**Laporan Latihan Praktikum ke-3  
Sistem Operasi**

**Disusun oleh:**

**RA Siti Zakiyah | 121140103**

**Kelas RD | Cluster RD**



**Program Studi Teknik Informatika**

**Jurusan Teknologi Produksi dan Industri**

**Institut Teknologi Sumatera**

**Lampung Selatan**

**2023**

**BAB I  
TEORI DASAR**

**1.1 Pengertian System Call**

System Call adalah permintaan yang dibuat oleh program ke sistem operasi. Ini memungkinkan aplikasi untuk mengakses fungsi dan perintah dari API sistem operasi. System Call melakukan operasi tingkat sistem, seperti berkomunikasi dengan perangkat perangkat keras dan membaca dan menulis file. Dengan melakukan system call, pengembang dapat menggunakan fungsi pra-tertulis yang didukung oleh Sistem Operasi (OS) alih-alih menulisnya dari awal. Ini menyederhanakan pengembangan, meningkatkan stabilitas aplikasi, dan membuat aplikasi lebih “portabel” antara berbagai versi OS. Setiap kali sebuah program ingin melakukan operasi tertentu pada sistem, seperti membuka file atau menutup jaringan, program tersebut akan memanggil System Call melalui API (Application Programming Interface) untuk mengirimkan permintaan tersebut ke kernel.

**1.2 Penggunaan System Call pada Linux Ubuntu**

Untuk menggunakan System Call pada Linux (Ubuntu), terdapat beberapa cara yang dapat dilakukan. Salah satunya adalah dengan menggunakan bahasa pemrograman C atau C++. Pada bahasa pemrograman tersebut, terdapat beberapa fungsi standar yang disediakan oleh sistem operasi untuk melakukan operasi pada sistem, seperti membuka file, menulis ke file, membaca dari file, dan lain sebagainya. Fungsi-fungsi ini menggunakan System Call untuk berkomunikasi dengan kernel.

**1.3 Jenis System Call pada Linux Ubuntu**

Sistem operasi Linux (Ubuntu) menyediakan banyak jenis System Call yang dapat digunakan untuk melakukan operasi pada sistem. Beberapa jenis System Call yang umum digunakan antara lain :

| **No.** | **Nama** | **Fungsi** |
| --- | --- | --- |
| 1 | open() | Digunakan untuk membuka file. |
| 2 | write() | Digunakan untuk menulis ke dalam file. |
| 3 | close() | Digunakan untuk menutup file yang telah dibuka. |
| 4 | fork() | Digunakan untuk membuat proses baru. |
| 5 | wait() | Digunakan untuk menunggu proses anak selesai dieksekusi. |
| 6 | read() | Digunakan untuk membaca isi file. |
| 7 | exec() | Digunakan untuk menjalankan program baru. |
| 8 | exit() | Digunakan untuk mengakhiri sebuah proses. |

**BAB II  
PEMBAHASAN DAN ANALISIS**

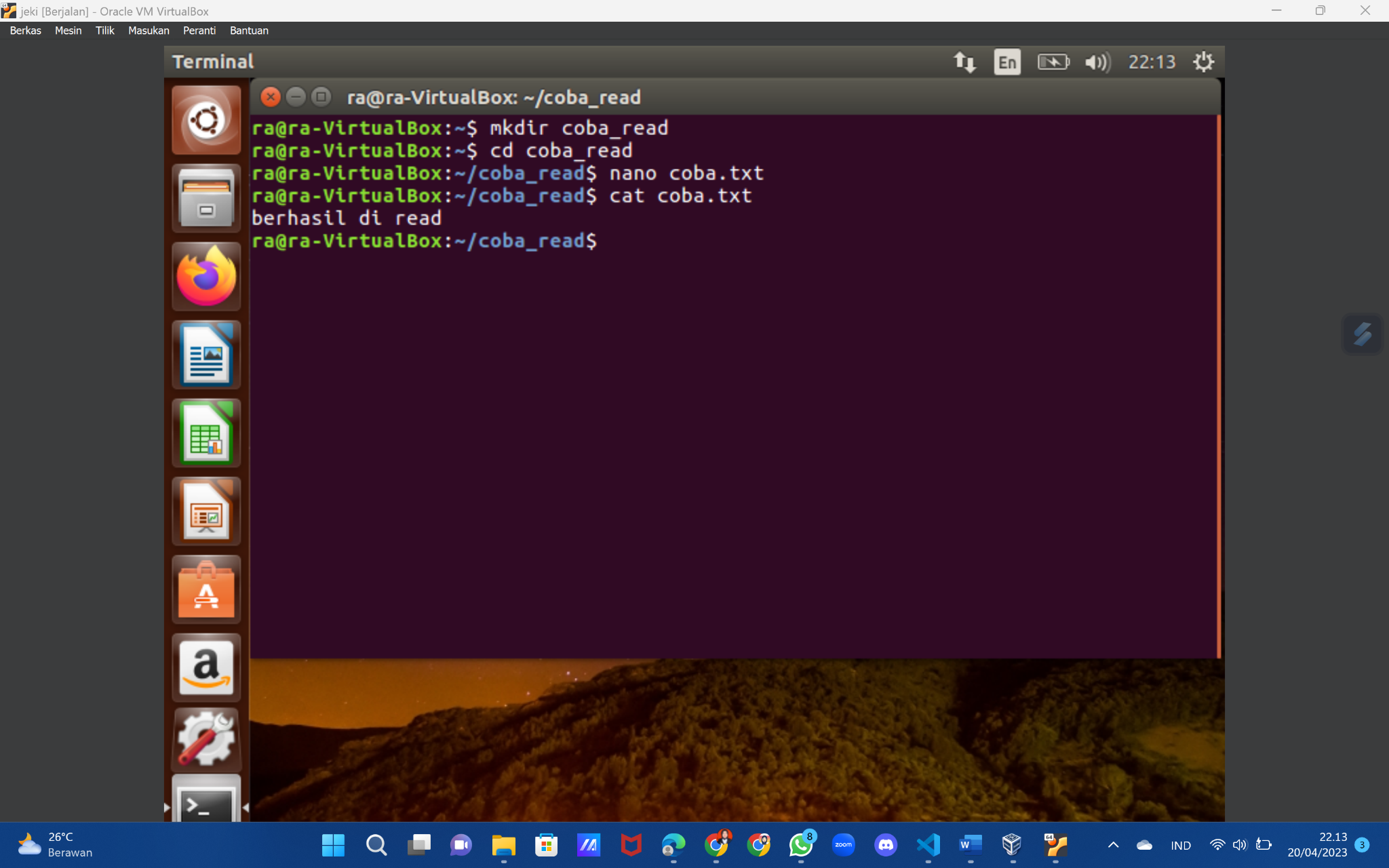
**2.1 Silahkan pelajari system call pada File Management dan implementasikan File Management jenis READ seperti pada Latihan yang dilakukan sebelumnya.**

**2.1.1 Definisi**

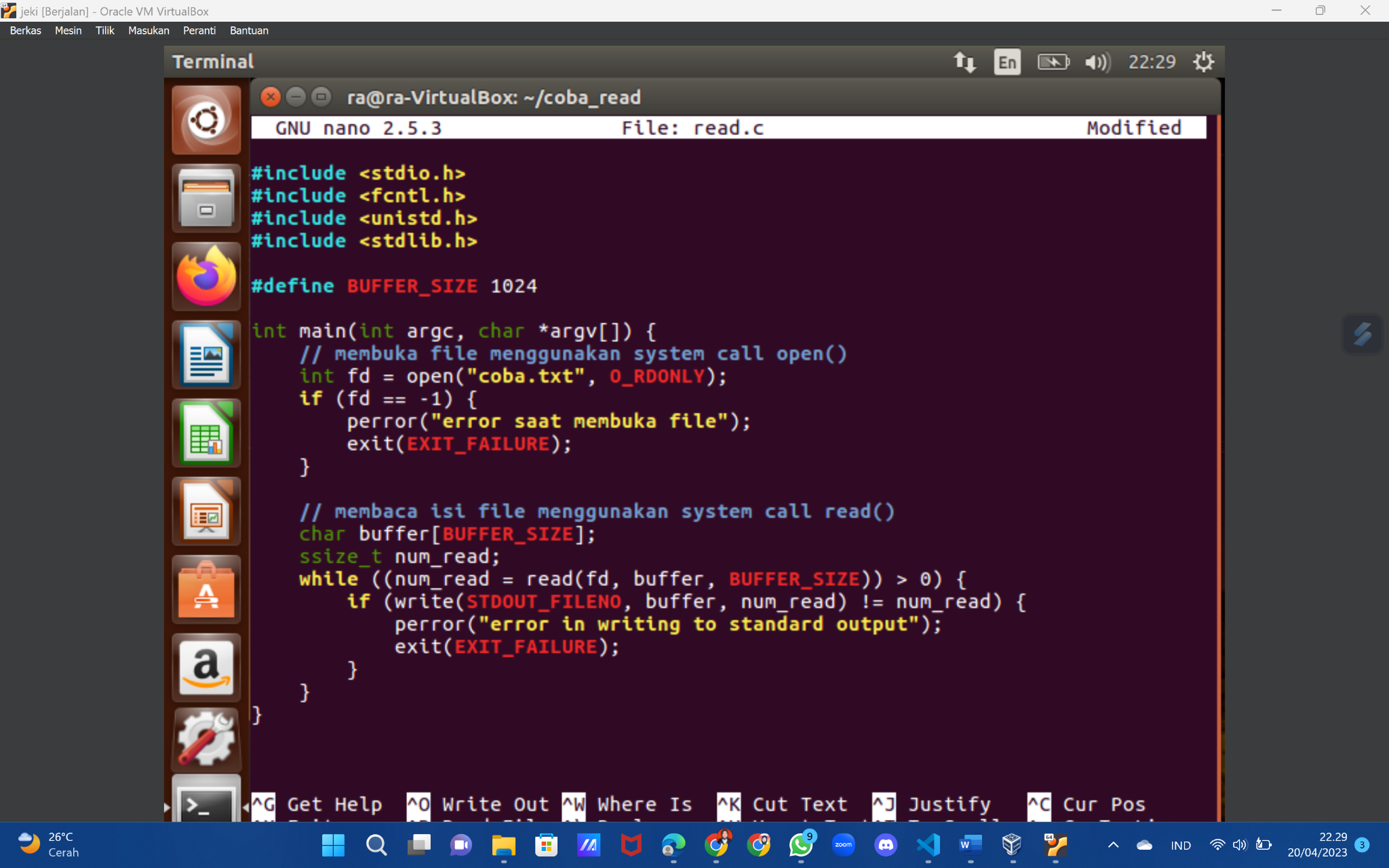
File Management jenis READ merupakan salah satu fungsi dalam sistem operasi yang memungkinkan user untuk membaca isi dari sebuah file. Fungsi READ pada File Management berfungsi untuk membaca data dari sebuah file ke dalam memori. Saat file dibuka dengan mode READ, sistem operasi memastikan bahwa file tersebut dapat dibaca dan kemudian membuka file tersebut. Setelah file dibuka, sistem operasi mengizinkan user untuk membaca isi file menggunakan system call **read()**. System call **read()** akan membaca isi file dan menyalinnya ke dalam buffer yang telah ditentukan oleh user.

**2.1.2 Implementasi**

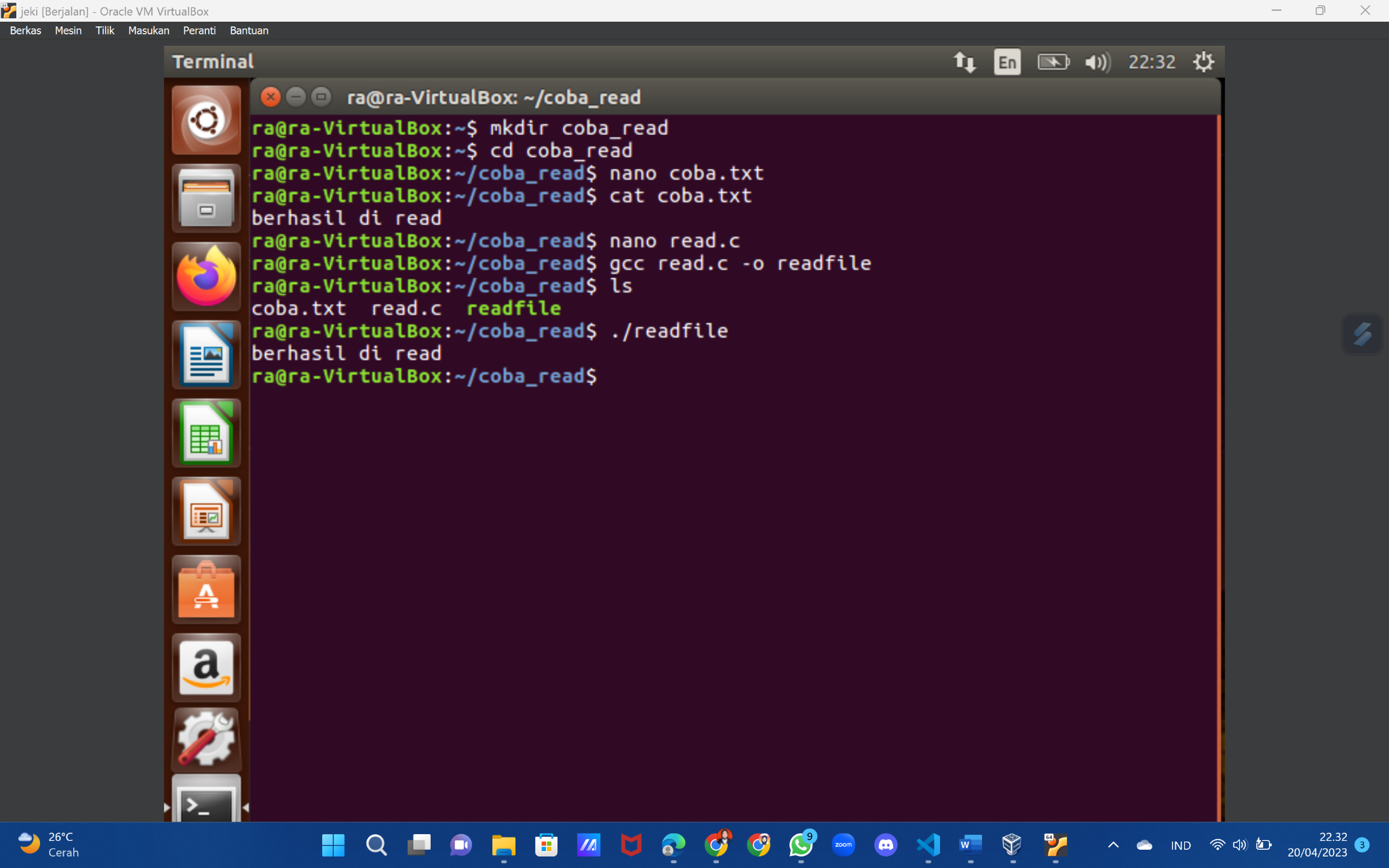
Untuk menggunakan system call **read()**, diperlukan sebuah file yang akan dibaca kemudian isi dari file tersebut akan ditampilkan. Untuk itu dibuat sebuah file txt baru menggunakan perintah “nano namafile” dengan nama file “coba.txt” yang isinya berupa teks=”berhasil di read”.



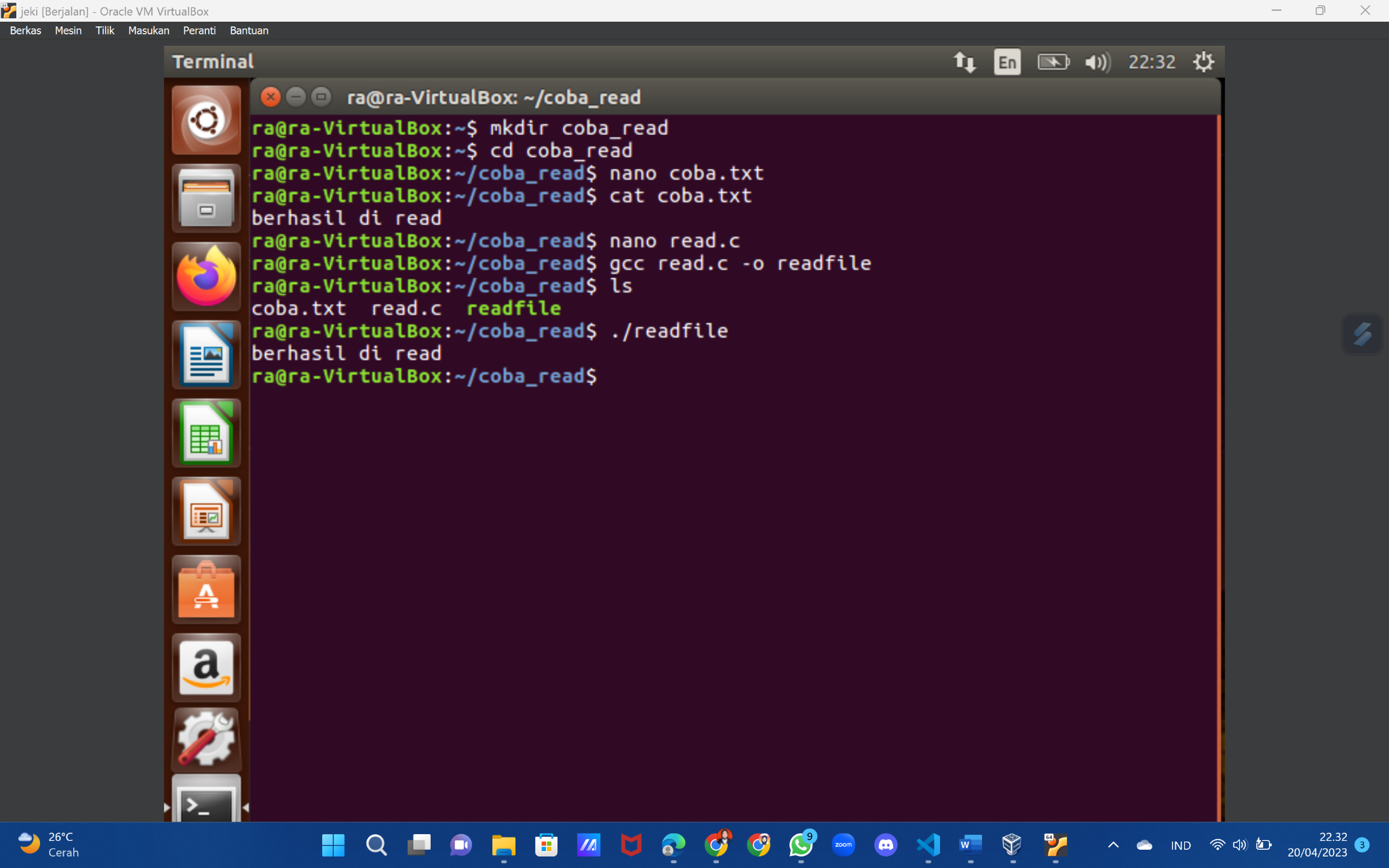
Kemudian membuat sebuah file dengan ekstensi c untuk melakukan system call **read()** menggunakan perintah “nano namafile.c” dengan nama file “read.c” yang diisi dengan code untuk melakukan read().



Kemudian jalankan perintah “gcc read.c -o readfile” untuk membuat sebuah file untuk mengeksekusi program “read.c”.



Lalu lakukan system call yang sudah dibuat sebelumnya menggunakan perintah “./namaperintah” kemudian akan muncul isi dari file “coba.txt”.



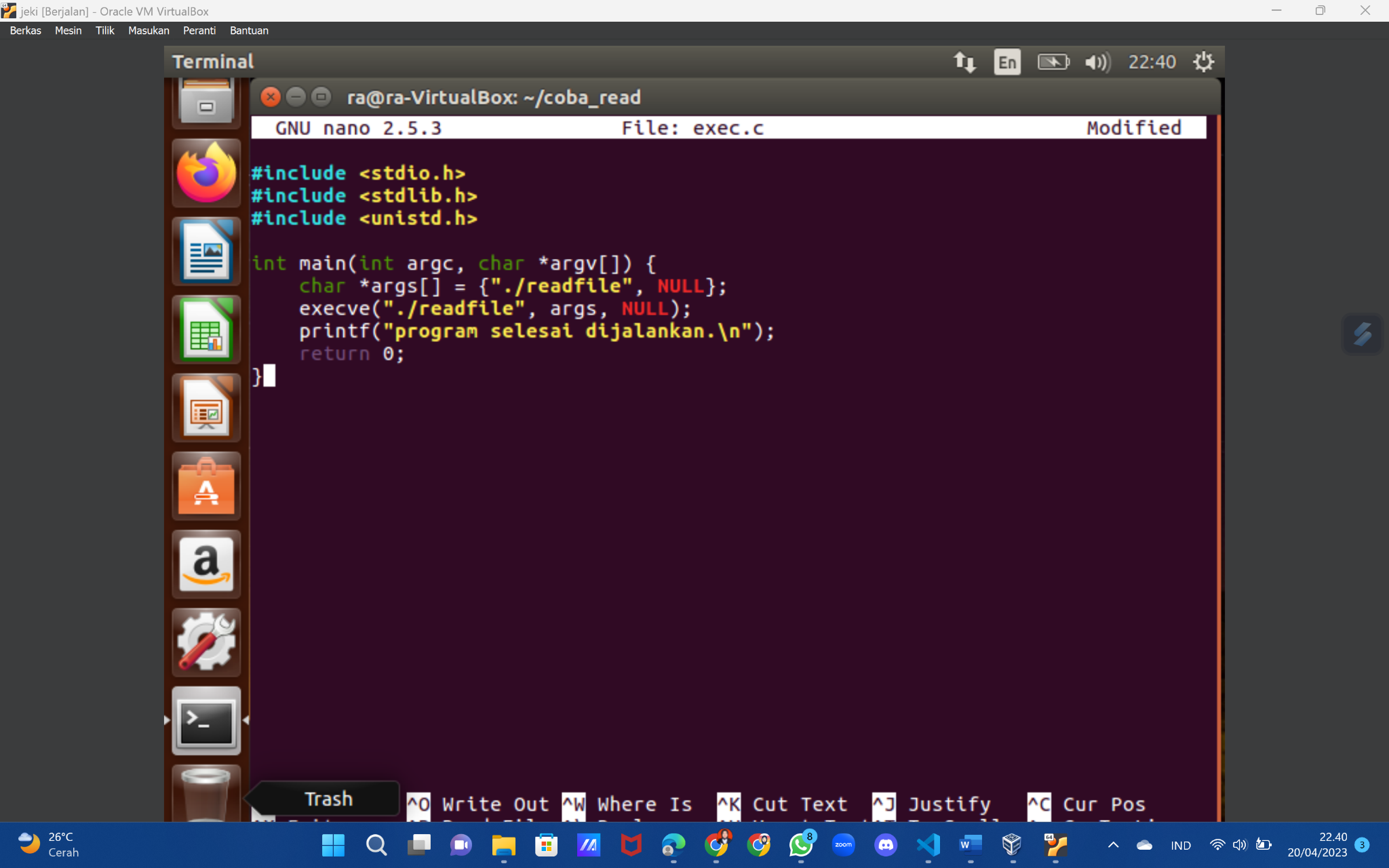
**2.2 Silahkan pelajari system call pada Control Proses dan implementasikan Control Proses jenis EXEC seperti pada Latihan yang dilakukan sebelumnya**

**2.2.1 Definisi**

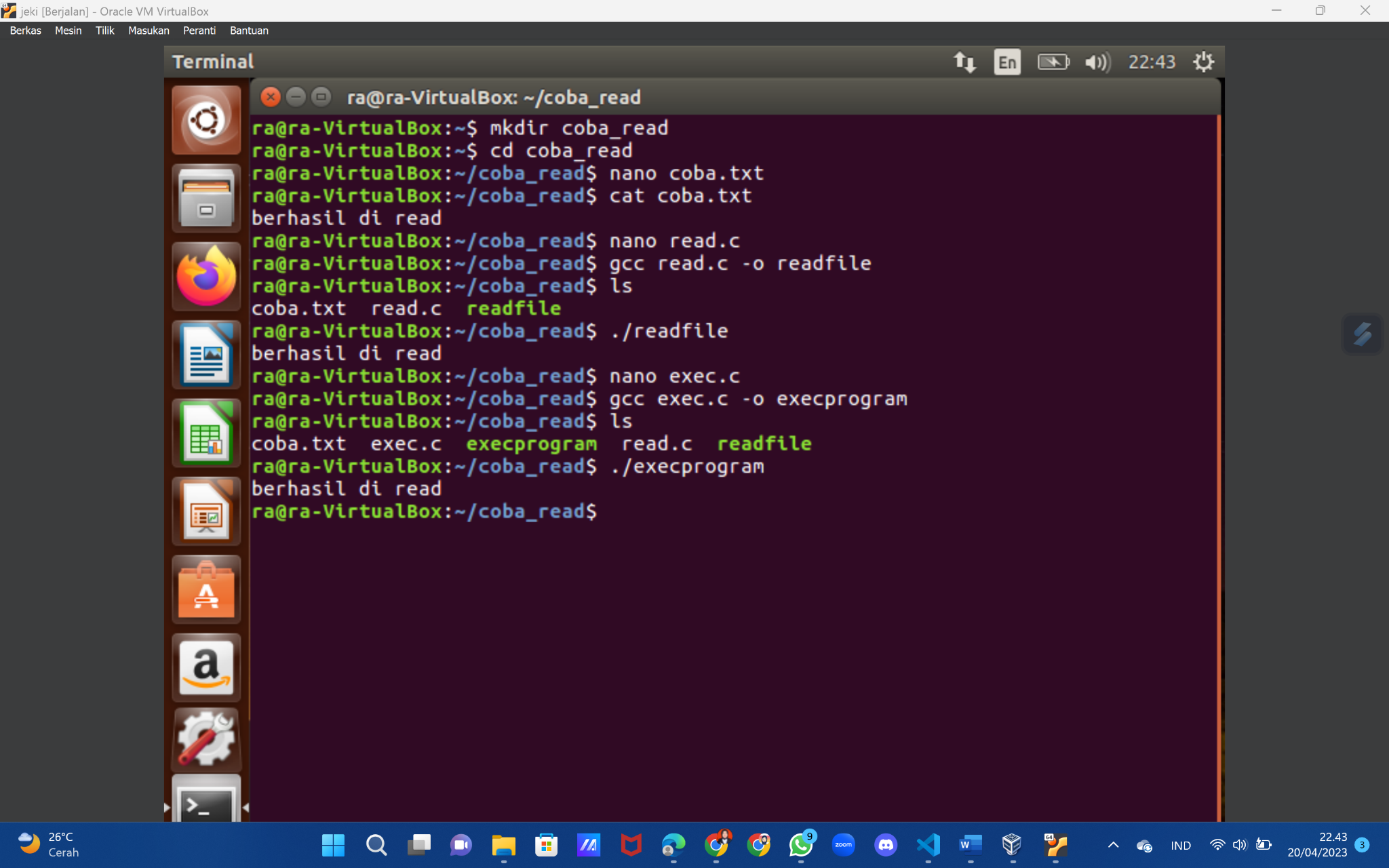
Control Proses jenis EXEC adalah salah satu fungsi dalam sistem operasi yang memungkinkan pengguna untuk menjalankan sebuah program atau perintah dalam sebuah proses yang baru. Ketika sebuah program dijalankan menggunakan Control Proses jenis EXEC, proses yang menjalankan program tersebut akan digantikan oleh proses baru yang akan menjalankan program yang diinginkan.

**2.2.2 Implementasi**

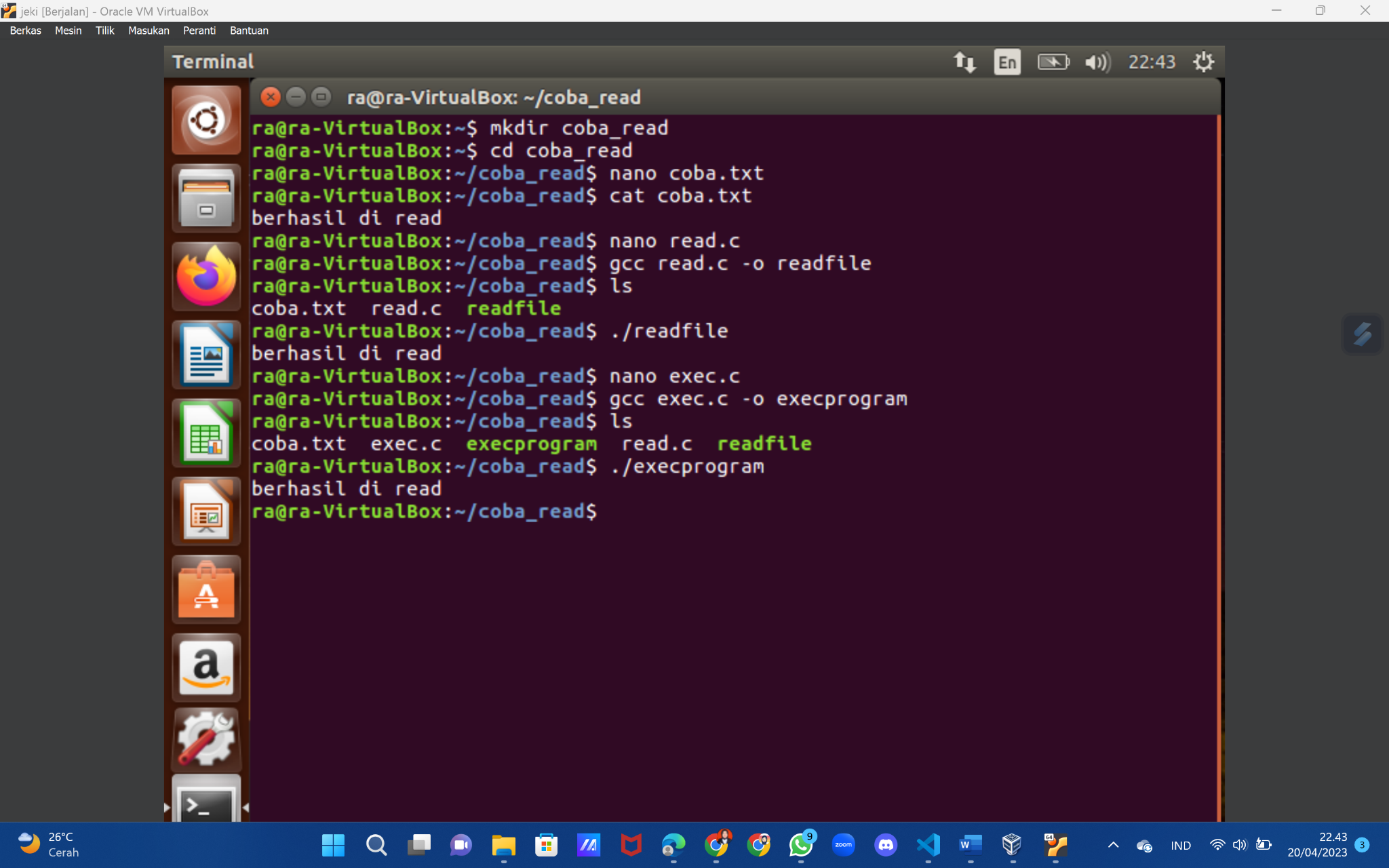
Untuk melakukan system call **exec()** dibutuhkan sebuah file dengan ekstensi C yang berisikan code untuk melakukan perintah **exec(),** dapat menggunakan perintah “nano namafile.c” dengan nama file “exec.c” dengan berisi format code untuk melakukan perintah **exec()** yang mana akan mengeksekusi program “read.c”. Karena untuk menjalankan program “read.c” dibutuhkan untuk menjalankan “./readfile” maka di dalam argumen **exec()** diisi dengan “./readfile”.



Lalu jalankan perintah “gcc exec.c -o execprogram” untuk membuat sebuah file untuk mengeksekusi program “exec.c”.



Kemudian lakukanlah system call yang sudah dibuat sebelumnya menggunakan perintah “./namaperintah” dan akan mengeksekusi program “read.c”, yaitu dengan menampilkan output berupa isi dari file “coba.txt” melalui program “exec.c”.



**BAB III  
KESIMPULAN**

Pada praktikum kali ini, dapat disimpulkan bahwa terdapat beberapa fungsi dari system call yang telah digunakan, antara lain :

1. **open()** : Digunakan untuk membuka file.
2. **read()** : Digunakan untuk membaca isi file.
3. **write()** : Digunakan untuk menulis ke dalam file.
4. **close()** : Digunakan untuk menutup file yang telah dibuka.
5. **fork()** : Digunakan untuk membuat proses baru.
6. **exec()** : Digunakan untuk menjalankan program baru.
7. **wait()** : Digunakan untuk menunggu proses anak selesai dieksekusi.