LAPORAN PRAKTIKUM LATIHAN 2
SISTEM OPERASI 2024

Nama : **Dito Rifki Irawan**

NIM : **122140153**

Kelas : **RC**

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI SUMATERA

2024

DAFTAR ISI

BAB 1
DASAR TEORI

1.1. Pengenalan System Call

System Call atau sistem panggilan merupakan mekanisme penting dalam komputasi yang memungkinkan program komputer untuk berinteraksi dengan sistem operasi. Ibarat jembatan, sistem panggilan menghubungkan program pengguna yang beroperasi di ruang pengguna (user-space) dengan kernel sistem operasi yang bertugas mengelola sumber daya perangkat keras dan menyediakan layanan dasar.

Peran Penting System Call/Sistem Panggilan:

1. **Akses ke Kernel**

   Sistem operasi memiliki kendali atas sumber daya perangkat keras dan menyediakan layanan fundamental. Sistem panggilan bertindak sebagai gerbang yang memungkinkan program pengguna untuk meminta layanan dari kernel. Tanpa sistem panggilan, program pengguna tidak dapat mengakses sumber daya penting dan menjalankan fungsi-fungsi vital.
2. **Mekanisme Pemanggilan**

   Setiap sistem panggilan memiliki nomor identifikasi unik yang digunakan oleh program pengguna untuk menunjuk layanan yang diinginkan. Program pengguna dapat menggunakan fungsi pembungkus (wrapper) atau instruksi khusus untuk memicu sistem panggilan.
3. **Kategori Sistem Panggilan**

   Beragam sistem panggilan dikategorikan berdasarkan fungsinya, seperti:
   - **Manajemen Proses:** Membuat, menghapus, dan mengelola proses.
   - **Manajemen File:** Membuka, membaca, menulis, dan menutup file.
   - **Alokasi Memori:** Mengalokasikan dan membebaskan memori.
   - **Manajemen Jaringan:** Membuat dan menerima koneksi jaringan.
   - **Dan banyak lagi.**
4. **Keamanan Sistem**

   Sistem operasi memastikan bahwa hanya program yang berwenang dengan izin yang sesuai yang dapat mengakses sistem panggilan. Mekanisme hak akses dan otentikasi digunakan untuk memverifikasi identitas pengguna dan mengizinkan atau menolak akses ke sistem panggilan tertentu.

5. **Pengelolaan Parameter**

   Sistem panggilan umumnya menerima parameter dari program pengguna. Parameter-parameter ini harus didefinisikan dengan jelas dan konvensi pemanggilan harus diikuti untuk memastikan kelancaran komunikasi antara program pengguna dan kernel.

Menguasai konsep sistem panggilan sangat penting bagi pengembang perangkat lunak. Dengan memahami dasar-dasar sistem panggilan, pengembang dapat membangun program yang efisien, aman, dan andal yang dapat berjalan dengan lancar di berbagai platform sistem operasi.

Sistem panggilan merupakan komponen fundamental dalam interaksi program dengan sistem operasi. Ibarat jembatan yang menghubungkan dua sisi, sistem panggilan memungkinkan program pengguna untuk memanfaatkan layanan vital dari kernel, membuka akses ke sumber daya perangkat keras, dan menjalankan fungsi-fungsi penting. Bagi pengembang perangkat lunak, memahami sistem panggilan merupakan kunci untuk menciptakan aplikasi yang kokoh dan tangguh.

BAB 2
ULASAN SOAL

2.1 Membuka File dengan System Call

Mula-mula, buatlah sebuah direktori baru dengan nama "Data". Di dalam direktori tersebut, gunakan perintah nano identitas.txt untuk membuat sebuah file baru dengan nama yang sesuai. Selanjutnya, gunakan perintah nano open.c untuk membuat sebuah file C yang baru. Setelah itu, jalankan perintah gcc open.c -o openfile untuk menghasilkan sebuah sistem yang dapat membuka file. Terakhir, panggil sistem tersebut dengan menggunakan perintah ./openfile.

2.2 Menutup File dengan System Call

Di dalam direktori "Data", gunakan perintah nano close.c untuk membuat file baru dengan ekstensi C. Selanjutnya, jalankan perintah gcc close.c -o closefile untuk menghasilkan sistem penutupan file. Terakhir, panggil sistem tersebut dengan perintah ./closefile.

2.3 Menulis ke dalam File dengan System Call

Di dalam direktori "Data", gunakan perintah nano write.c untuk membuat file baru dengan ekstensi C. Setelah itu, jalankan perintah gcc write.c -o writefile untuk membuat pemanggilan sistem guna menulis ke dalam file. Terakhir, lakukan pemanggilan sistem dengan menjalankan perintah ./writefile. Ini akan menghasilkan file contoh.txt yang dibuat dari kode yang kita tulis di write.c. Gunakan perintah cat contoh.txt untuk menampilkan output yang telah kita hasilkan.

2.4 Menghapus File dengan System Call

Di dalam direktori "Data", gunakan perintah nano delete.c untuk menciptakan file baru dengan ekstensi C. Kemudian, jalankan perintah gcc delete.c -o deletefile untuk membuat panggilan sistem guna menghapus file. Terakhir, eksekusi sistem dengan perintah ./deletefile untuk menghapus file contoh.txt. Setelah itu, gunakan perintah ls untuk memverifikasi bahwa file tersebut berhasil dihapus.

2.5 Implementasi System Call Fork

Buat direktori baru dengan nama "Data1". Di dalam direktori tersebut, gunakan perintah nano fork.c untuk membuat file baru dengan ekstensi C. Setelah itu, jalankan perintah gcc

fork.c -o ForkTes untuk membuat pemanggilan sistem fork. Terakhir, jalankan sistem dengan perintah ./ForkTes.

2.6 Implementasi System Call Wait

Di dalam direktori "Data1", gunakan perintah nano wait.c untuk menciptakan file baru dengan ekstensi C. Selanjutnya, jalankan perintah gcc wait.c -o WaitTes untuk membuat panggilan sistem Wait. Terakhir, eksekusi sistem dengan perintah ./WaitTes.

BAB 3
HASIL DAN JAWABAN

3.1 Membuka File dengan System Call

```
Vbox@dito:~$ mkdir data
Vbox@dito:~$ ls
copas  data  Desktop  Documents  Downloads  Music  Pictures  Public  snap  Templates  Videos
Vbox@dito:~$ cd data
Vbox@dito:~/data$ nano identitas.txt
Vbox@dito:~/data$ cat identitas.txt
Dito RIfki Irawan
122140153
RC
Vbox@dito:~/data$ nano open.c
Vbox@dito:~/data$ sudo apt install gcc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc is already the newest version (4:11.2.0-1ubuntu1).
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Vbox@dito:~/data$ gcc open.c openfile
/usr/bin/ld: cannot find openfile: No such file or directory
collect2: error: ld returned 1 exit status
Vbox@dito:~/data$ gcc open.c -o openfile
Vbox@dito:~/data$ ls
identitas.txt  open.c  openfile
Vbox@dito:~/data$ ./openfile
File berhasil dibuka.
```

Silakan buat sebuah direktori baru dengan nama "Data". Di dalam direktori "Data", gunakan perintah nano identitas.txt untuk membuat sebuah file baru yang sesuai dengan nama file yang diinginkan. Selanjutnya, gunakan perintah nano open.c untuk menciptakan sebuah file baru dengan ekstensi C. Setelah itu, jalankan perintah gcc open.c -o openfile untuk membuat sebuah pemanggil sistem yang bertujuan untuk membuka file. Terakhir, lakukan pemanggilan sistem dengan menggunakan perintah ./openfile.

3.2 Menutup File dengan System Call

```
Vbox@dito:~/data$ nano close.c
Vbox@dito:~/data$ gcc close.c -o closefile
Vbox@dito:~/data$ ls
close.c  closefile  identitas.txt  open.c  openfile
Vbox@dito:~/data$ ./closefile
file berhasil dibuka.
file berhasil ditutup.
```

Di direktori "Data", silakan menggunakan perintah nano close.c untuk menghasilkan sebuah file baru dengan ekstensi C. Kemudian, jalankan perintah gcc close.c -o closefile untuk menciptakan sebuah pemanggilan sistem yang bertujuan untuk menutup file. Terakhir, lakukan pemanggilan sistem dengan menjalankan perintah ./closefile.

3.3 Menulis ke dalam File dengan System Call

```
Vbox@dito:~/data$ nano write.c
Vbox@dito:~/data$ gcc write.c -o writefile
Vbox@dito:~/data$ ls
close.c  closefile  identitas.txt  open.c  openfile  write.c  writefile
Vbox@dito:~/data$ ./writefile
file berhasil dibuka.
Vbox@dito:~/data$ nano write.c
Vbox@dito:~/data$ ./writefile
error ketika membuka file.
Vbox@dito:~/data$ gcc write.c -o writefile
Vbox@dito:~/data$ ls
close.c    contoh.txt     open.c    write.c
closefile  identitas.txt  openfile  writefile
Vbox@dito:~/data$ ./writefile
error ketika membuka file.
Vbox@dito:~/data$ nano write.c
Vbox@dito:~/data$ rm contoh.txt
rm: remove write-protected regular file 'contoh.txt'?
Vbox@dito:~/data$ ls
close.c    contoh.txt     open.c    write.c
closefile  identitas.txt  openfile  writefile
Vbox@dito:~/data$ ls
close.c  closefile  identitas.txt  open.c  openfile  write.c  writefile
Vbox@dito:~/data$ gcc write.c -o writefile
Vbox@dito:~/data$ ./writefile
file berhasil ditulis.
Vbox@dito:~/data$ cat contoh.txt
cat: contoh.txt: Permission denied
Vbox@dito:~/data$ ls
close.c    contoh.txt     open.c    write.c
closefile  identitas.txt  openfile  writefile
Vbox@dito:~/data$ cat contoh.txt
cat: contoh.txt: Permission denied
```

Di dalam direktori "Data", gunakanlah perintah nano write.c untuk menciptakan sebuah file baru dengan ekstensi C. Setelah itu, jalankan perintah gcc write.c -o writefile untuk membuat sebuah pemanggilan sistem yang bertujuan untuk menulis ke dalam file. Terakhir, lakukan pemanggilan sistem dengan mengeksekusi perintah ./writefile. Dengan demikian, akan muncul file contoh.txt yang dibuat berdasarkan kode yang telah kita tulis di dalam file write.c. Untuk menampilkan hasil keluaran yang telah kita buat, gunakan perintah cat contoh.txt.

3.4 Menghapus File dengan System Call

```
Vbox@dito:~/data$ nano delete.c
Vbox@dito:~/data$ gcc delete.c -o deletefile
Vbox@dito:~/data$ ls
close.c     contoh.txt  deletefile     open.c     write.c
closefile   delete.c    identitas.txt  openfile   writefile
Vbox@dito:~/data$ ./deletefile
file contoh.txt berhasil dihapus.
Vbox@dito:~/data$ ls
close.c     delete.c    identitas.txt  openfile   writefile
closefile   deletefile  open.c         write.c
```

Di direktori "Data", gunakan perintah nano delete.c untuk membuat sebuah file baru dengan ekstensi C. Selanjutnya, jalankan perintah gcc delete.c -o deletefile untuk menciptakan sebuah pemanggilan sistem yang bertujuan untuk menghapus file. Terakhir, lakukan pemanggilan sistem dengan mengeksekusi perintah ./deletefile. Hasilnya, file contoh.txt akan dihapus, dan kita dapat memverifikasi hal tersebut dengan menggunakan perintah ls untuk mengetahui apakah file tersebut berhasil dihapus.

3.5 Implementasi System Call Fork

```
Vbox@dito:~$ nano fork.c
Vbox@dito:~$ ls
copas  data1      Documents  Music     Public  Templates
data   Desktop    Downloads  Pictures  snap    Videos
Vbox@dito:~$ cd data1
Vbox@dito:~/data1$ nano fork.c
Vbox@dito:~/data1$ gcc fork.c -o ForkTes
Vbox@dito:~/data1$ ls
fork.c  ForkTes
Vbox@dito:~/data1$ ./ForkTes
Ini adalah proses parent.
Ini adalah proses child.
```

Silakan buat sebuah direktori baru dengan nama "Data1". Di dalam direktori "Data1", gunakan perintah nano fork.c untuk menciptakan sebuah file baru dengan ekstensi C. Kemudian, jalankan perintah gcc fork.c -o ForkTes untuk menciptakan sebuah pemanggilan sistem yang bertujuan untuk melakukan operasi fork. Terakhir, lakukan pemanggilan sistem dengan menjalankan perintah ./ForkTes.

3.6 Implementasi System Call Wait

```
Vbox@dito:~/data1$ nano wait.c
Vbox@dito:~/data1$ gcc wait.c -o waitTes
Vbox@dito:~/data1$ ls
fork.c  ForkTes  wait.c  waitTes
Vbox@dito:~/data1$ ./waitTes
Ini adalah proses child.
Ini adalah proses parent.
Vbox@dito:~/data1$
```

Di dalam direktori "Data1", silakan gunakan perintah nano wait.c untuk menciptakan sebuah file baru dengan ekstensi C. Selanjutnya, jalankan perintah gcc wait.c -o WaitTes untuk membuat sebuah pemanggilan sistem yang bertujuan untuk melakukan operasi Wait. Terakhir, lakukan pemanggilan sistem dengan mengeksekusi perintah ./WaitTes.

BAB 4
KESIMPULAN DAN SARAN

Berdasarkan latihan praktikum yang telah dilakukan, berikut adalah kesimpulan yang dapat diambil:

1. Sistem panggilan (system call) berperan penting dalam memperluas pemahaman mahasiswa terkait interaksi program dengan sistem operasi melalui pemanggilan sistem untuk menyelesaikan tugas-tugas tertentu.

2. Implementasi sistem panggilan pada level kernel mengharuskan pemahaman tentang definisi, pemanggilan, dan eksekusi sistem panggilan oleh kernel.

3. Kemampuan untuk menganalisis kode sumber sistem operasi terkait sistem panggilan memungkinkan pemahaman mendalam mengenai struktur internal sistem operasi dan mekanisme kerja sistem panggilan.

4. Pemahaman terhadap aspek keamanan dan kinerja dalam konteks sistem panggilan menjadi penting, termasuk praktik terbaik untuk menjaga keamanan program dan mengoptimalkan kinerja sistem panggilan.