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

$ man rtnetlink.3

RTNETLINK(3) Linux Programmer's Manual RTNETLINK(3)

NAME

rtnetlink - macros to manipulate rtnetlink messages

SYNOPSIS

```
#include <asm/types.h>

#include <linux/netlink.h>

#include <linux/rtnetlink.h>

#include <sys/socket.h>

rtnetlink_socket = socket(AF_NETLINK, int socket_type, NETLINK_ROUTE);

int RTA_OK(struct rtattr *rta, int rtabuflen);

void *RTA_DATA(struct rtattr *rta);

unsigned int RTA_PAYLOAD(struct rtattr *rta);

struct rtattr *RTA_NEXT(struct rtattr *rta, unsigned int rtabuflen);

unsigned int RTA_LENGTH(unsigned int length);

unsigned int RTA_SPACE(unsigned int length);
```

DESCRIPTION

All rtnetlink(7) messages consist of a netlink(7) message header and appended attributes.

The attributes should be manipulated only using the macros provided here.

RTA_OK(rta, attrlen) returns true if rta points to a valid routing attribute; attrlen is the running length of the attribute buffer. When not true then you must assume there are no more attributes in the message, even if attrlen is nonzero.

RTA_DATA(rta) returns a pointer to the start of this attribute's data.

RTA_PAYLOAD(rta) returns the length of this attribute's data.

RTA_NEXT(rta, attrlen) gets the next attribute after rta. Calling this macro will update attrlen. You should use RTA_OK to check the validity of the returned pointer.

RTA_LENGTH(len) returns the length which is required for len bytes of data plus the header.

RTA_SPACE(len) returns the amount of space which will be needed in a message with len bytes of data.

CONFORMING TO

These macros are nonstandard Linux extensions.

BUGS

This manual page is incomplete.

EXAMPLES

Creating a rtnetlink message to set the MTU of a device:

```
#include <linux/rtnetlink.h>
...
struct {
    struct nlmsghdr  nh;
    struct ifinfomsg if;
    char             attrbuf[512];
} req;
struct rtattr *rta;
unsigned int mtu = 1000;
int rtnetlink_sk = socket(AF_NETLINK, SOCK_DGRAM, NETLINK_ROUTE);
memset(&req, 0, sizeof(req));
req.nh.nlmsg_len = NLMSG_LENGTH(sizeof(req.if));
req.nh.nlmsg_flags = NLM_F_REQUEST;
req.nh.nlmsg_type = RTM_NEWLINK;
req.if.ifi_family = AF_UNSPEC;
req.if.ifi_index = INTERFACE_INDEX;
req.if.ifi_change = 0xffffffff; /* ??? */
rta = (struct rtattr *)(((char *) &req) +
                         NLMSG_ALIGN(req.nh.nlmsg_len));
rta->rta_type = IFLA_MTU;
rta->rta_len = RTA_LENGTH(sizeof(mtu));
```

```
    req.nh.nlmsg_len = NLMSG_ALIGN(req.nh.nlmsg_len) +
                                   RTA_LENGTH(sizeof(mtu));
    memcpy(RTA_DATA(rta), &mtu, sizeof(mtu));
    send(rtnetlink_sk, &req, req.nh.nlmsg_len, 0);
```

SEE ALSO

netlink(3), netlink(7), rtnetlink(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.