System call implementation

System calls provide the interface between user space and the kernel. System calls such as munmap enable user applications to request services from the kernel, such as managing memory mappings. The munmap system call is used to unmap a previously mapped memory region, freeing resources and ensuring proper memory management

On Debian 12, which uses a version of the Linux kernel, the munmap system call is already implemented and handled by the kernel. However, if you’re learning how system calls work and want to understand the inner workings of munmap in the Linux kernel, this guide will give you an overview of its implementation.

What is munmap?

The munmap system call is part of the **memory management** system in

Its purpose is to **unmap** a region of memory previously mapped by mmap

Example C Program

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <unistd.h>
int main() {
    // Allocate a memory region using mmap
    size_t length = getpagesize();  // Length = one page of memory
    void *addr = mmap(NULL, length, PROT_READ | PROT_WRITE,
MAP_PRIVATE | MAP_ANONYMOUS, -1, 0);

    if (addr == MAP_FAILED) {
        perror(″mmap″);
        return 1;
    }
    printf("Memory mapped at address %p\n", addr);

    // Now unmap the memory
    if (munmap(addr, length) == -1) {
        perror("munmap");
        return 1;
    }

    printf("Memory unmapped successfully\n");

    return 0;
}
```

Explanation of the C Program:

We use `mmap` to allocate a page of memory.

`munmap` is then used to unmap that memory region.

If the `munmap` operation is successful, a message is printed confirming the unmapping.

Compilation and Execution:

```
gcc -o test_munmap test_munmap.c

./test_munmap
```