NAME

getgrouplist − get list of groups to which a user belongs

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <grp.h>

int getgrouplist(const char ****user*, gid_t** *group***,**
gid_t ****groups*, int ****ngroups***);**

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

getgrouplist():
Since glibc 2.19:
_DEFAULT_SOURCE
glibc 2.19 and earlier:
_BSD_SOURCE

DESCRIPTION

The **getgrouplist**() function scans the group database (see **group**(5)) to obtain the list of groups that *user* belongs to. Up to**ngroups* of these groups are returned in the array *groups*.

If it was not among the groups defined for *user* in the group database, then *group* is included in the list of groups returned by **getgrouplist**(); typically this argument is specified as the group ID from the password record for *user*.

The *ngroups* argument is a value-result argument: on return it always contains the number of groups found for *user*, including *group*; this value may be greater than the number of groups stored in *groups*.

RETURN VALUE

If the number of groups of which *user* is a member is less than or equal to **ngroups*, then the value **ngroups* is returned.

If the user is a member of more than**ngroups* groups, then **getgrouplist**() returns −1. In this case, the value returned in**ngroups* can be used to resize the buffer passed to a further call to **getgrouplist**().

VERSIONS

This function is present since glibc 2.2.4.

ATTRIBUTES

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

Interface	Attribute	Value
getgrouplist()	Thread safety	MT-Safe locale

STANDARDS

This function is nonstandard; it appears on most BSDs.

BUGS

Before glibc 2.3.3, the implementation of this function contains a buffer-overrun bug: it returns the complete list of groups for *user* in the array *groups*, even when the number of groups exceeds **ngroups*.

EXAMPLES

The program below displays the group list for the user named in its first command-line argument. The second command-line argument specifies the *ngroups* value to be supplied to **getgrouplist**(). The following shell session shows examples of the use of this program:

```
$ ./a.out cecilia 0
getgrouplist() returned -1; ngroups = 3
$ ./a.out cecilia 3
ngroups = 3
```

```
        16 (dialout)
        33 (video)
        100 (users)
```

Program source

```
#include <grp.h>
#include <pwd.h>
#include <stdio.h>
#include <stdlib.h>

int
main(int argc, char *argv[])
{
    int ngroups;
    struct passwd *pw;
    struct group *gr;
    gid_t *groups;

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <user> <ngroups>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    ngroups = atoi(argv[2]);

    groups = malloc(sizeof(*groups) * ngroups);
    if (groups == NULL) {
        perror("malloc");
        exit(EXIT_FAILURE);
    }

    /* Fetch passwd structure (contains first group ID for user). */

    pw = getpwnam(argv[1]);
    if (pw == NULL) {
        perror("getpwnam");
        exit(EXIT_SUCCESS);
    }

    /* Retrieve group list. */

    if (getgrouplist(argv[1], pw->pw_gid, groups, &ngroups) == -1) {
        fprintf(stderr, "getgrouplist() returned -1; ngroups = %d\n",
                ngroups);
        exit(EXIT_FAILURE);
    }

    /* Display list of retrieved groups, along with group names. */

    fprintf(stderr, "ngroups = %d\n", ngroups);
    for (size_t j = 0; j < ngroups; j++) {
        printf("%d", groups[j]);
        gr = getgrgid(groups[j]);
        if (gr != NULL)
```

```
                printf(" (%s)", gr->gr_name);
            printf("\n");
        }

        exit(EXIT_SUCCESS);
    }
```

SEE ALSO

getgroups(2), **setgroups**(2), **getgrent**(3), **group_member**(3), **group**(5), **passwd**(5)