

Responsi

Andreandhiki Riyanta Putra - 23/517511/PA/22191

 My GitHub

1. Memory Mapping (mmap)

Buatlah program C yang menggunakan mmap untuk memodifikasi isi file bernama hello.txt. Awalnya, file tersebut berisi string "Hello, world!". Program Anda harus mengubah karakter pertama file tersebut sehingga string menjadi "Jello, world!". Pastikan program Anda menangani pembukaan file, memory mapping, dan pembersihan dengan benar.

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>

int main() {
    const char* filename = "hello.txt";
    int fd;
    struct stat file_stat;
    char* mapped_file;
    size_t file_size;

    // Langkah 1: Buka file untuk membaca dan menulis
    fd = open(filename, O_RDWR);
    if (fd == -1) {
        perror("Error membuka file");
        return 1;
    }

    // Langkah 2: Dapatkan ukuran file
    if (fstat(fd, &file_stat) == -1) {
        perror("Error mendapatkan ukuran file");
        close(fd);
        return 1;
    }
    file_size = file_stat.st_size;

    // Langkah 3: Memory-map file
    mapped_file = mmap(NULL, file_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if (mapped_file == MAP_FAILED) {
        perror("Error mapping file");
        close(fd);
        return 1;
    }

    // Langkah 4: Modifikasi isi file
    mapped_file[0] = 'J'; // Ubah 'H' menjadi 'J'


    // Langkah 5: Sinkronisasi perubahan dan pembersihan
    if (msync(mapped_file, file_size, MS_SYNC) == -1) {
        perror("Error mensinkronisasi perubahan");
    }
    if (munmap(mapped_file, file_size) == -1) {
        perror("Error unmapping file");
    }
    close(fd);

    return 0;
}
```

Figure 1: Source Code

```
~/skj/responsi$ echo -n "Hello, world!" > hello.txt
~/skj/responsi$ nano hello.txt
```

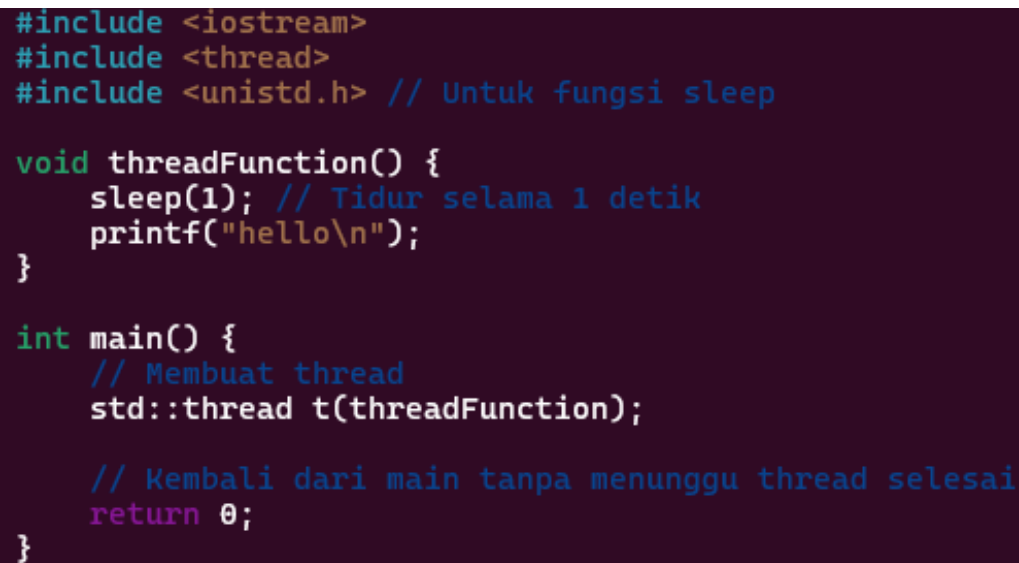
Figure 2: Command to add string in hello.txt



```
GNU nano 6.2
Hello, world!
```

Figure 3: Output hello.txt after changes

2. Debugging Threads



```
#include <iostream>
#include <thread>
#include <unistd.h> // Untuk fungsi sleep

void threadFunction() {
    sleep(1); // Tidur selama 1 detik
    printf("hello\n");
}

int main() {
    // Membuat thread
    std::thread t(threadFunction);

    // Kembali dari main tanpa menunggu thread selesai
    return 0;
}
```

Figure 4: Source Code

- (a) Jelaskan mengapa program ini tidak mencetak "hello" ketika dijalankan.
- Hal ini dikarenakan thread yang dijalankan oleh fungsi `threadFunction()` tidak menunggu sampai selesai sebelum program `main()` keluar. Dalam kode ini, setelah membuat thread baru dengan `std::thread t(threadFunction)`, program `main()` langsung mengembalikan nilai 0 tanpa menunggu thread selesai. Akibatnya, thread belum sempat mencetak "hello" sebelum program berakhir.
- Untuk membuat program mencetak "hello", kita perlu menunggu thread selesai sebelum `main()` keluar. Salah satu caranya adalah dengan memanggil `t.join()` di dalam `main()`, yang akan memblokir eksekusi `main()` hingga thread selesai.
- Dengan menambahkan `t.join()`, program `main()` akan menunggu thread yang dijalankan oleh `threadFunction()` selesai sebelum keluar, sehingga "hello" akan dicetak.
- (b) Modifikasi program agar mencetak "hello" sebelum program selesai. Tulis kode yang sudah diperbaiki.

```
#include <iostream>
#include <thread>
#include <unistd.h>

void threadFunction() {
    sleep(1);    // Simulate work
    printf("hello\n"); // Thread's task
}

int main() {
    std::thread t(threadFunction); // Create thread
    t.join(); // Wait for thread completion
    return 0;
}
```

Figure 5: Modification in hello.cpp

```
andre@LAPTOP-GKDG6LNE:~/skj/responsi$ ./hello_after
hello
```

Figure 6: Output after modification

3. Pemrograman Jaringan (Socket Programming)

Server.c Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);
    char *message = "Hello, Client!";

    // Langkah 1: Membuat socket
    server_fd = socket(AF_INET, SOCK_STREAM, 0);

    // Langkah 2: Bind socket ke port 8080
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(8080);
    if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Error bind socket");
        return 1;
    }

    // Langkah 3: Listen untuk koneksi masuk
    if (listen(server_fd, 3) < 0) {
        perror("Error listen koneksi");
        return 1;
    }
    printf("Server mendengarkan di port 8080...\n");

    // Langkah 4: Terima koneksi
    client_fd = accept(server_fd, (struct sockaddr *)&client_addr, &client_len);
    if (client_fd < 0) {
        perror("Error menerima koneksi");
        return 1;
    }

    // Langkah 5: Kirim pesan ke client
    write(client_fd, message, strlen(message));

    // Langkah 6: Tutup koneksi
    close(client_fd);
    close(server_fd);

    return 0;
}
```

Figure 7: Server.c

Client.c Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

int main() {
    int client_fd;
    struct sockaddr_in server_addr;
    char buffer[1024] = {0};

    // Langkah 1: Membuat socket
    client_fd = socket(AF_INET, SOCK_STREAM, 0);

    // Langkah 2: Tentukan alamat server
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(8080);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

    // Langkah 3: Terhubung ke server
    if (connect(client_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Error koneksi ke server");
        return 1;
    }

    // Langkah 4: Terima pesan dari server
    read(client_fd, buffer, sizeof(buffer));
    printf("Diterima dari server: %s\n", buffer);

    // Langkah 5: Tutup koneksi
    close(client_fd);

    return 0;
}
```

Figure 8: Client.c

```
andre@LAPTOP-GKDG6LNE:~/skj/responsi$ ./server
Server mendengarkan di port 8080...
andre@LAPTOP-GKDG6LNE:~/skj/responsi$
```

Figure 9: Run Server.c

```
andre@LAPTOP-GKDG6LNE:~/skj/responsi$ ./client
Diterima dari server: Hello, Client!
andre@LAPTOP-GKDG6LNE:~/skj/responsi$
```

Figure 10: Run Client.c