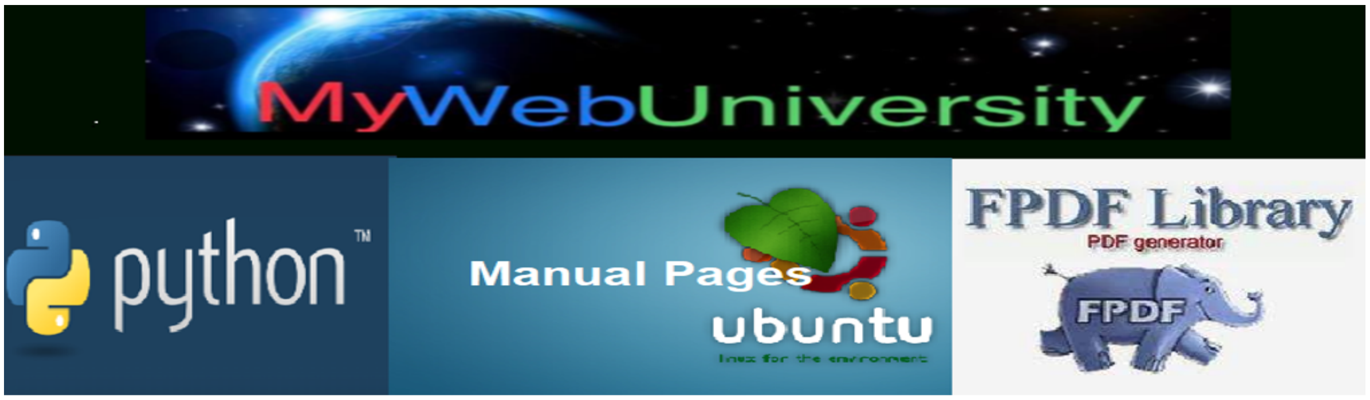

Rocky Enterprise Linux 9.2 Manual Pages on command 'redland.3'

$ man redland.3

```
redland(3)                Redland RDF Application Framework                redland(3)

NAME

       redland - Resource Description Framework (RDF) Library

VERSION

       1.0.16

SYNOPSIS

        #include <redland.h>

DESCRIPTION

       redland is a library providing support for the Resource Description Framework (RDF)

       written in ANSI C with APIs in several other languages.

       This manual page lists most of the redland public API functions but does not claim to be a

       complete summary of the entire API.  For the complete API with full details of the

       function interface, see the HTML API documentation either on the Redland web site at

       <http://librdf.org/> or with the software release in the docs/api directory.

FUNCTIONS

       The functions defined by redland are all defined with the "librdf_" prefix

   class world

       librdf_world* librdf_new_world(void)

       void librdf_free_world(librdf_world* world)

       void librdf_world_open(librdf_world* world)

       void librdf_world_set_error(librdf_world* world, void* user_data, void (*error_fn)(void*

       user_data, const char* msg, ...))

       void librdf_world_set_warning(librdf_world* world, void* user_data, void
```

(*warning_fn)(void* user_data, const char* msg, ...))

void librdf_world_set_digest(librdf_world*, const char* name)

void librdf_world_set_uris_hash(librdf_world* world, librdf_hash* uris_hash)

const char* librdf_world_get_feature(librdf_world* world, librdf_uri* feature)

int librdf_world_set_feature(librdf_world* world, librdf_uri* feature, const char* value)

class iterator

librdf_iterator* librdf_new_iterator(librdf_world* world, void* context, int

(*is_end)(void*), void* (*get_next)(void*), void (*finished)(void*))

void librdf_free_iterator(librdf_iterator*)

int librdf_iterator_end(librdf_iterator* iterator)

int librdf_iterator_finished(librdf_iterator* iterator)

int librdf_iterator_next(librdf_iterator* iterator)

void* librdf_iterator_get_object(librdf_iterator* iterator)

void* librdf_iterator_get_context(librdf_iterator* iterator)

void* librdf_iterator_get_key(librdf_iterator* iterator)

void* librdf_iterator_get_value(librdf_iterator* iterator)

int librdf_iterator_add_map(librdf_iterator* iterator, void* (*fn)(void* context, void*

item), void* context)

void* librdf_iterator_map_remove_duplicate_nodes(void* item, void* user_data)

class digest

void librdf_digest_register_factory(librdf_world* world, const char* name, void (*factory)

(librdf_digest_factory*))

librdf_digest_factory* librdf_get_digest_factory(librdf_world* world, const char* name)

librdf_digest* librdf_new_digest(librdf_world* world, char* name)

librdf_digest* librdf_new_digest_from_factory(librdf_world* world, librdf_digest_factory*

factory)

void librdf_free_digest(librdf_digest* digest)

void librdf_digest_init(librdf_digest* digest)

void librdf_digest_update(librdf_digest* digest, unsigned char* buf, size_t length)

void librdf_digest_final(librdf_digest* digest)

void* librdf_digest_get_digest(librdf_digest* digest)

char* librdf_digest_to_string(librdf_digest* digest)

void librdf_digest_print(librdf_digest* digest, FILE* fh) class uri

librdf_uri* librdf_new_uri(librdf_world* world, const unsigned char * string)

librdf_uri* librdf_new_uri_from_uri(librdf_uri* uri)

librdf_uri* librdf_new_uri_from_uri_local_name(librdf_uri* uri, const unsigned char* local_name)

void librdf_free_uri(librdf_uri* uri)

unsigned char* librdf_uri_as_string(librdf_uri* uri)

unsigned char* librdf_uri_as_counted_string(librdf_uri* uri, size_t* len_p)

librdf_digest* librdf_uri_get_digest(librdf_uri* uri)

void librdf_uri_print>(librdf_uri* uri, FILE* fh)

unsigned char* librdf_uri_to_string(librdf_uri* uri)

unsigned char* librdf_uri_to_counted_string(librdf_uri* uri, size_t* len_p)

int librdf_uri_equals(librdf_uri* first_uri, librdf_uri* second_uri)

int librdf_uri_is_file_uri(librdf_uri* uri)

const char* librdf_uri_to_filename(librdf_uri* uri)

librdf_uri* librdf_new_uri_normalised_to_base(const unsigned char* uri_string, librdf_uri* source_uri, librdf_uri* base_uri)

librdf_uri* librdf_new_uri_relative_to_base(librdf_uri* base_uri, const unsigned char* uri_string)

librdf_uri* librdf_new_uri_from_filename(librdf_world* world, const char* filename)

class node

librdf_node* librdf_new_node(librdf_world* world)

librdf_node* librdf_new_node_from_uri_string(librdf_world* world, const char* string)

librdf_node* librdf_new_node_from_uri(librdf_world* world, librdf_uri* uri)

librdf_node* librdf_new_node_from_uri_local_name(librdf_world* world, librdf_uri* uri, const char* local_name)

librdf_node* librdf_new_node_from_normalised_uri_string(librdf_world* world, const char* uri_string, librdf_uri* source_uri, librdf_uri* base_uri)

librdf_node* librdf_new_node_from_literal(librdf_world* world, const char* string, const char* xml_language, int xml_space, int is_wf_xml)

librdf_node* librdf_new_node_from_typed_literal(librdf_world* world, const unsigned char* string, const char* xml_language, librdf_uri* datatype_uri)

librdf_node* librdf_new_node_from_blank_identifier(librdf_world* world, const unsigned char* identifier)

librdf_node* librdf_new_node_from_node(librdf_node* node)

void librdf_node_init(librdf_world* world, librdf_node* node)

void librdf_free_node(librdf_node* r)

librdf_uri* librdf_node_get_uri(librdf_node* node)

librdf_node_type librdf_node_get_type(librdf_node* node)

unsigned char* librdf_node_get_literal_value(librdf_node* node)

unsigned char* librdf_node_get_literal_value_as_counted_string(librdf_node* node, size_t*

len_p)

char* librdf_node_get_literal_value_as_latin1(librdf_node* node)

char* librdf_node_get_literal_value_language(librdf_node* node)

int librdf_node_get_literal_value_is_wf_xml(librdf_node* node)

librdf_uri* librdf_node_get_literal_value_datatype_uri(librdf_node* node)

int librdf_node_get_li_ordinal(librdf_node* node)

unsigned char* librdf_node_get_blank_identifier(librdf_node* node)

int librdf_node_is_resource(librdf_node* node)

int librdf_node_is_literal(librdf_node* node)

int librdf_node_is_blank(librdf_node* node)

librdf_digest* librdf_node_get_digest(librdf_node* node)

size_t librdf_node_encode(librdf_node* node, unsigned char* buffer, size_t length)

size_t librdf_node_decode(librdf_node* node, unsigned char* buffer, size_t length)

unsigned char* librdf_node_to_string(librdf_node* node)

unsigned char* librdf_node_to_counted_string(librdf_node* node, size_t* len_p)

void librdf_node_print(librdf_node* node, FILE* fh)

int librdf_node_equals(librdf_node* first_node, librdf_node* second_node)

class concepts

The library provides macros for all of the RDF and RDFS concepts - nodes and URIs. For example, "LIBRDF_MS_Alt" for the librdf_node for the rdf:Alt concept and "LIBRDF_MS_Alt_URI" for the librdf_uri for the URI reference of rdf:Alt.

"LIBRDF_URI_RDF_MS" and "LIBRDF_URI_RDF_SCHEMA" provide the librdf_uri objects for the RDF and RDFS namespace URIs. They must be copied using librdf_new_uri_from_uri to be shared correctly.

void librdf_get_concept_by_name(librdf_world* world, int is_ms, const char* name, librdf_uri **uri_p, librdf_node **node_p)

class statement

librdf_statement* librdf_new_statement(librdf_world* world)

librdf_statement* librdf_new_statement_from_statement(librdf_statement* statement)

librdf_statement* librdf_new_statement_from_nodes(librdf_world* world, librdf_node* subject, librdf_node* predicate, librdf_node* object)

void librdf_statement_init(librdf_world* world, librdf_statement* statement)

void librdf_statement_clear(librdf_statement* statement)

void librdf_free_statement(librdf_statement* statement)

librdf_node* librdf_statement_get_subject(librdf_statement* statement)

void librdf_statement_set_subject(librdf_statement* statement, librdf_node* subject)

librdf_node* librdf_statement_get_predicate(librdf_statement* statement)

void librdf_statement_set_predicate(librdf_statement* statement, librdf_node* predicate)

librdf_node* librdf_statement_get_object(librdf_statement* statement)

void librdf_statement_set_object(librdf_statement* statement, librdf_node* object)

int librdf_statement_is_complete(librdf_statement* statement)

char* librdf_statement_to_string(librdf_statement* statement)

void librdf_statement_print(librdf_statement* statement, FILE* fh)

int librdf_statement_equals(librdf_statement* statement1, librdf_statement* statement2)

int librdf_statement_match(librdf_statement* statement, librdf_statement* partial_statement)

size_t librdf_statement_encode(librdf_statement* statement, unsigned char* buffer, size_t length)

size_t librdf_statement_encode_parts(librdf_statement* statement, unsigned char* buffer, size_t length, librdf_statement_part fields)

size_t librdf_statement_decode(librdf_statement* statement, unsigned char* buffer, size_t length)

size_t librdf_statement_decode_parts(librdf_statement* statement, librdf_node** context_node, unsigned char* buffer, size_t length)

class model

librdf_model* librdf_new_model(librdf_world* world, librdf_storage* storage, char* options_string)

librdf_model* librdf_new_model_with_options(librdf_world* world, librdf_storage* storage, librdf_hash* options)

librdf_model* librdf_new_model_from_model(librdf_model* model)

void librdf_free_model(librdf_model* model)

int librdf_model_size(librdf_model* model)

int librdf_model_add(librdf_model* model, librdf_node* subject, librdf_node* predicate,

librdf_node* object)

int librdf_model_add_string_literal_statement(librdf_model* model, librdf_node* subject,

librdf_node* predicate, char* string, char* xml_language, int xml_space, int is_wf_xml)

int librdf_model_add_typed_literal_statement(librdf_model* model, librdf_node* subject,

librdf_node* predicate, const unsigned char* string, char* xml_language, librdf_uri*

datatype_uri)

int librdf_model_add_statement(librdf_model* model, librdf_statement* statement)

int librdf_model_add_statements(librdf_model* model, librdf_stream* statement_stream)

int librdf_model_remove_statement(librdf_model* model, librdf_statement* statement)

int librdf_model_contains_statement(librdf_model* model, librdf_statement* statement)

int librdf_model_has_arc_in(librdf_model* model, librdf_node* node, librdf_node* property)

int librdf_model_has_arc_out(librdf_model* model, librdf_node* node, librdf_node*

property)

librdf_stream* librdf_model_as_stream(librdf_model* model)

librdf_stream* librdf_model_find_statements(librdf_model* model, librdf_statement*

statement)

librdf_stream* librdf_model_find_statements_in_context(librdf_model* model,

librdf_statement* statement, librdf_node* context_node)

librdf_stream* librdf_model_find_statements_with_options(librdf_model* model,

librdf_statement* statement, librdf_node* context_node, librdf_hash* options)

librdf_iterator* librdf_model_get_contexts(librdf_model* model)

librdf_iterator* librdf_model_get_sources(librdf_model* model, librdf_node* arc,

librdf_node* target)

librdf_iterator* librdf_model_get_arcs(librdf_model* model, librdf_node* source,

librdf_node* target)

librdf_iterator* librdf_model_get_targets(librdf_model* model, librdf_node* source,

librdf_node* arc)

librdf_node* librdf_model_get_source(librdf_model* model, librdf_node* arc, librdf_node* target)

librdf_node* librdf_model_get_arc(librdf_model* model, librdf_node* source, librdf_node*

target)

librdf_node* librdf_model_get_target(librdf_model* model, librdf_node* source,

librdf_node* arc)

librdf_iterator* librdf_model_get_arcs_in(librdf_model* model, librdf_node* node)

librdf_iterator* librdf_model_get_arcs_out(librdf_model* model, librdf_node* node)

int librdf_model_add_submodel(librdf_model* model, librdf_model* sub_model)

int librdf_model_remove_submodel(librdf_model* model, librdf_model* sub_model)

void librdf_model_print(librdf_model* model, FILE* fh)

int librdf_model_context_add_statement(librdf_model* model, librdf_node* context,

librdf_statement* statement)

int librdf_model_context_add_statements(librdf_model* model, librdf_node* context,

librdf_stream* stream)

int librdf_model_context_remove_statement(librdf_model* model, librdf_node* context,

librdf_statement* statement)

int librdf_model_context_remove_statements(librdf_model* model, librdf_node* context)

librdf_stream* librdf_model_context_as_stream(librdf_model* model, librdf_node* context)

librdf_stream* librdf_model_query(librdf_model* model, librdf_query* query)

librdf_stream* librdf_model_query_string(librdf_model* model, const char* name,

librdf_uri* uri, const unsigned char* query_string)

void librdf_model_sync(librdf_model* model)

librdf_storage* librdf_model_get_storage(librdf_model* model)

librdf_node* librdf_model_get_feature(librdf_model* model, librdf_uri* feature) =item int

librdf_model_set_feature(librdf_model* model, librdf_uri* feature, librdf_node* value)

class storage

void librdf_storage_register_factory(const char* name, void (*factory)

(librdf_storage_factory*))

librdf_storage* librdf_new_storage(librdf_world* world, char* storage_name, char* name,

char* options_string)

librdf_storage* librdf_new_storage_with_options(librdf_world* world, char* storage_name,

char* name, librdf_hash* options)

librdf_storage* librdf_new_storage_from_storage(librdf_storage* old_storage) librdf_storage* librdf_new_storage_from_factory(librdf_world* world,

librdf_storage_factory* factory, char* name, librdf_hash* options)

void librdf_free_storage(librdf_storage* storage)

int librdf_storage_open(librdf_storage* storage, librdf_model* model)

int librdf_storage_close(librdf_storage* storage)

int librdf_storage_get(librdf_storage* storage, void* key, size_t key_len, void **value,

size_t* value_len, unsigned int flags)

int librdf_storage_size(librdf_storage* storage)

int librdf_storage_add_statement(librdf_storage* storage, librdf_statement* statement)

int librdf_storage_add_statements(librdf_storage* storage, librdf_stream*

statement_stream)

int librdf_storage_remove_statement(librdf_storage* storage, librdf_statement* statement)

int librdf_storage_contains_statement(librdf_storage* storage, librdf_statement*

statement)

librdf_stream* librdf_storage_serialise(librdf_storage* storage)

librdf_stream* librdf_storage_find_statements(librdf_storage* storage, librdf_statement*

statement)

librdf_iterator* librdf_storage_get_sources(librdf_storage* storage, librdf_node* arc,

librdf_node* target)

librdf_iterator* librdf_storage_get_arcs(librdf_storage* storage, librdf_node* source,

librdf_node* target)

librdf_iterator* librdf_storage_get_targets(librdf_storage* storage, librdf_node* source,

librdf_node* arc)

librdf_iterator* librdf_storage_get_arcs_in(librdf_storage* storage, librdf_node* node)

librdf_iterator* librdf_storage_get_arcs_out(librdf_storage* storage, librdf_node* node)

int librdf_storage_has_arc_in(librdf_storage* storage, librdf_node* node, librdf_node*

property)

int librdf_storage_has_arc_out(librdf_storage* storage, librdf_node* node, librdf_node*

property)

int librdf_storage_context_add_statement(librdf_storage* storage, librdf_node* context,

librdf_statement* statement)

int librdf_storage_context_add_statements(librdf_storage* storage, librdf_node* context,

librdf_stream* stream)

int librdf_storage_context_remove_statement(librdf_storage* storage, librdf_node* context, librdf_statement* statement)

int librdf_storage_context_remove_statements(librdf_storage* storage, librdf_node* context)

librdf_stream* librdf_storage_context_as_stream(librdf_storage* storage, librdf_node* context)

int librdf_storage_supports_query(librdf_storage* storage, librdf_query* query)

librdf_stream* librdf_storage_query(librdf_storage* storage, librdf_query* query)

void librdf_storage_sync(librdf_storage* storage)

class parser

void librdf_parser_register_factory(librdf_world* world, const char* name, const char* mime_type, const char* uri_string, void (*factory) (librdf_parser_factory*))

librdf_parser* librdf_new_parser(librdf_world* world, const char* name, const char* mime_type, librdf_uri* type_uri)

librdf_parser* librdf_new_parser_from_factory(librdf_world* world, librdf_parser_factory* factory)

void librdf_free_parser(librdf_parser* parser)

librdf_stream* librdf_parser_parse_as_stream(librdf_parser* parser, librdf_uri* uri, librdf_uri* base_uri)

int librdf_parser_parse_into_model(librdf_parser* parser, librdf_uri* uri, librdf_uri* base_uri, librdf_model* model)

librdf_stream* librdf_parser_parse_string_as_stream(librdf_parser* parser, const unsigned char* string, librdf_uri* base_uri)

int librdf_parser_parse_string_into_model(librdf_parser* parser, const unsigned char* string, librdf_uri* base_uri, librdf_model* model)

void librdf_parser_set_error(librdf_parser* parser, void* user_data, void (*error_fn)(void* user_data, const char* msg, ...))

void librdf_parser_set_warning(librdf_parser* parser, void* user_data, void (*warning_fn)(void* user_data, const char* msg, ...))

librdf_node* librdf_parser_get_feature(librdf_parser* parser, librdf_uri* feature)

int librdf_parser_set_feature(librdf_parser* parser, librdf_uri* feature, librdf_node* value)

class serializer

librdf_serializer* librdf_new_serializer(librdf_world* world, const char *name, const char *mime_type, librdf_uri *type_uri)

librdf_serializer* librdf_new_serializer_from_factory(librdf_world* world, librdf_serializer_factory *factory)

void librdf_free_serializer(librdf_serializer *serializer)

int librdf_serializer_serialize_model(librdf_serializer* serializer, FILE* handle, librdf_uri* base_uri, librdf_model* model)

int librdf_serializer_serialize_model_to_file(librdf_serializer* serializer, const char *name, librdf_uri* base_uri, librdf_model* model)

void librdf_serializer_set_error(librdf_serializer* serializer, void *user_data, void (*error_fn)(void *user_data, const char *msg, ...))

void librdf_serializer_set_warning(librdf_serializer* serializer, void *user_data, void (*warning_fn)(void *user_data, const char *msg, ...))

librdf_node* librdf_serializer_get_feature(librdf_serializer* serializer, librdf_uri* feature)

int librdf_serializer_set_feature(librdf_serializer* serializer, librdf_uri* feature, librdf_node* value)b

int librdf_serializer_set_namespace(librdf_serializer* serializer, librdf_uri* uri, const char* prefix)

class stream

librdf_stream* librdf_new_stream(librdf_world* world, void* context, int (*end_of_stream)(void*), librdf_statement* (*next_statement)(void*), void (*finished)(void*))

librdf_stream* librdf_new_stream_from_node_iterator(librdf_iterator* iterator, librdf_statement* statement, librdf_statement_part field)

void librdf_free_stream(librdf_stream* stream)

int librdf_stream_end(librdf_stream* stream)

int librdf_stream_next(librdf_stream* stream)

librdf_statement* librdf_stream_get_object(librdf_stream* stream)

void* librdf_stream_get_context(librdf_stream* stream)

void librdf_stream_set_map(librdf_stream* stream, librdf_statement* (*map)(void* context, librdf_statement* statement), void* map_context)

void librdf_stream_print(librdf_stream* stream, FILE* fh)

EXAMPLES

```
#include <redland.h>

librdf_storage *storage;

librdf_model* model;

librdf_statement* statement;

librdf_world* world

world=librdf_new_world();

librdf_world_open(world);

storage=librdf_new_storage(world, "hashes", "test", "hash-type='bdb',dir='.'");

model=librdf_new_model(world, storage, NULL);

                                    statement=librdf_new_statement_from_nodes(world,     librdf_new_node_from_uri_string(world,
"http://purl.org/net/dajobe/"),             librdf_new_node_from_uri_string(world,             "http://purl.org/dc/elements/1.1/creator"),
librdf_new_node_from_literal(world, "Dave Beckett", NULL, 0));

librdf_model_add_statement(model, statement);

librdf_free_statement(statement);

librdf_model_print(model, stdout);

librdf_free_model(model);

librdf_free_storage(storage);

librdf_free_world(world);
```

SEE ALSO

libraptor(3), libxml(4).

HISTORY

The redland RDF library was created by Dave Beckett in June 2000.

AUTHOR

Dave Beckett <http://purl.org/net/dajobe/>,