.

Parameters

- **list** (*pkgconf_list_t**) – The dependency list to add a dependency node to.
- **tail** (*pkgconf_dependency_t**) – The dependency node to add to the tail of the dependency list.

Returns nothing

void __pkgconf_dependency_free_one__(pkgconf_dependency_t *dep)

Frees a dependency node.

Parameters

- **dep** (*pkgconf_dependency_t**) – The dependency node to free.

Returns nothing

pkgconf_dependency_t *__pkgconf_dependency_ref__(pkgconf_client_t *owner, pkgconf_dependency_t *dep)

Increases a dependency node's refcount.

Parameters

- **owner** (*pkgconf_client_t**) – The client object which owns the memory of this dependency node.
- **dep** (*pkgconf_dependency_t**) – The dependency to increase the refcount of.

Returns the dependency node on success, else NULL

void __pkgconf_dependency_unref__(pkgconf_client_t *owner, pkgconf_dependency_t *dep)

Decreases a dependency node's refcount and frees it if necessary.

Parameters

- **owner** (*pkgconf_client_t**) – The client object which owns the memory of this dependency node.
- **dep** (*pkgconf_dependency_t**) – The dependency to decrease the refcount of.

Returns nothing

void **pkgconf_dependency_free**(pkgconf_list_t **list*)
Release a dependency list and it's child dependency nodes.

Parameters

- **list** (*pkgconf_list_t**) – The dependency list to release.

Returns nothing

void **pkgconf_dependency_parse_str**(pkgconf_list_t **deplist_head*, const char **depends*)
Parse a dependency declaration into a dependency list. Commas are counted as whitespace to allow for constructs such as `@SUBSTVAR@, zlib` being processed into `, zlib`.

Parameters

- **client** (*pkgconf_client_t**) – The client object that owns the package this dependency list belongs to.
- **deplist_head** (*pkgconf_list_t**) – The dependency list to populate with dependency nodes.
- **depends** (*char**) – The dependency data to parse.
- **flags** (*uint*) – Any flags to attach to the dependency nodes.

Returns nothing

void **pkgconf_dependency_parse**(const pkgconf_client_t **client*, pkgconf_pkg_t **pkg*, pkgconf_list_t **deplist*, const char **depends*)
Preprocess dependency data and then process that dependency declaration into a dependency list. Commas are counted as whitespace to allow for constructs such as `@SUBSTVAR@, zlib` being processed into `, zlib`.

Parameters

- **client** (*pkgconf_client_t**) – The client object that owns the package this dependency list belongs to.
- **pkg** (*pkgconf_pkg_t**) – The package object that owns this dependency list.
- **deplist** (*pkgconf_list_t**) – The dependency list to populate with dependency nodes.
- **depends** (*char**) – The dependency data to parse.
- **flags** (*uint*) – Any flags to attach to the dependency nodes.

Returns nothing

pkgconf_dependency_t ***pkgconf_dependency_copy**(pkgconf_client_t **client*, const pkgconf_dependency_t **dep*)
Copies a dependency node to a new one.

Parameters

- **client** (*pkgconf_client_t**) – The client object that will own this dependency.
- **dep** (*pkgconf_dependency_t**) – The dependency node to copy.

Returns a pointer to a new dependency node, else NULL

1.6 libpkgconf *fragment* module

The *fragment* module provides low-level management and rendering of fragment lists. A *fragment list* contains various *fragments* of text (such as `-I /usr/include`) in a matter which is composable, mergeable and reorderable.

void **pkgconf_fragment_add**(const pkgconf_client_t **client*, pkgconf_list_t **list*, const char **string*, unsigned int *flags*)

Adds a *fragment* of text to a *fragment list*, possibly modifying the fragment if a sysroot is set.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client being accessed.
- **list** (*pkgconf_list_t**) – The fragment list.
- **string** (*char**) – The string of text to add as a fragment to the fragment list.
- **flags** (*uint*) – Parsing-related flags for the package.

Returns nothing

bool **pkgconf_fragment_has_system_dir**(const pkgconf_client_t **client*, const pkgconf_fragment_t **frag*)

Checks if a *fragment* contains a *system path*. System paths are detected at compile time and optionally overridden by the `PKG_CONFIG_SYSTEM_INCLUDE_PATH` and `PKG_CONFIG_SYSTEM_LIBRARY_PATH` environment variables.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object the fragment belongs to.
- **frag** (*pkgconf_fragment_t**) – The fragment being checked.

Returns true if the fragment contains a system path, else false

Return type bool

void **pkgconf_fragment_copy**(const pkgconf_client_t **client*, pkgconf_list_t **list*, const pkgconf_fragment_t **base*, bool *is_private*)

Copies a *fragment* to another *fragment list*, possibly removing a previous copy of the *fragment* in a process known as *mergeback*.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client being accessed.
- **list** (*pkgconf_list_t**) – The list the fragment is being added to.
- **base** (*pkgconf_fragment_t**) – The fragment being copied.
- **is_private** (*bool*) – Whether the fragment list is a *private* fragment list (static linking).

Returns nothing

void **pkgconf_fragment_copy_list**(const pkgconf_client_t **client*, pkgconf_list_t **list*, const pkgconf_list_t **base*)

Copies a *fragment list* to another *fragment list*, possibly removing a previous copy of the fragments in a process known as *mergeback*.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client being accessed.
- **list** (*pkgconf_list_t**) – The list the fragments are being added to.
- **base** (*pkgconf_list_t**) – The list the fragments are being copied from.

Returns nothing

void **pkgconf_fragment_filter**(const pkgconf_client_t **client*, pkgconf_list_t **dest*, pkgconf_list_t **src*, pkgconf_fragment_filter_func_t *filter_func*)

Copies a *fragment list* to another *fragment list* which match a user-specified filtering function.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client being accessed.
- **dest** (*pkgconf_list_t**) – The destination list.
- **src** (*pkgconf_list_t**) – The source list.
- **filter_func** (*pkgconf_fragment_filter_func_t*) – The filter function to use.
- **data** (*void**) – Optional data to pass to the filter function.

Returns nothing

size_t **pkgconf_fragment_render_len**(const pkgconf_list_t **list*, bool *escape*, const pkgconf_fragment_render_ops_t **ops*)

Calculates the required memory to store a *fragment list* when rendered as a string.

Parameters

- **list** (*pkgconf_list_t**) – The *fragment list* being rendered.
- **escape** (*bool*) – Whether or not to escape special shell characters (deprecated).
- **ops** (*pkgconf_fragment_render_ops_t**) – An optional ops structure to use for custom renderers, else `NULL`.

Returns the amount of bytes required to represent the *fragment list* when rendered

Return type size_t

void **pkgconf_fragment_render_buf**(const pkgconf_list_t **list*, char **buf*, size_t *buflen*, bool *escape*, const pkgconf_fragment_render_ops_t **ops*)

Renders a *fragment list* into a buffer.

Parameters

- **list** (*pkgconf_list_t**) – The *fragment list* being rendered.
- **buf** (*char**) – The buffer to render the fragment list into.
- **buflen** (*size_t*) – The length of the buffer.
- **escape** (*bool*) – Whether or not to escape special shell characters (deprecated).
- **ops** (*pkgconf_fragment_render_ops_t**) – An optional ops structure to use for custom renderers, else `NULL`.

Returns nothing

char ***pkgconf_fragment_render**(const pkgconf_list_t **list*)

Allocate memory and render a *fragment list* into it.

Parameters

- **list** (*pkgconf_list_t**) – The *fragment list* being rendered.
- **escape** (*bool*) – Whether or not to escape special shell characters (deprecated).
- **ops** (*pkgconf_fragment_render_ops_t**) – An optional ops structure to use for custom renderers, else `NULL`.

Returns An allocated string containing the rendered *fragment list*.

Return type char *

void **pkgconf_fragment_delete**(pkgconf_list_t *list, pkgconf_fragment_t *node)
Delete a *fragment node* from a *fragment list*.

Parameters

- **list** (*pkgconf_list_t **) – The *fragment list* to delete from.
- **node** (*pkgconf_fragment_t **) – The *fragment node* to delete.

Returns nothing

void **pkgconf_fragment_free**(pkgconf_list_t *list)
Delete an entire *fragment list*.

Parameters

- **list** (*pkgconf_list_t **) – The *fragment list* to delete.

Returns nothing

bool **pkgconf_fragment_parse**(const pkgconf_client_t *client, pkgconf_list_t *list, pkgconf_list_t *vars, const char *value)
Parse a string into a *fragment list*.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client being accessed.
- **list** (*pkgconf_list_t **) – The *fragment list* to add the fragment entries to.
- **vars** (*pkgconf_list_t **) – A list of variables to use for variable substitution.
- **flags** (*uint*) – Any parsing flags to be aware of.
- **value** (*char**) – The string to parse into fragments.

Returns true on success, false on parse error

1.7 libpkgconf *path* module

The *path* module provides functions for manipulating lists of paths in a cross-platform manner. Notably, it is used by the *pkgconf client* to parse the PKG_CONFIG_PATH, PKG_CONFIG_LIBDIR and related environment variables.

void **pkgconf_path_add**(const char *text, pkgconf_list_t *dirlist)
Adds a path node to a path list. If the path is already in the list, do nothing.

Parameters

- **text** (*char**) – The path text to add as a path node.
- **dirlist** (*pkgconf_list_t **) – The path list to add the path node to.
- **filter** (*bool*) – Whether to perform duplicate filtering.

Returns nothing

size_t **pkgconf_path_split**(const char *text, pkgconf_list_t *dirlist)
Splits a given text input and inserts paths into a path list.

Parameters

- **text** (*char**) – The path text to split and add as path nodes.
- **dirlist** (*pkgconf_list_t **) – The path list to have the path nodes added to.

- **filter** (*bool*) – Whether to perform duplicate filtering.

Returns number of path nodes added to the path list

Return type size_t

size_t **pkgconf_path_build_from_environ**(const char **envvarname*, const char **fallback*, pkgconf_list_t **dirlist*)

Adds the paths specified in an environment variable to a path list. If the environment variable is not set, an optional default set of paths is added.

Parameters

- **envvarname** (*char**) – The environment variable to look up.
- **fallback** (*char**) – The fallback paths to use if the environment variable is not set.
- **dirlist** (*pkgconf_list_t**) – The path list to add the path nodes to.
- **filter** (*bool*) – Whether to perform duplicate filtering.

Returns number of path nodes added to the path list

Return type size_t

bool **pkgconf_path_match_list**(const char **path*, const pkgconf_list_t **dirlist*)

Checks whether a path has a matching prefix in a path list.

Parameters

- **path** (*char**) – The path to check against a path list.
- **dirlist** (*pkgconf_list_t**) – The path list to check the path against.

Returns true if the path list has a matching prefix, otherwise false

Return type bool

void **pkgconf_path_copy_list**(pkgconf_list_t **dst*, const pkgconf_list_t **src*)

Copies a path list to another path list.

Parameters

- **dst** (*pkgconf_list_t**) – The path list to copy to.
- **src** (*pkgconf_list_t**) – The path list to copy from.

Returns nothing

void **pkgconf_path_free**(pkgconf_list_t **dirlist*)

Releases any path nodes attached to the given path list.

Parameters

- **dirlist** (*pkgconf_list_t**) – The path list to clean up.

Returns nothing

bool **pkgconf_path_relocate**(char **buf*, size_t *buflen*)

Relocates a path, possibly calling normpath() on it.

Parameters

- **buf** (*char**) – The path to relocate.
- **buflen** (*size_t*) – The buffer length the path is contained in.

Returns true on success, false on error

Return type bool

1.8 libpkgconf *personality* module

const pkgconf_cross_personality_t ***pkgconf_cross_personality_default**(void)
Returns the default cross-compile personality.

Not thread safe.

Return type pkgconf_cross_personality_t*

Returns the default cross-compile personality

void **pkgconf_cross_personality_deinit**(pkgconf_cross_personality_t *)
Decrements the count of default cross personality instances.

Not thread safe.

Return type void

pkgconf_cross_personality_t ***pkgconf_cross_personality_find**(const char *triplet*)
Attempts to find a cross-compile personality given a triplet.

Return type pkgconf_cross_personality_t*

Returns the default cross-compile personality

1.9 libpkgconf *pkg* module

The *pkg* module provides dependency resolution services and the overall *.pc* file parsing routines.

pkgconf_pkg_t ***pkgconf_pkg_new_from_file**(const pkgconf_client_t *client*, const char *filename*, FILE *f*, unsigned int *flags*)
Parse a .pc file into a pkgconf_pkg_t object structure.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **filename** (*char**) – The filename of the package file (including full path).
- **f** (*FILE**) – The file object to read from.
- **flags** (*uint*) – The flags to use when parsing.

Returns A `pkgconf_pkg_t` object which contains the package data.

Return type pkgconf_pkg_t *

void **pkgconf_pkg_free**(pkgconf_client_t *client*, pkgconf_pkg_t *pkg*)
Releases all releases for a given `pkgconf_pkg_t` object.

Parameters

- **client** (*pkgconf_client_t**) – The client which owns the `pkgconf_pkg_t` object, *pkg*.
- **pkg** (*pkgconf_pkg_t**) – The package to free.

Returns nothing

pkgconf_pkg_t ***pkgconf_pkg_ref**(const pkgconf_client_t *client*, pkgconf_pkg_t *pkg*)
Adds an additional reference to the package object.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object which owns the package being referenced.
- **pkg** (*pkgconf_pkg_t **) – The package object being referenced.

Returns The package itself with an incremented reference count.

Return type pkgconf_pkg_t *

void **pkgconf_pkg_unref**(pkgconf_client_t *client, pkgconf_pkg_t *pkg)

Releases a reference on the package object. If the reference count is 0, then also free the package.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object which owns the package being dereferenced.
- **pkg** (*pkgconf_pkg_t **) – The package object being dereferenced.

Returns nothing

pkgconf_pkg_t ***pkgconf_scan_all**(pkgconf_client_t *client, void *data, pkgconf_pkg_iteration_func_t *func*)

Iterates over all packages found in the *package directory list*, running `func` on them. If `func` returns true, then stop iteration and return the last iterated package.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to use for dependency resolution.
- **data** (*void**) – An opaque pointer to data to provide the iteration function with.
- **func** (*pkgconf_pkg_iteration_func_t*) – A function which is called for each package to determine if the package matches, always return `false` to iterate over all packages.

Returns A package object reference if one is found by the scan function, else `NULL`.

Return type pkgconf_pkg_t *

pkgconf_pkg_t ***pkgconf_pkg_find**(pkgconf_client_t *client, const char *name*)

Search for a package.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to use for dependency resolution.
- **name** (*char**) – The name of the package *atom* to use for searching.

Returns A package object reference if the package was found, else `NULL`.

Return type pkgconf_pkg_t *

int **pkgconf_compare_version**(const char *a, const char *b)

Compare versions using RPM version comparison rules as described in the LSB.

Parameters

- **a** (*char**) – The first version to compare in the pair.
- **b** (*char**) – The second version to compare in the pair.

Returns -1 if the first version is less than, 0 if both versions are equal, 1 if the second version is less than.

Return type int

pkgconf_pkg_t ***pkgconf_builtin_pkg_get*** (const char **name*)
Looks up a built-in package. The package should not be freed or dereferenced.

Parameters

- **name** (*char**) – An atom corresponding to a built-in package to search for.

Returns the built-in package if present, else NULL.

Return type pkgconf_pkg_t *

const char ***pkgconf_pkg_get_comparator*** (const pkgconf_dependency_t **pkgdep*)
Returns the comparator used in a depgraph dependency node as a string.

Parameters

- **pkgdep** (*pkgconf_dependency_t**) – The depgraph dependency node to return the comparator for.

Returns A string matching the comparator or "???".

Return type char *

pkgconf_pkg_comparator_t ***pkgconf_pkg_comparator_lookup_by_name*** (const char **name*)
Look up the appropriate comparator bytecode in the comparator set (defined in pkg.c, see pkgconf_pkg_comparator_names and pkgconf_pkg_comparator_impls).

Parameters

- **name** (*char**) – The comparator to look up by *name*.

Returns The comparator bytecode if found, else PKGCONF_CMP_ANY.

Return type pkgconf_pkg_comparator_t

pkgconf_pkg_t ***pkgconf_pkg_verify_dependency*** (pkgconf_client_t **client*, pkgconf_dependency_t **pkgdep*, unsigned int **eflags*)
Verify a pkgconf_dependency_t node in the depgraph. If the dependency is solvable, return the appropriate pkgconf_pkg_t object, else NULL.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **pkgdep** (*pkgconf_dependency_t**) – The dependency graph node to solve.
- **eflags** (*uint**) – An optional pointer that, if set, will be populated with an error code from the resolver.

Returns On success, the appropriate pkgconf_pkg_t object to solve the dependency, else NULL.

Return type pkgconf_pkg_t *

unsigned int ***pkgconf_pkg_verify_graph*** (pkgconf_client_t **client*, pkgconf_pkg_t **root*, int *depth*)
Verify the graph dependency nodes are satisfiable by walking the tree using pkgconf_pkg_traverse().

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **root** (*pkgconf_pkg_t**) – The root entry in the package dependency graph which should contain the top-level dependencies to resolve.
- **depth** (*int*) – The maximum allowed depth for dependency resolution.

Returns On success, PKGCONF_PKG_ERRF_OK (0), else an error code.

Return type unsigned int

unsigned int **pkgconf_pkg_traverse**(pkgconf_client_t **client*, pkgconf_pkg_t **root*, pkgconf_pkg_traverse_func_t *func*, void **data*, int *maxdepth*, unsigned int *skip_flags*)

Walk and resolve the dependency graph up to *maxdepth* levels.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **root** (*pkgconf_pkg_t**) – The root of the dependency graph.
- **func** (*pkgconf_pkg_traverse_func_t*) – A traversal function to call for each resolved node in the dependency graph.
- **data** (*void**) – An opaque pointer to data to be passed to the traversal function.
- **maxdepth** (*int*) – The maximum depth to walk the dependency graph for. -1 means infinite recursion.
- **skip_flags** (*uint*) – Skip over dependency nodes containing the specified flags. A setting of 0 skips no dependency nodes.

Returns PKGCONF_PKG_ERRF_OK on success, else an error code.

Return type unsigned int

int **pkgconf_pkg_cflags**(pkgconf_client_t **client*, pkgconf_pkg_t **root*, pkgconf_list_t **list*, int *maxdepth*)

Walks a dependency graph and extracts relevant CFLAGS fragments.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **root** (*pkgconf_pkg_t**) – The root of the dependency graph.
- **list** (*pkgconf_list_t**) – The fragment list to add the extracted CFLAGS fragments to.
- **maxdepth** (*int*) – The maximum allowed depth for dependency resolution. -1 means infinite recursion.

Returns PKGCONF_PKG_ERRF_OK if successful, otherwise an error code.

Return type unsigned int

int **pkgconf_pkg_libs**(pkgconf_client_t **client*, pkgconf_pkg_t **root*, pkgconf_list_t **list*, int *maxdepth*)

Walks a dependency graph and extracts relevant LIBS fragments.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **root** (*pkgconf_pkg_t**) – The root of the dependency graph.
- **list** (*pkgconf_list_t**) – The fragment list to add the extracted LIBS fragments to.
- **maxdepth** (*int*) – The maximum allowed depth for dependency resolution. -1 means infinite recursion.

Returns PKGCONF_PKG_ERRF_OK if successful, otherwise an error code.

Return type unsigned int

1.10 libpkgconf *queue* module

The *queue* module provides an interface that allows easily building a dependency graph from an arbitrary set of dependencies. It also provides support for doing "preflight" checks on the entire dependency graph prior to working with it.

Using the *queue* module functions is the recommended way of working with dependency graphs.

void **pkgconf_queue_push** (pkgconf_list_t **list*, const char **package*)
Pushes a requested dependency onto the dependency resolver's queue.

Parameters

- **list** (*pkgconf_list_t**) – the dependency resolution queue to add the package request to.
- **package** (*char**) – the dependency atom requested

Returns nothing

bool **pkgconf_queue_compile** (pkgconf_client_t **client*, pkgconf_pkg_t **world*, pkgconf_list_t **list*)
Compile a dependency resolution queue into a dependency resolution problem if possible, otherwise report an error.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **world** (*pkgconf_pkg_t**) – The designated root of the dependency graph.
- **list** (*pkgconf_list_t**) – The list of dependency requests to consider.

Returns true if the built dependency resolution problem is consistent, else false

Return type bool

void **pkgconf_queue_free** (pkgconf_list_t **list*)
Release any memory related to a dependency resolution queue.

Parameters

- **list** (*pkgconf_list_t**) – The dependency resolution queue to release.

Returns nothing

void **pkgconf_solution_free** (pkgconf_client_t **client*, pkgconf_pkg_t **world*, int *maxdepth*)
Removes references to package nodes contained in a solution.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **world** (*pkgconf_pkg_t**) – The root for the generated dependency graph. Should have PKGCONF_PKG_PROPF_VIRTUAL flag.

Returns nothing

bool **pkgconf_queue_solve**(pkgconf_client_t **client*, pkgconf_list_t **list*, pkgconf_pkg_t **world*, int *maxdepth*)

Solves and flattens the dependency graph for the supplied dependency list.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **list** (*pkgconf_list_t**) – The list of dependency requests to consider.
- **world** (*pkgconf_pkg_t**) – The root for the generated dependency graph, provided by the caller. Should have PKGCONF_PKG_PROPF_VIRTUAL flag.
- **maxdepth** (*int*) – The maximum allowed depth for the dependency resolver. A depth of -1 means unlimited.

Returns true if the dependency resolver found a solution, otherwise false.

Return type bool

void **pkgconf_queue_apply**(pkgconf_client_t **client*, pkgconf_list_t **list*, pkgconf_queue_apply_func_t *func*, int *maxdepth*, void **data*)

Attempt to compile a dependency resolution queue into a dependency resolution problem, then attempt to solve the problem and feed the solution to a callback function if a complete dependency graph is found.

This function should not be used in new code. Use pkgconf_queue_solve instead.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **list** (*pkgconf_list_t**) – The list of dependency requests to consider.
- **func** (*pkgconf_queue_apply_func_t*) – The callback function to call if a solution is found by the dependency resolver.
- **maxdepth** (*int*) – The maximum allowed depth for the dependency resolver. A depth of -1 means unlimited.
- **data** (*void**) – An opaque pointer which is passed to the callback function.

Returns true if the dependency resolver found a solution, otherwise false.

Return type bool

void **pkgconf_queue_validate**(pkgconf_client_t **client*, pkgconf_list_t **list*, pkgconf_queue_apply_func_t *func*, int *maxdepth*, void **data*)

Attempt to compile a dependency resolution queue into a dependency resolution problem, then attempt to solve the problem.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to use for dependency resolution.
- **list** (*pkgconf_list_t**) – The list of dependency requests to consider.
- **maxdepth** (*int*) – The maximum allowed depth for the dependency resolver. A depth of -1 means unlimited.

Returns true if the dependency resolver found a solution, otherwise false.

Return type bool

1.11 libpkgconf *tuple* module

The *tuple* module provides key-value mappings backed by a linked list. The key-value mapping is mainly used for variable substitution when parsing .pc files.

There are two sets of mappings: a `pkgconf_pkg_t` specific mapping, and a *global* mapping. The *tuple* module provides convenience wrappers for managing the *global* mapping, which is attached to a given client object.

void **pkgconf_tuple_add_global**(pkgconf_client_t **client*, const char **key*, const char **value*)
Defines a global variable, replacing the previous declaration if one was set.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to modify.
- **key** (*char**) – The key for the mapping (variable name).
- **value** (*char**) – The value for the mapped entry.

Returns nothing

void **pkgconf_tuple_find_global**(const pkgconf_client_t **client*, const char **key*)
Looks up a global variable.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to access.
- **key** (*char**) – The key or variable name to look up.

Returns the contents of the variable or `NULL`

Return type char *

void **pkgconf_tuple_free_global**(pkgconf_client_t **client*)
Delete all global variables associated with a pkgconf client object.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to modify.

Returns nothing

void **pkgconf_tuple_define_global**(pkgconf_client_t **client*, const char **kv*)
Parse and define a global variable.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to modify.
- **kv** (*char**) – The variable in the form of `key=value`.

Returns nothing

pkgconf_tuple_t *__pkgconf_tuple_add__(const pkgconf_client_t **client*, pkgconf_list_t **list*, const char **key*, const char **value*, bool *parse*)
Optionally parse and then define a variable.

Parameters

- **client** (*pkgconf_client_t **) – The pkgconf client object to access.
- **list** (*pkgconf_list_t **) – The variable list to add the new variable to.
- **key** (*char**) – The name of the variable being added.
- **value** (*char**) – The value of the variable being added.

- **parse** (*bool*) – Whether or not to parse the value for variable substitution.

Returns a variable object

Return type pkgconf_tuple_t *

char ***pkgconf_tuple_find**(const pkgconf_client_t **client*, pkgconf_list_t **list*, const char **key*)
Look up a variable in a variable list.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to access.
- **list** (*pkgconf_list_t**) – The variable list to search.
- **key** (*char**) – The variable name to search for.

Returns the value of the variable or `NULL`

Return type char *

char ***pkgconf_tuple_parse**(const pkgconf_client_t **client*, pkgconf_list_t **vars*, const char **value*, unsigned int *flags*)
Parse an expression for variable substitution.

Parameters

- **client** (*pkgconf_client_t**) – The pkgconf client object to access.
- **list** (*pkgconf_list_t**) – The variable list to search for variables (along side the global variable list).
- **value** (*char**) – The `key=value` string to parse.
- **flags** (*uint*) – Any flags to consider while parsing.

Returns the variable data with any variables substituted

Return type char *

void **pkgconf_tuple_free_entry**(pkgconf_tuple_t **tuple*, pkgconf_list_t **list*)
Deletes a variable object, removing it from any variable lists and releasing any memory associated with it.

Parameters

- **tuple** (*pkgconf_tuple_t**) – The variable object to release.
- **list** (*pkgconf_list_t**) – The variable list the variable object is attached to.

Returns nothing

void **pkgconf_tuple_free**(pkgconf_list_t **list*)
Deletes a variable list and any variables attached to it.

Parameters

- **list** (*pkgconf_list_t**) – The variable list to delete.

Returns nothing

CHAPTER 2

Indices and tables

- genindex
- search

Index

P