## NAME

getgrent_r, fgetgrent_r – get group file entry reentrantly

## SYNOPSIS

**#include <grp.h>**

**int getgrent_r(struct group ****gbuf***, char ****buf***,**
**size_t** *buflen***, struct group *****gbufp***);**

**int fgetgrent_r(FILE ****stream***, struct group ****gbuf***, char ****buf***,**
**size_t** *buflen***, struct group *****gbufp***);**

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

**getgrent_r**(): _GNU_SOURCE
**fgetgrent_r**():
Since glibc 2.19:
_DEFAULT_SOURCE
Glibc 2.19 and earlier:
_SVID_SOURCE

## DESCRIPTION

The functions **getgrent_r**() and **fgetgrent_r**() are the reentrant versions of **getgrent**(3) and **fgetgrent**(3). The former reads the next group entry from the stream initialized by **setgrent**(3). The latter reads the next group entry from *stream*.

The *group* structure is defined in *<grp.h>* as follows:

```
struct group {
    char   *gr_name;        /* group name */
    char   *gr_passwd;      /* group password */
    gid_t   gr_gid;         /* group ID */
    char  **gr_mem;         /* NULL-terminated array of pointers
                               to names of group members */
};
```

For more information about the fields of this structure, see **group**(5).

The nonreentrant functions return a pointer to static storage, where this static storage contains further pointers to group name, password and members. The reentrant functions described here return all of that in caller-provided buffers. First of all there is the buffer *gbuf* that can hold a *struct group*. And next the buffer *buf* of size *buflen* that can hold additional strings. The result of these functions, the *struct group* read from the stream, is stored in the provided buffer **gbuf*, and a pointer to this *struct group* is returned in **gbufp*.

## RETURN VALUE

On success, these functions return 0 and **gbufp* is a pointer to the *struct group*. On error, these functions return an error value and **gbufp* is NULL.

## ERRORS

**ENOENT**
No more entries.

**ERANGE**
Insufficient buffer space supplied. Try again with larger buffer.

## ATTRIBUTES

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

| **Interface** | **Attribute** | **Value** |
|---|---|---|
| **getgrent_r**() | Thread safety | MT-Unsafe race:grent locale |
| **fgetgrent_r**() | Thread safety | MT-Safe |

In the above table, *grent* in *race:grent* signifies that if any of the functions **setgrent**(), **getgrent**(), **endgrent**(), or **getgrent_r**() are used in parallel in different threads of a program, then data races could occur.

## CONFORMING TO

These functions are GNU extensions, done in a style resembling the POSIX version of functions like **getpwnam_r**(3). Other systems use the prototype

```
struct group *getgrent_r(struct group *grp, char *buf,
                         int buflen);
```

or, better,

```
int getgrent_r(struct group *grp, char *buf, int buflen,
               FILE **gr_fp);
```

## NOTES

The function **getgrent_r**() is not really reentrant since it shares the reading position in the stream with all other threads.

## EXAMPLE

```
#define _GNU_SOURCE
#include <grp.h>
#include <stdio.h>
#include <stdlib.h>
#define BUFLEN 4096

int
main(void)
{
    struct group grp, *grpp;
    char buf[BUFLEN];
    int i;

    setgrent();
    while (1) {
        i = getgrent_r(&grp, buf, BUFLEN, &grpp);
        if (i)
            break;
        printf("%s (%d):", grpp->gr_name, grpp->gr_gid);
        for (i = 0; ; i++) {
            if (grpp->gr_mem[i] == NULL)
                break;
            printf(" %s", grpp->gr_mem[i]);
        }
        printf("\n");
    }
    endgrent();
    exit(EXIT_SUCCESS);
}
```

## SEE ALSO

**fgetgrent**(3), **getgrent**(3), **getgrgid**(3), **getgrnam**(3), **putgrent**(3), **group**(5)

## COLOPHON

This page is part of release 5.05 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/.