Debian works well with more advanced tools like KVM and QEMU, so it’s trusted by both beginners and pros.

Question 4: Playing With System Calls in Linux

a. What’s a System Call?

A system call is how a program asks the operating system for help when it wants to do something serious—like reading a file, writing to the screen, or communicating with hardware. You can think of it like a polite request: “Hey OS, can you help me with this?” Without system calls, most programs couldn’t do much at all.

b. What I Did

To get hands-on with system calls, I wrote a small program in C++. It opens a file, reads its contents, and prints it to the screen using real system calls like open(), read(), and write()—not just C++-style file handling. I wanted to see how the system really works under the hood.

c. Example Code:

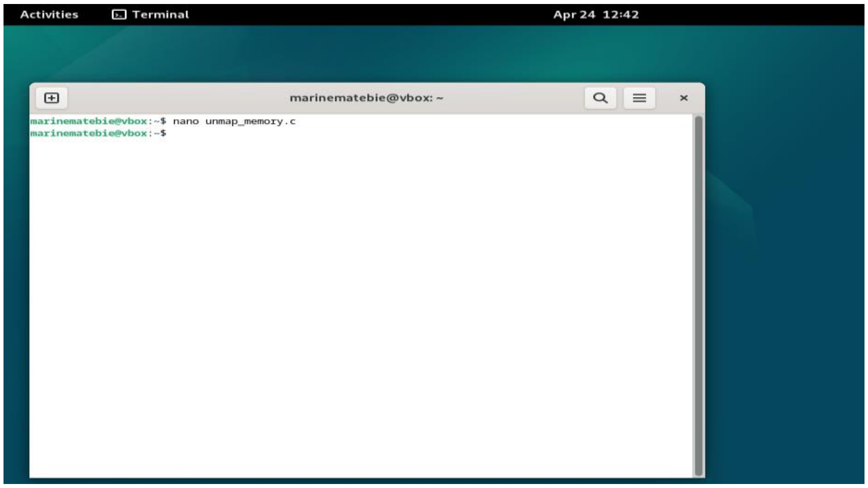
Basic File Reader Here’s the basic version of my C++ code:

```
#include <fcntl.h>
#include <unisted.h>
#include<iostream>
 int main()
 { char buffer[128];
int fd = open("example.txt" , O_RDONLY);
if (fd < 0)
 { std::cerr << "Failed to open file." << std::endl;
return 1; }
int bytes = read(fd, buffer, sizeof(buffer));
write(1, buffer, bytes);
close(fd);
return 0; }
```

This code opens example.txt, reads it into a buffer, and prints it out. It’s simple, butit shows how powerful and direct system calls can be.

d.Going Deeper: Memory Mapping Files

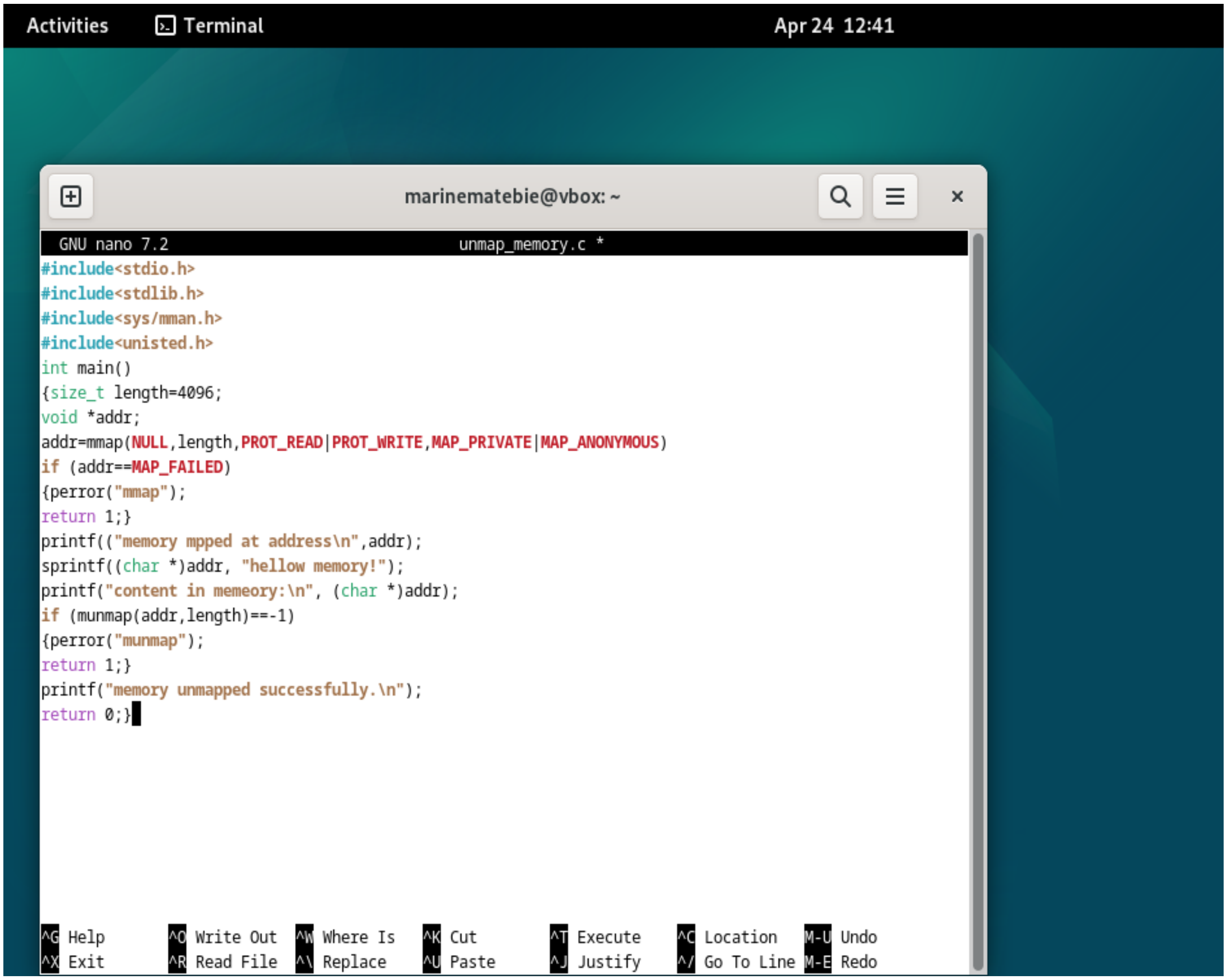
Instead of reading files bit by bit, sometimes it’s better to just map the whole file into memory—this is called memory mapping. It lets the OS handle things more efficiently in the background. Here's the code I used to try it out:



```
marinematebie@vbox:~$ nano unmap_memory.c
marinematebie@vbox:~$
```

```
#include<iostream>
 #include<fcntl.h>
 #include <unisted.h>
#include<sys/mman.h>
 #include <sys/stat.h>
int main()
{ const char* filename = "example.txt";
 int fd = open(filename, O_RDONLY);
if (fd < 0) { perror("open");
return 1;}
struct stat sb;
if (fstat(fd, &sb) == -1)
{ perror("fstat");
 close(fd);
return 1; }
size_t length = sb.st_size;
```

```
 char* data = static_cast(mmap(nullptr, length, PROT_READ,
MAP_PRIVATE, fd, 0));
 if (data == MAP_FAILED) { perror("mmap");
close(fd);
return 1; }
std::cout.write(data, length);
 if (munmap(data, length) == -1) { perror("munmap");
} close(fd);
 return 0; }
```



```
#include<stdio.h>
#include<stdlib.h>
#include<sys/mman.h>
#include<unisted.h>
int main()
{size_t length=4096;
void *addr;
addr=mmap(NULL,length,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS)
if (addr==MAP_FAILED)
{perror("mmap");
return 1;}
printf(("memory mpped at address\n",addr);
sprintf((char *)addr, "hellow memory!");
printf("content in memeory:\n", (char *)addr);
if (munmap(addr,length)==-1)
{perror("munmap");
return 1;}
printf("memory unmapped successfully.\n");
return 0;}
```