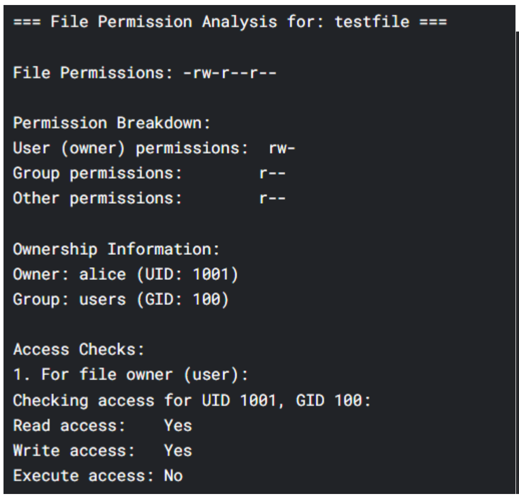
EXP 40: Illustrate the various File Access Permission and different types users in Linux.

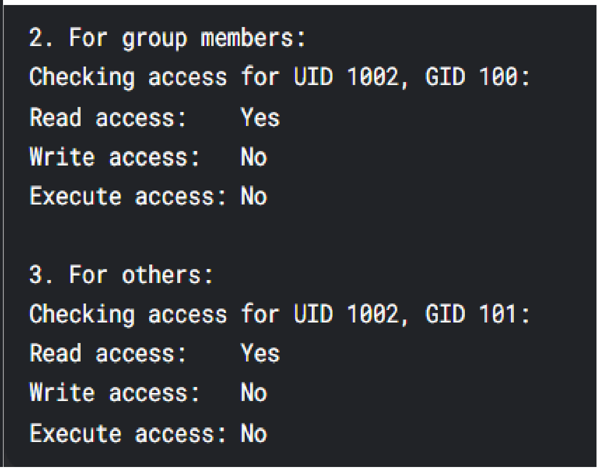
```
#include <stdio.h>

#include <stdlib.h>

#include <sys/stat.h>

#include <unistd.h>

#include <pwd.h>

#include <grp.h>


void display_permissions(mode_t mode);

void display_user_info(uid_t uid, gid_t gid);

void check_file_access(const char *filename, uid_t uid, gid_t gid);


int main(int argc, char *argv[]) {

  if (argc != 2) {

    fprintf(stderr, "Usage: %s <filename>\n", argv[0]);

    exit(EXIT_FAILURE);

  }


  const char *filename = argv[1];

  struct stat file_stat;


  if (stat(filename, &file_stat) == -1) {

    perror("stat");

    exit(EXIT_FAILURE);

  }


  printf("\n=== File Permission Analysis for: %s ===\n", filename);
```

```
    // Display file permissions
    printf("\nFile Permissions: ");
    display_permissions(file_stat.st_mode);

    // Display owner and group information
    printf("\nOwnership Information:\n");
    display_user_info(file_stat.st_uid, file_stat.st_gid);

    // Check access for different users
    printf("\nAccess Checks:\n");
    printf("1. For file owner (user):\n");
    check_file_access(filename, file_stat.st_uid, file_stat.st_gid);

    printf("\n2. For group members:\n");
    check_file_access(filename, getuid(), file_stat.st_gid); // Using current user but file's group

    printf("\n3. For others:\n");
    check_file_access(filename, getuid(), getgid()); // Using current user's UID and GID

    return 0;
}

void display_permissions(mode_t mode) {
    printf((S_ISDIR(mode)) ? "d" : "-");
    printf((mode & S_IRUSR) ? "r" : "-");
    printf((mode & S_IWUSR) ? "w" : "-");
    printf((mode & S_IXUSR) ? "x" : "-");
    printf((mode & S_IRGRP) ? "r" : "-");
    printf((mode & S_IWGRP) ? "w" : "-");
    printf((mode & S_IXGRP) ? "x" : "-");
    printf((mode & S_IROTH) ? "r" : "-");
    printf((mode & S_IWOTH) ? "w" : "-");
    printf((mode & S_IXOTH) ? "x" : "-");
    printf("\n");

    printf("\nPermission Breakdown:\n");
    printf("User (owner) permissions:  %c%c%c\n",
           (mode & S_IRUSR) ? 'r' : '-',
           (mode & S_IWUSR) ? 'w' : '-',
           (mode & S_IXUSR) ? 'x' : '-');
    printf("Group permissions:        %c%c%c\n",
           (mode & S_IRGRP) ? 'r' : '-',
           (mode & S_IWGRP) ? 'w' : '-',
           (mode & S_IXGRP) ? 'x' : '-');
    printf("Other permissions:        %c%c%c\n",
           (mode & S_IROTH) ? 'r' : '-',
           (mode & S_IWOTH) ? 'w' : '-',
           (mode & S_IXOTH) ? 'x' : '-');
}

void display_user_info(uid_t uid, gid_t gid) {
    struct passwd *pw = getpwuid(uid);
    struct group *gr = getgrgid(gid);

    if (pw != NULL) {
        printf("Owner: %s (UID: %d)\n", pw->pw_name, uid);
    } else {
        printf("Owner: UID: %d\n", uid);
    }

    if (gr != NULL) {
        printf("Group: %s (GID: %d)\n", gr->gr_name, gid);
    } else {
        printf("Group: GID: %d\n", gid);
    }
}

void check_file_access(const char *filename, uid_t uid, gid_t gid) {
    printf("Checking access for UID %d, GID %d:\n", uid, gid);

    // Check real access (considering all permission bits)
    printf("Read access:    %s\n", access(filename, R_OK) == 0 ? "Yes" : "No");
    printf("Write access:   %s\n", access(filename, W_OK) == 0 ? "Yes" : "No");
    printf("Execute access: %s\n", access(filename, X_OK) == 0 ? "Yes" : "No");
}
```

Sample Output



```
=== File Permission Analysis for: testfile ===

File Permissions: -rw-r--r--

Permission Breakdown:
User (owner) permissions:  rw-
Group permissions:        r--
Other permissions:        r--

Ownership Information:
Owner: alice (UID: 1001)
Group: users (GID: 100)

Access Checks:
1. For file owner (user):
Checking access for UID 1001, GID 100:
Read access:    Yes
Write access:   Yes
Execute access: No
```



```
2. For group members:
Checking access for UID 1002, GID 100:
Read access:    Yes
Write access:   No
Execute access: No

3. For others:
Checking access for UID 1002, GID 101:
Read access:    Yes
Write access:   No
Execute access: No
```