getcwd()

The `getcwd()` (Get Current Working Directory) function retrieves the absolute path of the current working directory of the calling process. It’s defined in the “unistd.h” system header.

It’s commonly used in shells, file managers, and programs that need to track or verify their working directory.

```
#include <unistd.h>

// "buf" is where the path will be stored.
//      If NULL, dynamic memory is allocated to it
// "size" is the buffer size. It's ignored if "buf" is NULL
char *getcwd(char *buf, size_t size);
```

On success, returns “buf”. If “buf” is NULL, returns a dynamically allocated string.

On failure, returns NULL and sets “errno”.

Common “errno” values:

- “ERANGE” → Buffer too small
- “ENOMEM” → Memory allocation failed (”buf” may be NULL)

When “buf” is NULL and “size” is 0, the function ignores “size” and allocates a buffer large enough to hold the current working directory.

- "char *path = getcwd(NULL, 0);”
- “path” must be freed later to avoid memory leaks
- Traditional “getcwd()” requires guessing a safe buffer size (”PATH_MAX”), which isn’t always reliable. This configuration lets the system determine the correct size at runtime, handling arbitrarily long paths

PATH_MAX

`PATH_MAX` is a system-defined (”limits.h”) constant that specifies the maximum allowed length for a file path on a given operating system.

It includes the null terminator “\0”.

It’s used when working with file paths in C programs to ensure buffer sizes are large enough to store any valid path.

- Typical values
    - Linux: usually 4096 B
    - MacOS: 1024 B
    - Windows (”MAX_PATH”): 260 B
- Purpose
    - Used when declaring buffers for file paths
    - Prevents buffer overflows when dealing with filesystem paths

Example Usage

```
#include <unistd.h>
#include <limits.h>
#include <stdio.h>

int main(void)
{
    char cwd[PATH_MAX];

    if (getcwd(cwd, sizeof(cwd))
        printf("Current dir: %s\n", cwd);
    else
        perror("getcwd failed");
    return (0);
}
```