LAPORAN PRAKTIKUM
SISTEM OPERASI RD
MODUL 3

Oleh :

Muhammad Yusuf (122140193)

Program Studi Teknik Informatika

Institut Teknologi Sumatera

2024

Daftar Isi

1. Dasar Teori

Tulisan di atas membahas mengenai System Call, yang merupakan metode bagi program komputer untuk meminta layanan dari kernel sistem operasi di mana program tersebut dijalankan. Berikut adalah rangkumannya:

System Call (Panggilan Sistem)

1. Definisi: System call adalah metode program komputer untuk meminta layanan dari kernel sistem operasi.
2. Antarmuka: System call menyediakan antarmuka yang terdefinisi antara program pengguna dan sistem operasi melalui API.
3. Fungsi Utama:
    a. Interface: Program membuat permintaan dengan memanggil fungsi tertentu, dan sistem operasi menjalankan layanan yang diminta.
    b. Protection: System call digunakan untuk mengakses operasi istimewa yang tidak tersedia untuk program pengguna normal, dengan hak istimewa untuk melindungi sistem dari akses berbahaya atau tidak sah.
    c. Kernel: Saat system call dibuat, program dialihkan dari mode pengguna ke mode kernel untuk mengakses sumber daya sistem.
    d. Context Switching: System call memerlukan pengalihan konteks yang dapat menimbulkan biaya tambahan.
    e. Error Handling: System call dapat mengembalikan kode kesalahan yang harus ditangani program dengan tepat.
    f. Synchronization: System call digunakan untuk menyinkronkan akses ke sumber daya bersama.
4. Jenis System Call:
    a. Process Control: Mengendalikan proses dalam sistem operasi.
    b. File Management: Mengelola file dan direktori.
    c. Device Management: Mengelola perangkat keras.
    d. Information Maintenance: Mengelola informasi sistem.
    e. Communication System: Mengelola komunikasi antar proses dan jaringan.
    f. Memory Management: Mengelola memori dalam sistem operasi.

2. Ulasan Hasil & Jawaban

Percobaan 1 Open

1. Membuat folder “Data” dan mengakses isi folder tersebut

```
Vbox@yusuf:~$ cd data
```

2. Membuat file dengan nama “identitas.txt” dengan isi Nama, Nim, Kelas

```
Vbox@yusuf:~/data$ nano identitas.txt
Vbox@yusuf:~/data$ cat identitas.txt
Muhammad Yusuf
122140193
RD
```

3. Membuaf file dengan ektensi C dengan nama “open.c”

```
Vbox@yusuf:~/data$ nano open.c
Vbox@yusuf:~/data$ cat open.c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main(){
int fd;

fd=open("identitas.txt", O_RDWR);

if(fd==-1){
printf("file tidak ditemukan.\n");
return 1;
}

printf("file berhasil dibuka.\n");
close(fd);

return 0;
}
```

4. Lakukan instalasi gcc dengan perintah “sudo apt install gcc” dan jalankan perintah “gcc open.c -o openfile”

```
Vbox@yusuf:~/data$ sudo apt install gcc
[sudo] password for wupxy:
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 27 not upgraded.
```

5. Lakukan pemanggilan sistem yang telah kita buat dengan perintah “./openfile”. Jika file berhasil dibuka maka akan menampilkan luaran berikut

```
Vbox@yusuf:~/data$ gcc open.c -o openfile
Vbox@yusuf:~/data$ ./openfile
file berhasil dibuka.
```

Perobaan 2 Close

1. Mengakses folder “Data” dan membuaf file dengan ektensi C dengan nama “close.c”

```
Vbox@yusuf:~/data$ nano close.c
Vbox@yusuf:~/data$ cat close.c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main(){
int fd;
fd=open("identitas.txt", O_RDONLY);

if(fd==-1){
printf("file tidak ditemukan.\n");
return 1;
}

printf("file berhasil dibuka.\n");

close(fd);
printf("file berhasil ditutup.\n");

return 0;
}
```

2. Jalankan perintah “gcc close.c -o closefile”

```
Vbox@yusuf:~/data$ gcc close.c -o closefile
Vbox@yusuf:~/data$ ls
close.c  closefile  identitas.txt  open.c  openfile
```

3. Lakukan pemanggilan sistem yang telah kita buat dengan perintah “./closefile”

```
Vbox@yusuf:~/data$ ./closefile
file berhasil dibuka.
file berhasil ditutup.
```

Percobaan 3 Write

1. Mengakses folder “Data” dan membuaf file dengan ektensi C dengan nama “write.c”

```
Vbox@yusuf:~/data$ nano write.c
Vbox@yusuf:~/data$ cat write.c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main(){
int fd;
char buf[1024];

fd=open("contoh.txt", O_CREAT | O_WRONLY | O_TRUNC);

if(fd==-1){
printf("error ketika membuka file.\n");
exit(EXIT_FAILURE);
}

strcpy(buf, "Saya sedang melakukan praktikum mata kuliah sistem operasi.\n");
write(fd, buf, strlen(buf));

printf("file berhasil ditulis.\n");
close(fd);

return 0;
}
```

2. Jalankan perintah “gcc write.c -o writefile”.

```
Vbox@yusuf:~/data$ gcc write.c -o writefile
Vbox@yusuf:~/data$ ls
close.c  closefile  identitas.txt  open.c  openfile  write.c  writefile
```

3. Lakukan pemanggilan sistem yang telah kita buat dengan perintah “./writefile”. Jika file berhasil ditulis maka akan menghasilkan file “contoh.txt”

```
Vbox@yusuf:~/data$ ./writefile
file berhasil ditulis.
Vbox@yusuf:~/data$ ls
close.c  closefile  contoh.txt  identitas.txt  open.c  openfile  write.c  writefile
Vbox@yusuf:~/data$ cat contoh.txt
Saya sedang melakukan praktikum mata kuliah sistem operasi.
```

Percobaan 4 Delete

1. Mengakses folder “Data” dan membuaf file dengan ektensi C dengan nama “delete.c”

```
Vbox@yusuf:~/data$ nano delete.c
Vbox@yusuf:~/data$ cat delete.c
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

int main(){

char filename[]="contoh.txt";
int result=unlink(filename);

if(result==0){
printf("file %s berhasil dihapus. \n", filename);
}else{
printf("gagal menghapus file %s. \n", filename);
}

return 0;
}
```

2. Jalankan perintah “gcc delete.c -o deletefile”

```
Vbox@yusuf:~/data$ gcc delete.c -o deletefile
Vbox@yusuf:~/data$ ls
close.c  closefile  contoh.txt  delete.c  deletefile  identitas.txt  open.c  openfile  write.c  wri
```

3. Lakukan pemanggilan sistem yang telah kita buat dengan perintah “./deletefile”. Jika file berhasil dihapus maka akan menghasilkan luaran

```
Vbox@yusuf:~/data$ ./deletefile
file contoh.txt berhasil dihapus.
```

Percobaan 5 Fork

1. Mengakses folder “Data1” dan membuaf file dengan ektensi C dengan nama “fork.c”

```
Vbox@yusuf:~$ mkdir data1
Vbox@yusuf:~$ cd data1
Vbox@yusuf:~/data1$ nano fork.c
Vbox@yusuf:~/data1$ cat fork.c
#include <stdio.h>
#include <unistd.h>

int main(){
pid_t pid;
pid=fork();

if(pid==1){
printf("Fork gagal");
}else if(pid==0){
printf("Ini adalah proses child. \n");
}else{
printf("Ini adalah proses parent. \n");
}

return 0;
}
```

2. Jalankan perintah “gcc fork.c -o ForkTes”

```
Vbox@yusuf:~/data1$ gcc fork.c -o forkTes
Vbox@yusuf:~/data1$ ls
fork.c  forkTes
```

3. Lakukan pemanggilan sistem yang telah kita buat dengan perintah “./Fork.Test”. Jika file berhasil dihapus maka akan menghasilkan luaran

```
Vbox@yusuf:~/data1$ gcc fork.c -o forkTes
Vbox@yusuf:~/data1$ ls
fork.c  forkTes
Vbox@yusuf:~/data1$ ./forkTes
Ini adalah proses parent.
Ini adalah proses child.
```

Percobaan 6 Wait

1. Mengakses folder “Data1” dan membuaf file dengan ektensi C dengan nama “wait.c”

```
Vbox@yusuf:~/data1$ nano wait.c
Vbox@yusuf:~/data1$ cat wait.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

int main(){
pid_t pid;
int status;

pid=fork();

if(pid==1){
printf("Fork gagal");
}else if(pid==0){
printf("Ini adalah proses child. \n");
exit(0);
}else{
wait(&status);
printf("Ini adalah proses parent. \n");
}

return 0;
}
```

2. Jalankan perintah “gcc wait.c -o WaitTes”

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

3. Lakukan pemanggilan sistem yang telah kita buat dengan perintah “./Wait.Test”. Jika file berhasil dihapus maka akan menghasilkan luaran

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

3. Kesimpulan dan Saran

Dalam praktikum yang melibatkan pemahaman tentang System Call, saya dapat menarik beberapa kesimpulan kunci. Pertama, System Call merupakan jembatan krusial antara program pengguna dan kernel sistem operasi. Melalui panggilan sistem, program dapat meminta layanan tertentu dari sistem operasi seperti mengakses file, mengelola proses, mengelola perangkat keras, dan melakukan komunikasi antar proses. Antarmuka yang terdefinisi dengan baik memungkinkan program untuk berinteraksi dengan sistem operasi secara efisien dan aman.

Kedua, dalam penggunaan System Call, terdapat beberapa aspek yang perlu diperhatikan. Salah satunya adalah pentingnya perlindungan data dan hak istimewa yang diberikan kepada program melalui System Call. Selain itu, proses Context Switching saat melakukan panggilan sistem dapat memengaruhi kinerja sistem, sehingga diperlukan pengelolaan yang baik dalam meminimalkan biaya tambahan yang timbul. Kesalahan handling juga menjadi aspek penting yang harus diperhatikan agar program dapat merespons dengan tepat terhadap kondisi yang tidak diinginkan. Dengan pemahaman yang mendalam tentang System Call, program dapat berjalan dengan lebih efisien dan dapat diandalkan dalam mengakses sumber daya sistem.