Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'getservbyport_r.3' command

$ man getservbyport_r.3

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

NAME

getservent_r, getservbyname_r, getservbyport_r - get service entry (reentrant)

SYNOPSIS

```
#include <netdb.h>
int getservent_r(struct servent *result_buf, char *buf,
                 size_t buflen, struct servent **result);
int getservbyname_r(const char *name, const char *proto,
                 struct servent *result_buf, char *buf,
                 size_t buflen, struct servent **result);
int getservbyport_r(int port, const char *proto,
                 struct servent *result_buf, char *buf,
                 size_t buflen, struct servent **result);
```

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

getservent_r(), getservbyname_r(), getservbyport_r():

Since glibc 2.19:

_DEFAULT_SOURCE

Glibc 2.19 and earlier:

_BSD_SOURCE || _SVID_SOURCE

DESCRIPTION

The getservent_r(), getservbyname_r(), and getservbyport_r() functions are the reentrant equivalents of, respectively, getservent(3), get?

servbyname(3), and getservbyport(3). They differ in the way that the servent 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 servent struc? ture 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 re? turned servent structure. (The nonreentrant functions allocate these strings in static storage.) The size of this array is specified in bu? flen. 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 service record, then *re? sult 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 (getservbyname_r(), getservbyport_r()), or end of input (getservent_r()) result is set to NULL.

ERRORS

ENOENT (getservent_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 at? tributes(7).

Interface	Attribute	Value
getservent_r(),	Thread safety	MT-Safe locale
getservbyname_r(),		
getservbyport_r()		

??

CONFORMING TO

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

EXAMPLES

The program below uses getservbyport_r() to retrieve the service record for the port and protocol named in its first command-line argument. If a third (integer) command-line argument is supplied, it is used as the initial value for buflen; if getservbyport_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 7 tcp 1
ERANGE! Retrying with larger buffer
getservbyport_r() returned: 0 (success)  (buflen=87)
s_name=echo; s_proto=tcp; s_port=7; aliases=
$ ./a.out 77777 tcp
getservbyport_r() returned: 0 (success)  (buflen=1024)
Call failed/record not found
```

Program source

```
#define _GNU_SOURCE
#include <ctype.h>
#include <netdb.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#define MAX_BUF 10000
int
main(int argc, char *argv[])
{
    int buflen, erange_cnt, port, s;
    struct servent result_buf;
```

```
struct servent *result;
char buf[MAX_BUF];
char *protop;
if (argc < 3) {
    printf("Usage: %s port-num proto-name [buflen]\n", argv[0]);
    exit(EXIT_FAILURE);
}
port = htons(atoi(argv[1]));
protop = (strcmp(argv[2], "null") == 0 ||
          strcmp(argv[2], "NULL") == 0) ?  NULL : argv[2];
buflen = 1024;
if (argc > 3)
    buflen = atoi(argv[3]);
if (buflen > MAX_BUF) {
    printf("Exceeded buffer limit (%d)\n", MAX_BUF);
    exit(EXIT_FAILURE);
}
erange_cnt = 0;
do {
    s = getservbyport_r(port, protop, &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("getservbyport_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("s_name=%s; s_proto=%s; s_port=%d; aliases=",
                result_buf.s_name, result_buf.s_proto,
                ntohs(result_buf.s_port));

        for (char **p = result_buf.s_aliases; *p != NULL; p++)
            printf("%s ", *p);

        printf("\n");

        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

getservent(3), services(5)

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/.