access()

The `access()` function is used to check whether the calling process has (read, write, execute) permission on a target file or directory. It's defined in the "unistd.h" system header.

It's commonly used to test file existence and permissions before performing operations.

```
#include <unistd.h>

// "pathname" is the absolute/relative path
//           to the file or directory
// "mode" is a bitmask of permission checks
int access(const char *pathname, int mode);
```

Flags for parameter "mode" include:

- Flag "F_OK" checks if the file exists
- Flag "R_OK" checks read permission
- Flag "W_OK" checks write permission
- Flag "X_OK" checks execute (or directory search) permission

Returns 0 if all requested permissions are granted. In case of failure, returns -1 and sets "errno".

Common “errno” values include:

- “EACCES” → Permission denied
- “ENOENT” → File does not exist
- “ENOTDIR” → A path component is not a directory
- “EROFS” → File is on a read-only filesystem

❓ TOCTOU means Time-Of-Check to Time-Of-Use.

It’s a class of software bugs caused by a race condition involving the checking of the state of a part of a system and the use of the results of that check.

Additional information:

- Race condition between “access()” and “open()”, the file permissions may change
  - This is known as TOCTOU vulnerability
  - To avoid this, just call “open()” with the required flags
- Doesn’t follow symbolic links. If “pathname” is a symlink, checks the symlink itself, not the target
- Uses the real User/Group ID for permission checks, instead of the effective ones
  - Real UID/GID refers to the user/group who started the process
  - Effective UID/GID refers to the user/group permissions used by the process
- Use it for quick checks (e.g. “Does this config file exist before loading?”)

Example Usage

```
// check if a file exists
#include <unistd.h>
#include <stdio.h>

int main()
{
   if (access("file.txt", F_OK) == 0)
      printf("File exists!\n");
   else  // "Error: No such file or directory" (ENOENT)
      perror("Error");
   return (0);
}

// check read & write permissions
{
   if (access("data.log", R_OK | W_OK) == 0)
      printf("Can read and write this file.\n");
   else
      perror("Access denied");
}

// check if a program is executable
{
   if (access("/usr/bin/ls", X_OK) == 0)
      printf("You can execute 'ls'.\n");
   else
       perror("Execution not allowed");
}
```