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

**Disusun oleh:**

**Pannes Diba Sabila | 121140117**

**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 System Call**

System call adalah antarmuka yang disediakan oleh sistem operasi (SO) untuk memungkinkan aplikasi atau program berkomunikasi dengan inti (kernel) dari sistem operasi. Dalam istilah sederhana, system call adalah cara bagi program yang berjalan di dalam mode pengguna (user mode) untuk meminta layanan dari kernel yang berjalan di dalam mode kernel (kernel mode).

System call digunakan oleh aplikasi atau program untuk meminta akses ke sumber daya sistem operasi, seperti berkomunikasi dengan perangkat keras (seperti membaca atau menulis ke disk, mengirim atau menerima data melalui jaringan), mengelola proses (seperti membuat, menghentikan, atau mengatur prioritas proses) , mengelola memori (seperti alokasi dan dealokasi memori), dan melakukan operasi lain yang memerlukan hak akses khusus yang hanya diberikan pada kernel.

Contoh system call umum yang dapat ditemukan dalam sistem operasi adalah:

* Pembacaan atau penulisan berkas (file I/O): System call seperti open(), read(), write(), dan close() digunakan untuk membaca dan menulis data dari atau ke berkas atau input/output perangkat.
* Pengelolaan proses: System call seperti fork(), exec(), dan exit() digunakan untuk membuat, menggantikan, atau menghentikan proses dalam sistem operasi.
* Pengelolaan memori: System call seperti malloc() dan free() digunakan untuk alokasi dan dealokasi memori dalam sistem operasi.
* Komunikasi antar proses: System call seperti pipe(), shmget(), dan semop() digunakan untuk membuka komunikasi antara proses yang berjalan pada sistem operasi yang sama.
* Manajemen jaringan: System call seperti socket(), connect(), dan send() digunakan untuk komunikasi jaringan antara aplikasi dengan perangkat jaringan.

System call biasanya diberikan sebagai antarmuka pemrograman aplikasi (API) yang dapat dipanggil oleh pengembang untuk berinteraksi dengan sistem operasi, sehingga memungkinkan pengembangan aplikasi yang dapat berjalan di atas sistem operasi yang berbeda.

**1.2 Teori Dasar 2**

Fitur Sistem Call

1. Antarmuka

Panggilan sistem menyediakan antarmuka yang terdefinisi dengan baik antara program pengguna dan sistem operasi. Program membuat permintaan dengan memanggil fungsi tertentu, dan sistem operasi merespons dengan menjalankan layanan yang diminta dan mengembalikan hasilnya.

1. Perlindungan

Panggilan sistem digunakan untuk mengakses operasi istimewa yang tidak tersedia

untuk program pengguna normal. Sistem operasi menggunakan hak istimewa ini

untuk melindungi sistem dari akses berbahaya atau tidak sah.

1. Kernel

Ketika sistem dibuat, program untuk sementara dialihkan dari mode

pengguna ke mode kernel. Dalam mode kernel, program memiliki akses ke semua

sumber daya sistem, termasuk perangkat keras, memori, dan proses lainnya.

1. Peralihan Konteks

Panggilan sistem memerlukan peralihan konteks, yang melibatkan penyimpanan

status proses saat ini dan beralih ke mode kernel untuk menjalankan layanan yang

diminta. Ini dapat menimbulkan biaya tambahan, yang dapat memengaruhi kinerja

sistem.

1. Menangani Kesalahan

Panggilan sistem dapat mengembalikan kode kesalahan untuk menunjukkan masalah

dengan layanan yang diminta. Program harus memeriksa kesalahan ini dan

menanganinya dengan tepat.

1. Sinkronisasi Akses

Sistem panggilan dapat digunakan untuk menyinkronkan akses ke sumber daya

bersama, seperti file atau koneksi jaringan. Sistem operasi menyediakan mekanisme

sinkronisasi, seperti kunci atau semafor, untuk memastikan bahwa banyak program

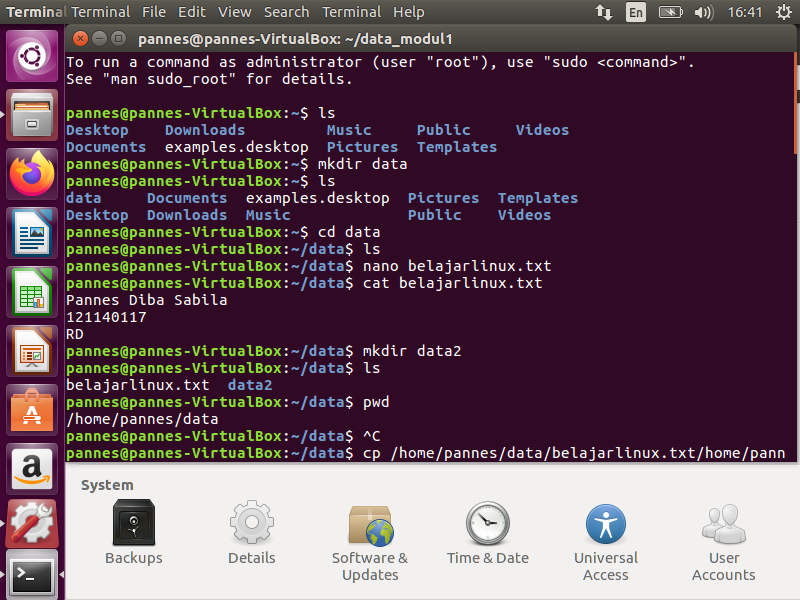
dapat mengakses sumber daya ini dengan aman.

**BAB II  
PEMBAHASAN DAN ANALISIS**

**2.1 Latihan Pertama**

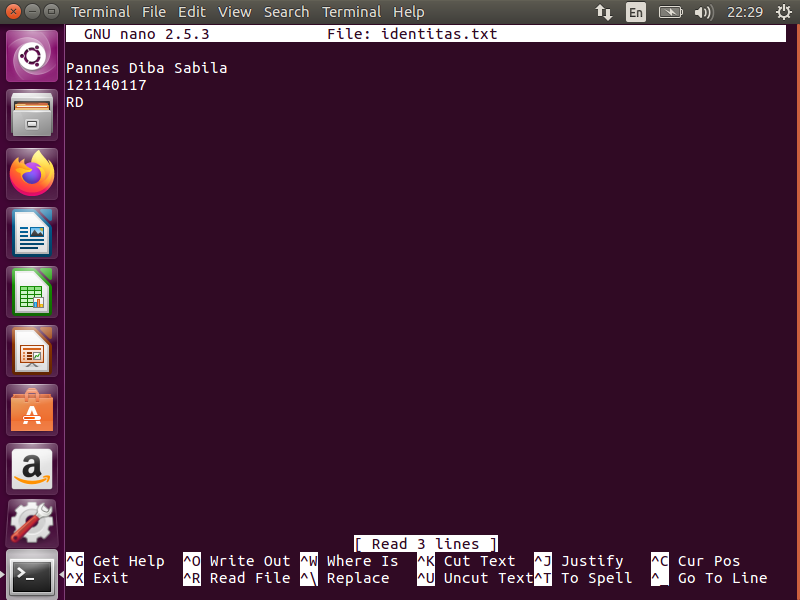
**2.1.1 Langkah Pertama**

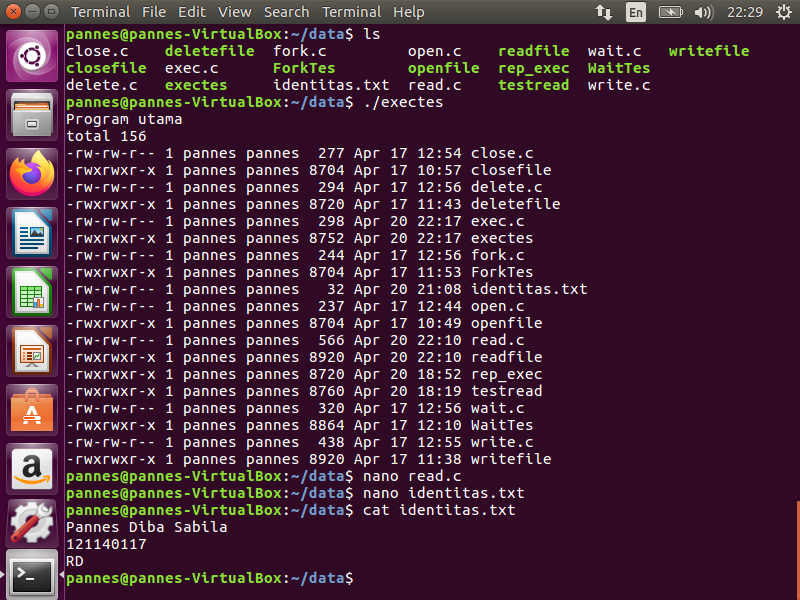
Membuat folder bernama **‘data’.**



**2.1.2 Langkah Kedua**

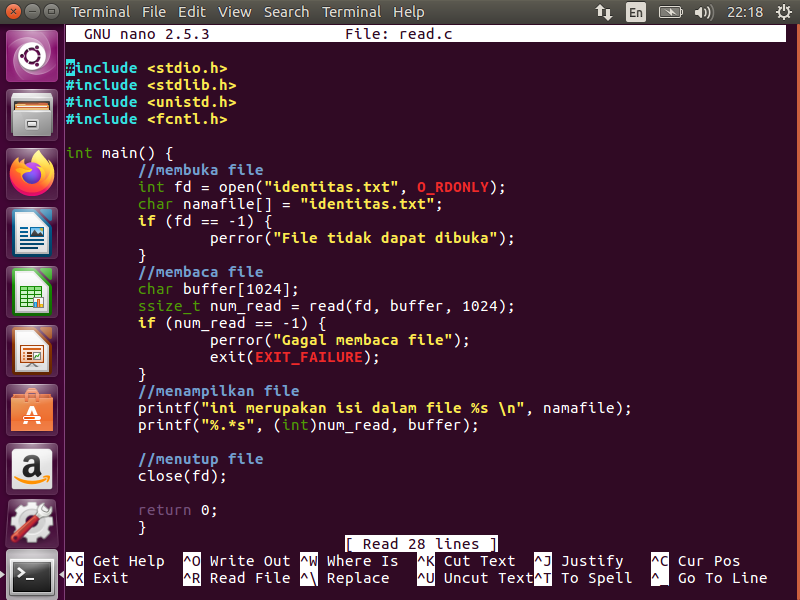
Masuk kedalam folder **‘data’** kemudian membuat file **identitas.txt** yang berisi:





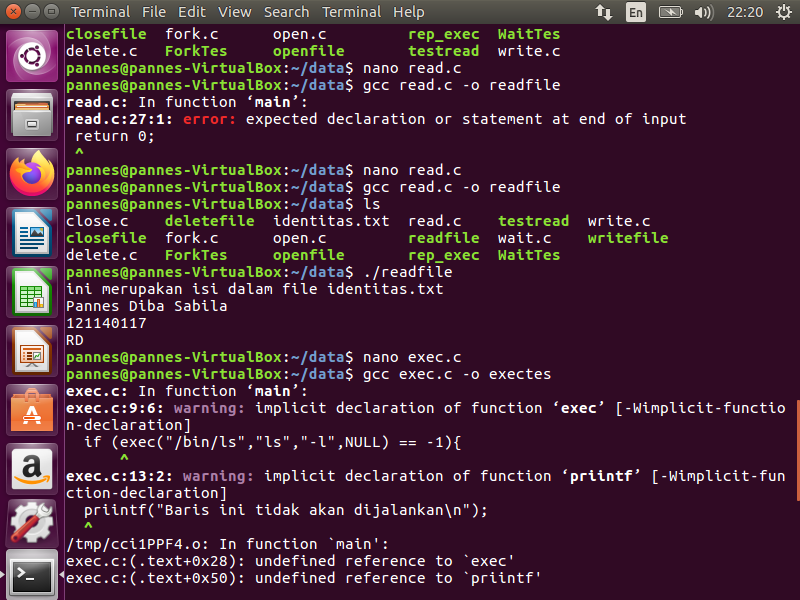
**2.1.3 Langkah Ketiga**

Membuat file dengan extensi C bernama **‘read.c’** pada folder **“data”** yang berisi sebagai berikut.



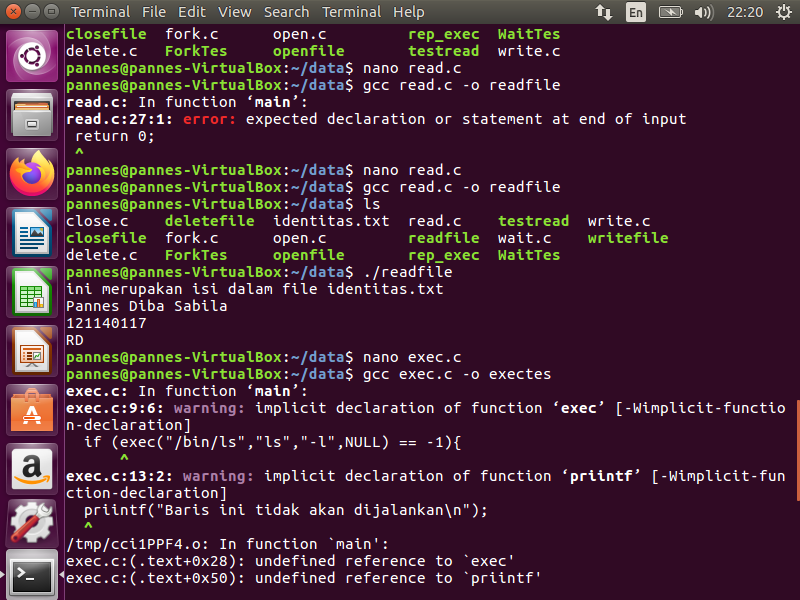
**2.1.4 Langkah Keempat**

Kemudian menjalankan perintah **‘gcc read.c -o readfile’**



**2.1.5 Langkah Kelima**

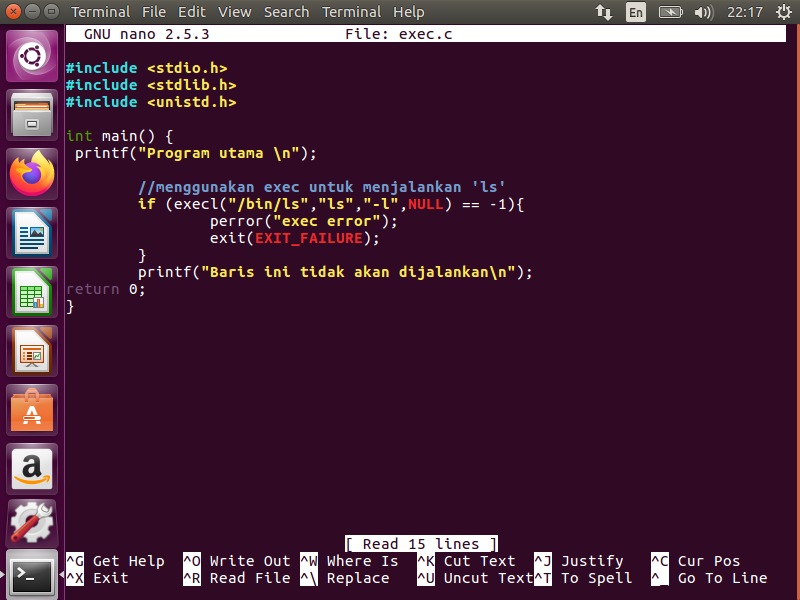
Menjalankan file dengan perintah **“./readfile”**



**2.2 Latihan Kedua**

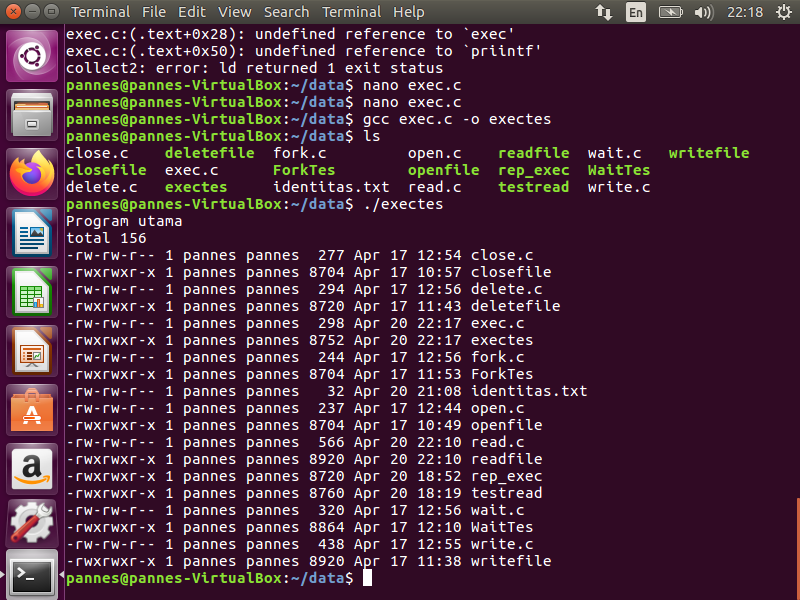
**2.2.1 Langkah Pertama**

Membuat file dengan extensi C bernama **‘exec.c’** pada folder **“data”.**



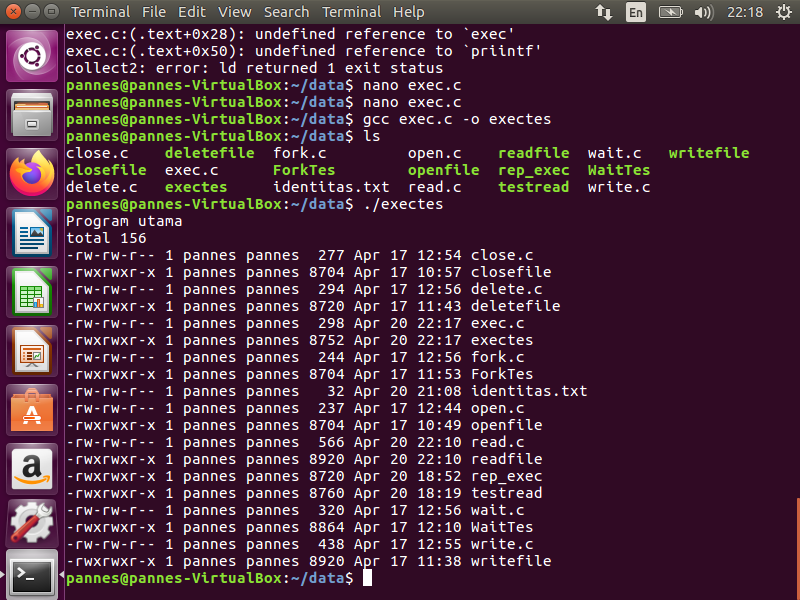
**2.2.2 Langkah Kedua**

Menjalankan perintah **‘gcc exec.c -o exectes’** kemudian periksa file.



**2.2.3 Langkah Ketiga**

Lalu jalankan perintah **“./exectes”** untuk memeriksa file **“exectes”**.



**BAB III  
KESIMPULAN**

Pada bab ini dijabarkan beberapa kesimpulan terkait kegunaan perintah yang digunakan pada praktikum:

1. Perintah ‘nano’ untuk membuat file atau folder
2. Membuka file ‘gcc read.c -o readfile’ dan ‘gcc exec.c -o exectes’
3. Cd untuk membuka file atau folder
4. Sebuah karakter bernama buffer dengan ukuran 1024 akan digunakan untuk menyimpan file konten yang akan dibaca.
5. Fungsi read() digunakan untuk membaca isi file dan mengembalikan jumlah byte yang berhasil dibaca dalam variabel num\_read{ssize\_t num\_read = read[1024]}. Jika terjadi error, fungsi ini akan mengembalikan -1.