NAME

getprotoent_r, getprotobyname_r, getprotobynumber_r − get protocol entry (reentrant)

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

```
#include <netdb.h>

int getprotoent_r(struct protoent *restrict result_buf,
            char buf[restrict .buflen], size_t buflen,
            struct protoent **restrict result);
int getprotobyname_r(const char *restrict name,
            struct protoent *restrict result_buf,
            char buf[restrict .buflen], size_t buflen,
            struct protoent **restrict result);
int getprotobynumber_r(int proto,
            struct protoent *restrict result_buf,
            char buf[restrict .buflen], size_t buflen,
            struct protoent **restrict result);
```

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

getprotoent_r(), **getprotobyname_r**(), **getprotobynumber_r**():
  Since glibc 2.19:
    _DEFAULT_SOURCE
  glibc 2.19 and earlier:
    _BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

The **getprotoent_r**(), **getprotobyname_r**(), and **getprotobynumber_r**() functions are the reentrant equivalents of, respectively, **getprotoent**(3), **getprotobyname**(3), and **getprotobynumber**(3). They differ in the way that the *protoent* structure is returned, and in the function calling signature and return value. This manual page describes just the differences from the nonreentrant functions.

Instead of returning a pointer to a statically allocated *protoent* structure as the function result, these functions copy the structure into the location pointed to by *result_buf*.

The *buf* array is used to store the string fields pointed to by the returned *protoent* structure. (The nonreentrant functions allocate these strings in static storage.) The size of this array is specified in *buflen*. If *buf* is too small, the call fails with the error **ERANGE**, and the caller must try again with a larger buffer. (A buffer of length 1024 bytes should be sufficient for most applications.)

If the function call successfully obtains a protocol record, then *result is set pointing to *result_buf*; otherwise, *result is set to NULL.

RETURN VALUE

On success, these functions return 0. On error, they return one of the positive error numbers listed in ERRORS.

On error, record not found (**getprotobyname_r**(), **getprotobynumber_r**()), or end of input (**getprotoent_r**()) *result* is set to NULL.

ERRORS

ENOENT
(**getprotoent_r**()) No more records in database.

ERANGE
buf is too small. Try again with a larger buffer (and increased *buflen*).

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	**Attribute**	**Value**
getprotoent_r(), **getprotobyname_r**(), **getprotobynumber_r**()	Thread safety	MT-Safe locale

STANDARDS

These functions are GNU extensions. Functions with similar names exist on some other systems, though typically with different calling signatures.

EXAMPLES

The program below uses **getprotobyname_r**() to retrieve the protocol record for the protocol named in its first command-line argument. If a second (integer) command-line argument is supplied, it is used as the initial value for *buflen*; if **getprotobyname_r**() fails with the error **ERANGE**, the program retries with larger buffer sizes. The following shell session shows a couple of sample runs:

```
$ ./a.out tcp 1
ERANGE! Retrying with larger buffer
getprotobyname_r() returned: 0 (success)  (buflen=78)
p_name=tcp; p_proto=6; aliases=TCP
$ ./a.out xxx 1
ERANGE! Retrying with larger buffer
getprotobyname_r() returned: 0 (success)  (buflen=100)
Call failed/record not found
```

Program source

```
#define _GNU_SOURCE
#include <ctype.h>
#include <errno.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_BUF 10000

int
main(int argc, char *argv[])
{
    int buflen, erange_cnt, s;
    struct protoent result_buf;
    struct protoent *result;
    char buf[MAX_BUF];

    if (argc < 2) {
        printf("Usage: %s proto−name [buflen]\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    buflen = 1024;
    if (argc > 2)
        buflen = atoi(argv[2]);

    if (buflen > MAX_BUF) {
        printf("Exceeded buffer limit (%d)\n", MAX_BUF);
        exit(EXIT_FAILURE);
    }
```

```
        erange_cnt = 0;
        do {
            s = getprotobyname_r(argv[1], &result_buf,
                                 buf, buflen, &result);
            if (s == ERANGE) {
                if (erange_cnt == 0)
                    printf("ERANGE! Retrying with larger buffer\n");
                erange_cnt++;

                /* Increment a byte at a time so we can see exactly
                   what size buffer was required. */

                buflen++;

                if (buflen > MAX_BUF) {
                    printf("Exceeded buffer limit (%d)\n", MAX_BUF);
                    exit(EXIT_FAILURE);
                }
            }
        } while (s == ERANGE);

        printf("getprotobyname_r() returned: %s  (buflen=%d)\n",
               (s == 0) ? "0 (success)" : (s == ENOENT) ? "ENOENT" :
               strerror(s), buflen);

        if (s != 0 || result == NULL) {
            printf("Call failed/record not found\n");
            exit(EXIT_FAILURE);
        }

        printf("p_name=%s; p_proto=%d; aliases=",
               result_buf.p_name, result_buf.p_proto);
        for (char **p = result_buf.p_aliases; *p != NULL; p++)
            printf("%s ", *p);
        printf("\n");

        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

getprotoent(3), **protocols**(5)