NAME

snl_init, **snl_free**, **snl_read_message**, **snl_send**, **snl_get_seq**, **snl_allocz**, **snl_clear_lb**, **snl_parse_nlmsg**, **snl_parse_header**, **snl_parse_attrs**, **snl_parse_attrs_raw**, **snl_attr_get_flag**, **snl_attr_get_ip**, **snl_attr_get_uint16**, **snl_attr_get_uint32**, **snl_attr_get_string**, **snl_attr_get_stringn**, **snl_attr_get_nla**, **snl_field_get_uint8**, **snl_field_get_uint16**, **snl_field_get_uint32** **-** simple netlink library

SYNOPSIS

#include <netlink/netlink_snl.h>
#include <netlink/netlink_snl_route.h>

bool
snl_init(*struct snl_state *ss*, *int netlink_family*);

snl_free(*struct snl_state *ss*);

*struct nlmsghdr **
snl_read_message(*struct snl_state *ss*);

bool
snl_send(*struct snl_state *ss*, *void *data*, *int sz*);

uint32_t
snl_get_seq(*struct snl_state *ss*);

*void **
snl_allocz(*struct snl_state *ss*, *int len*);

snl_clear_lb(*struct snl_state *ss*);

bool
snl_parse_nlmsg(*struct snl_state *ss*, *struct nlmsghdr *hdr*, *const struct snl_hdr_parser *ps*, *void *target*);

bool
snl_parse_header(*struct snl_state *ss*, *void *hdr*, *int len*, *const struct snl_hdr_parser *ps*, *int pslen*, *void *target*);

bool
snl_parse_attrs(*struct snl_state *ss*, *struct nlmsghdr *hdr*, *int hdrlen*, *const struct snl_attr_parser *ps*, *int pslen*, *void *target*);

bool
snl_parse_attrs_raw(*struct snl_state *ss*, *struct nlattr *nla_head*, *int len*, *const struct snl_attr_parser *ps*, *int pslen*, *void *target*);

bool
snl_attr_get_flag(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_uint8(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_uint16(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_uint32(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_uint64(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_string(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_stringn(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_nla(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_ip(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

bool
snl_attr_get_ipvia(*struct snl_state *ss*, *struct nlattr *nla*, *void *target*);

DESCRIPTION

The snl(3) library provides an easy way of sending and receiving Netlink messages, taking care of serialisation and deserialisation.

INITIALISATION

Call **snl_init**() with a pointer to the struct snl_state and the desired Netlink family to initialise the library instance. To free the library instance, call **snl_free**().

The library functions are NOT multithread-safe. If multithreading is desired, consider initializing an instance per thread.

MEMORY ALLOCATION

The library uses pre-allocated extendable memory buffers to handle message parsing. The typical usage pattern is to allocate the necessary data structures during the message parsing or writing process via **snl_allocz**() and free all allocated data at once using **snl_clear_lb**() after handling the message.

COMPOSING AND SENDING MESSAGES

The library does not currently offer any wrappers for writing netlink messages. Simple request messages can be composed by filling in all needed fields directly. Example for constructing an interface dump request:

```
struct {
        struct nlmsghdr hdr;
        struct ifinfomsg ifmsg;
} msg = {
        .hdr.nlmsg_type = RTM_GETLINK,
        .hdr.nlmsg_flags = NLM_F_DUMP | NLM_F_REQUEST,
        .hdr.nlmsg_seq = snl_get_seq(ss),
};
msg.hdr.nlmsg_len = sizeof(msg);
```

snl_get_seq() can be used to generate a unique message number. To send the resulting message, **snl_send**() can be used.

RECEIVING AND PARSING MESSAGES

To receive a message, use **snl_read_message**(). Currently, this call is blocking.

The library provides an easy way to convert the message to the pre-defined C structure. For each message type, one needs to define rules, converting the protocol header fields and the desired attributes to the specified structure. It can be accomplished by using message parsers. Each message parser consists of an array of attribute getters and an array of header field getters. The former array needs to be sorted by the attribute type. There is a **SNL_VERIFY_PARSERS**() macro to check if the order is correct. **SNL_DECLARE_PARSER**(*parser_name*, *family header type*, *struct snl_field_parser[]*, *struct snl_attr_parser[]*) can be used to create a new parser. **SNL_DECLARE_ATTR_PARSER**(*parser_name*, *struct snl_field_parser[]*) can be used to create an attribute-only message parser.

Each attribute getter needs to be embedded in the following structure:

```
typedef bool snl_parse_attr_f(struct snl_state *ss, struct nlattr *attr, const void *arg, void *target);
struct snl_attr_parser {
        uint16_t          type;     /* Attribute type */
        uint16_t          off;      /* field offset in the target structure */
        snl_parse_attr_f  *cb;      /* getter function to call */
        const void                  *arg;     /* getter function custom argument */
};
```

The generic attribute getter has the following signature: *bool* **snl_attr_get_<type>**(*struct snl_state *ss*, *struct nlattr *nla*, *const void *arg*, *void *target*). nla contains the pointer of the attribute to use as the datasource. The target field is the pointer to the field in the target structure. It is up to the getter to know the type of the target field. The getter must check the input attribute and return false if the attribute is not formed correctly. Otherwise, the getter fetches the attribute value and stores it in the target, then returns true. It is possible to use **snl_allocz**() to create the desired data structure . A number of predefined getters for the common data types exist. **snl_attr_get_flag**() converts a flag-type attribute to an uint8_t value of 1 or 0, depending on the attribute presence. **snl_attr_get_uint8**() stores a uint8_t type attribute into the uint8_t target field. **snl_attr_get_uint16**() stores a uint16_t type attribute into the uint16_t target field. **snl_attr_get_uint32**() stores a uint32_t type attribute into the uint32_t target field. **snl_attr_get_uint64**() stores a uint64_t type attribute into the uint64_t target field. **snl_attr_get_ip**() and **snl_attr_get_ipvia**() stores a pointer to the sockaddr structure with the IPv4/IPv6 address contained in the attribute. Sockaddr is allocated using **snl_allocz**(). **snl_attr_get_string**() stores a pointer to the NULL-terminated string. The string itself is allocated using **snl_allocz**(). **snl_attr_get_nla**() stores a pointer to the specified attribute. **snl_attr_get_stringn**() stores a pointer to the non-NULL-terminated string.

Similarly, each family header getter needs to be embedded in the following structure:

```
typedef void snl_parse_field_f(struct snl_state *ss, void *hdr, void *target);
struct snl_field_parser {
        uint16_t          off_in;   /* field offset in the input structure */
        uint16_t        off_out;/* field offset in the target structure */
        snl_parse_field_f   *cb;    /* getter function to call */
};
```

The generic field getter has the following signature: *void* snl_field_get_<type> "struct snl_state *ss" "void *src" "void *target" . A number of pre-defined getters for the common data types exist. **snl_field_get_uint8**() fetches an uint8_t value and stores it in the target. **snl_field_get_uint16**() fetches an uint8_t value and stores it in the target. **snl_field_get_uint32**() fetches an uint32_t value and stores it in the target.

EXAMPLES

The following example demonstrates how to list all system interfaces using netlink.

```
#include <stdio.h>

#include <netlink/netlink.h>
#include <netlink/netlink_route.h>
#include "netlink/netlink_snl.h"
#include "netlink/netlink_snl_route.h"

struct nl_parsed_link {
        uint32_t        ifi_index;
        uint32_t        ifla_mtu;
        char                    *ifla_ifname;
};

#define _IN(_field)     offsetof(struct ifinfomsg, _field)
#define _OUT(_field)    offsetof(struct nl_parsed_link, _field)
static const struct snl_attr_parser ap_link[] = {
        { .type = IFLA_IFNAME, .off = _OUT(ifla_ifname), .cb = snl_attr_get_string },
        { .type = IFLA_MTU, .off = _OUT(ifla_mtu), .cb = snl_attr_get_uint32 },
};
static const struct snl_field_parser fp_link[] = {
        {.off_in = _IN(ifi_index), .off_out = _OUT(ifi_index), .cb = snl_field_get_uint32 },
};
#undef _IN
#undef _OUT
SNL_DECLARE_PARSER(link_parser, struct ifinfomsg, fp_link, ap_link);


int
main(int ac, char *argv[])
{
        struct snl_state ss;

        if (!snl_init(&ss, NETLINK_ROUTE))
                return (1);

        struct {
                struct nlmsghdr hdr;
                struct ifinfomsg ifmsg;
        } msg = {
                .hdr.nlmsg_type = RTM_GETLINK,
                .hdr.nlmsg_flags = NLM_F_DUMP | NLM_F_REQUEST,
                .hdr.nlmsg_seq = snl_get_seq(&ss),
        };
        msg.hdr.nlmsg_len = sizeof(msg);

        if (!snl_send(&ss, &msg, sizeof(msg))) {
                snl_free(&ss);
                return (1);
        }

        struct nlmsghdr *hdr;
        while ((hdr = snl_read_message(&ss)) != NULL && hdr->nlmsg_type != NLMSG_DONE) {
                if (hdr->nlmsg_seq != msg.hdr.nlmsg_seq)
                        break;

                struct nl_parsed_link link = {};
                if (!snl_parse_nlmsg(&ss, hdr, &link_parser, &link))
                        continue;
                printf("Link#%u %s mtu %u0, link.ifi_index, link.ifla_ifname, link.ifla_mtu);
        }

        return (0);
}
```

SEE ALSO

genetlink(4), netlink(4), and rtnetlink(4)

HISTORY

The SNL library appeared in FreeBSD 13.2.

AUTHORS

This library was implemented by Alexander Chernikov <*melifaro@FreeBSD.org*>.