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.

System Call READ

System call read digunakan untuk membaca data dari suatu file descriptor atau input standar. Data yang dibaca akan disimpan di dalam buffer yang disediakan oleh pengguna.

System Call Exec

System call EXEC digunakan untuk menggantikan proses yang sedang berjalan dengan proses baru yang dieksekusi. Proses baru ini akan mengambil alih proses yang sebelumnya sedang berjalan.

2. Ulasan Hasil & Jawaban

Percobaan 1 READ

1. Buat file dengan ekstensi C dengan nama ”read.c” dan file dengan code untuk mempraktikan contoh system call jenis READ.

```
Vbox@yusuf:~/data1$ nano read.txt
Vbox@yusuf:~/data1$ rm read.txt
Vbox@yusuf:~/data1$ nano read.txt
Vbox@yusuf:~/data1$ cat read.txt
Ini file Read.txt

Vbox@yusuf:~/data1$ nano read.c
```

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

int main() {
    // Buka file untuk dibaca (READ-ONLY)
    int file_descriptor = open("read.txt", O_RDONLY);

    // Periksa apakah file berhasil dibuka
    if (file_descriptor == -1) {
        perror("Error opening file");
        return EXIT_FAILURE;
    }

    // Baca isi file
    char buffer[100];  // Buffer untuk menyimpan data yang dibaca
    int bytes_read = read(file_descriptor, buffer, sizeof(buffer));

    // Periksa apakah pembacaan file berhasil
    if (bytes_read == -1) {
        perror("Error reading file");
        close(file_descriptor);  // Tutup file
        return EXIT_FAILURE;
    }

    // Tampilkan isi file yang telah dibaca
    printf("Isi file 'read.txt':\n%s\n", buffer);

    // Tutup file setelah selesai membaca
    close(file_descriptor);

    return EXIT_SUCCESS;
}
```

2. Jalankan perintah ”gcc read.c -o readTes” dan run perintah ”./readTes”.

```
Vbox@yusuf:~/data1$ gcc read.c -o readTes
Vbox@yusuf:~/data1$ ls
fork.c  forkTes  read.c  readTes  read.txt  wait.c  waitTes
Vbox@yusuf:~/data1$ ./readTes
Isi file 'read.txt':
Ini file Read.txt
```

Perobaan 2 EXEC

1. Buat file dengan ekstensi C dengan nama ”exec.c” dan file dengan code untuk mempraktikan contoh system call jenis EXEC.

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

int main() {
    // Mengeksekusi perintah/command menggunakan system()
    printf("Mengeksekusi perintah 'ls -l':\n");
    system("ls -l");

    return EXIT_SUCCESS;
}
```

2. Jalankan perintah ”gcc exec.c -o exec” dan run perintah ”./exec”.

```
Vbox@yusuf:~/data1$ gcc exec.c -o execTes
Vbox@yusuf:~/data1$ ls
exec.c  execTes  fork.c  forkTes  read.c  readTes  read.txt  wait.c  waitTes
Vbox@yusuf:~/data1$ ./execTes
Mengeksekusi perintah 'ls -l':
total 84
-rw-rw-r-- 1 wupxy wupxy   208 Apr 23 21:28 exec.c
-rwxrwxr-x 1 wupxy wupxy 16000 Apr 23 21:29 execTes
-rw-rw-r-- 1 wupxy wupxy   229 Apr 23 21:13 fork.c
-rwxrwxr-x 1 wupxy wupxy 16040 Apr 23 21:13 forkTes
-rw-rw-r-- 1 wupxy wupxy   883 Apr 23 21:27 read.c
-rwxrwxr-x 1 wupxy wupxy 16176 Apr 23 21:27 readTes
-rw-rw-r-- 1 wupxy wupxy    19 Apr 23 21:27 read.txt
-rw-rw-r-- 1 wupxy wupxy   308 Apr 23 21:16 wait.c
-rwxrwxr-x 1 wupxy wupxy 16176 Apr 23 21:16 waitTes
```

3. Kesimpulan dan Saran

Setelah melakukan praktikum yang fokus pada System Call READ dan EXEC, dapat disimpulkan bahwa kedua System Call ini memiliki peran yang penting dalam interaksi antara program pengguna dan kernel sistem operasi. System Call READ memungkinkan program untuk membaca data dari file atau perangkat input tertentu, sedangkan System Call EXEC digunakan untuk menjalankan program baru di dalam proses yang sedang berjalan.