MAKALAH

SISTEM OPERASI

Oleh:

Rio Firmansyah

E32230939

Golongan B

PROGRAM STUDI TEKNIK KOMPUTER

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI JEMBER

TAHUN 2023/2024

KATA PENGANTAR

Puji syukur diucapkan kehadirat Allah Swt. atas segala rahmat-Nya sehingga makalah ini dapat tersusun sampai selesai. Tidak lupa kami mengucapkan terima kasih terhadap bantuan dari pihak yang telah berkontribusi dengan memberikan sumbangan baik pikiran maupun materi.

Penulis sangat berharap semoga makalah ini dapat menambah pengetahuan dan pengalaman bagi pembaca. Bahkan kami berharap lebih jauh lagi agar makalah ini bisa pembaca praktikkan dalam kehidupan sehari-hari.

Bagi kami sebagai penyusun merasa bahwa masih banyak kekurangan dalam penyusunan makalah ini karena keterbatasan pengetahuan dan pengalaman kami. Untuk itu kami sangat mengharapkan kritik dan saran yang membangun dari pembaca demi kesempurnaan makalah ini.

Jember,24 Desember 2023

Rio Firmansyah

E32230939

DAFTAR ISI

BAB I

PENDAHULUAN

A . Latar Belakang

Sistem operasi (SO) adalah perangkat lunak sistem yang mengelola sumber daya perangkat keras dan menyediakan layanan dasar bagi perangkat lunak aplikasi. Fungsinya sangat penting dalam operasi komputer, dan tanpa sistem operasi, interaksi antara pengguna, perangkat keras, dan perangkat lunak aplikasi akan menjadi sangat sulit. Sistem operasi menyediakan lingkungan eksekusi untuk aplikasi dan menyatukan semua sumber daya perangkat keras agar dapat digunakan secara efektif. Berbagai jenis sistem operasi yang ada termasuk Windows, macOS, Linux, Android, dan banyak lagi, masing-masing dirancang untuk berbagai platform dan kebutuhan.

B. Rumusan Masalah

1. Apa itu Konsep Thread?
2. Apa itu Konsep Penjadwalan?
3. Apa itu Penjadwalan CPU?
4. Apa itu Sinkronisasi?

C. Tujuan

1. Mengetahui tentang Konsep Thread?
2. Mengetahui tentang Konsep Penjadwalan?
3. Mengetahui tentang Penjadwalan CPU?
4. Mengetahui tentang Sinkronisasi?

D. Manfaat

Dengan menyediakan kerangka kerja untuk menjalankan aplikasi, berkomunikasi dengan perangkat keras, dan menyatukan pengelolaan sumber daya, sistem operasi memungkinkan komputer berfungsi dengan efisien, andal, dan dapat dioperasikan dengan mudah oleh pengguna.

BAB II

PEMBAHASAN

A. Konsep Thread

Konsep "thread" dalam konteks pemrograman mengacu pada unit kecil dari eksekusi program. Thread merupakan bagian kecil dari suatu proses yang dapat dieksekusi secara independen. Dalam suatu program, proses dapat memiliki satu atau lebih thread, dan masing-masing thread dapat menjalankan tugas-tugasnya sendiri secara bersamaan.

Beberapa konsep utama terkait thread meliputi:

1. Concurrency (Kesesuaian): Thread memungkinkan suatu program untuk menjalankan beberapa tugas secara bersamaan. Dengan menggunakan thread, program dapat melakukan beberapa operasi sekaligus tanpa harus menunggu tugas tertentu selesai sebelum melanjutkan ke yang lain.
2. Share Data: Thread dalam suatu proses dapat berbagi data. Namun, ini juga bisa menjadi sumber potensi konflik dan masalah (race condition) jika tidak dielola dengan benar. Oleh karena itu, sinkronisasi diperlukan untuk memastikan bahwa akses ke data bersama dilakukan secara aman.
3. Efisiensi: Penggunaan thread dapat meningkatkan efisiensi penggunaan sumber daya. Misalnya, dalam aplikasi yang melibatkan operasi input/output (I/O), satu thread dapat melakukan operasi I/O sementara thread lainnya melanjutkan tugas-tugas pemrosesan.
4. Kerja Paralel: Thread juga memungkinkan untuk implementasi kerja paralel. Dengan menggunakan beberapa thread, beberapa tugas dapat dilakukan secara paralel, meningkatkan kinerja program pada sistem dengan multiple core atau processor.
5. Deadlock dan Race Condition: Salah satu tantangan utama dalam pemrograman thread adalah mengelola situasi deadlock (kekakuan) dan race condition (perlombaan). Deadlock terjadi ketika dua atau lebih thread saling menunggu satu sama lain tanpa bisa melanjutkan, sedangkan race condition terjadi ketika dua atau lebih thread mencoba mengakses dan memodifikasi data bersama secara bersamaan tanpa sinkronisasi yang benar.

Berbagai bahasa pemrograman menyediakan dukungan untuk pengelolaan thread, seperti Java, Python, C++, dan lainnya. Pemahaman yang baik tentang thread dan pengelolaannya menjadi penting ketika mengembangkan aplikasi yang membutuhkan pemrosesan paralel atau konkurensi.

B. Konsep Penjadwalan

Konsep penjadwalan (scheduling) dalam sistem komputer merujuk pada proses pengelolaan sumber daya komputer untuk menentukan urutan eksekusi tugas atau pekerjaan. Penjadwalan sangat penting untuk memastikan efisiensi penggunaan sumber daya dan memenuhi kebutuhan dari berbagai proses atau tugas yang berjalan pada sistem operasi yang sama. Beberapa konsep penjadwalan utama melibatkan:

1. Penjadwalan Proses (Process Scheduling): Penjadwalan proses berkaitan dengan pengelolaan eksekusi proses pada sistem operasi. Sistem operasi harus memutuskan proses mana yang akan dieksekusi selanjutnya dan kapan proses tersebut akan dieksekusi. Penjadwalan proses dapat dilakukan menggunakan berbagai algoritma penjadwalan seperti First-Come, First-Served (FCFS), Shortest Job Next (SJN), Round Robin, dan sebagainya.
2. Penjadwalan CPU (CPU Scheduling): Ini adalah subkonsep dari penjadwalan proses yang berkaitan dengan alokasi CPU kepada proses yang siap dieksekusi. Algoritma penjadwalan CPU mencakup pengaturan prioritas, alokasi waktu CPU, dan mekanisme pengaturan beralih antar proses.
3. Penjadwalan I/O (I/O Scheduling): Selain CPU, penjadwalan juga diperlukan untuk operasi input/output (I/O). Proses dapat terlibat dalam operasi I/O seperti membaca atau menulis ke disk, dan penjadwalan I/O memastikan bahwa sumber daya I/O digunakan secara efisien.
4. Penjadwalan Disk (Disk Scheduling): Jika beberapa proses atau tugas perlu mengakses disk secara bersamaan, penjadwalan disk membantu menentukan urutan akses ke disk agar penggunaan disk menjadi lebih efisien.
5. Penjadwalan Jaringan (Network Scheduling): Dalam lingkungan jaringan komputer, penjadwalan juga penting untuk menentukan cara data dan paket dikirim dan diterima antar perangkat.

Tujuan utama dari penjadwalan adalah meningkatkan throughput sistem, meminimalkan waktu respons, memaksimalkan penggunaan sumber daya, dan memberikan layanan yang adil kepada berbagai proses atau pengguna. Algoritma penjadwalan yang efektif dapat berkontribusi pada kinerja sistem operasi secara keseluruhan.

C. Penjadwal CPU

Penjadwalan CPU adalah salah satu aspek penting dalam manajemen sistem operasi yang melibatkan pengelolaan eksekusi proses pada unit pemrosesan pusat (CPU). Penjadwalan CPU bertujuan untuk memastikan efisienya penggunaan CPU dan memenuhi kebutuhan berbagai proses yang berjalan dalam sistem operasi yang sama.

Beberapa konsep utama dalam penjadwalan CPU termasuk:

1. Penjadwalan Proses (Process Scheduling): Ini adalah bagian dari penjadwalan CPU yang berkaitan dengan pengelolaan eksekusi proses pada sistem operasi. Sistem operasi harus memutuskan proses mana yang akan dieksekusi selanjutnya dan kapan proses tersebut akan dieksekusi.
2. Penjadwalan CPU (CPU Scheduling): Fokus utama dari penjadwalan CPU adalah alokasi CPU kepada proses yang siap dieksekusi. Algoritma penjadwalan CPU menentukan urutan eksekusi proses dan durasi waktu yang diberikan kepada setiap proses sebelum beralih ke proses berikutnya.
3. Prioritas Proses (Process Priority): Beberapa algoritma penjadwalan memperhitungkan tingkat prioritas yang diberikan kepada setiap proses. Proses dengan prioritas yang lebih tinggi mungkin mendapatkan akses CPU lebih banyak daripada proses dengan prioritas yang lebih rendah.
4. Burst Time: Waktu yang diperlukan oleh suatu proses untuk menyelesaikan satu periode eksekusi pada CPU disebut "burst time." Algoritma penjadwalan menggunakan burst time untuk menghitung durasi waktu yang harus dialokasikan kepada setiap proses.
5. Waiting Time dan Turnaround Time: Waiting time adalah total waktu yang dihabiskan oleh suatu proses dalam antrian sebelum mendapatkan akses ke CPU, sementara turnaround time adalah total waktu yang diperlukan oleh suatu proses untuk menyelesaikan eksekusinya dari awal hingga selesai.

Contoh algoritma penjadwalan CPU termasuk:

- First-Come, First-Served (FCFS): Proses yang pertama kali tiba, pertama kali dilayani.
- Shortest Job Next (SJN) atau Shortest Job First (SJF): Proses dengan burst time terpendek dijadwalkan terlebih dahulu.
- Round Robin (RR): Setiap proses diberikan quantum waktu eksekusi, dan setelah waktu tersebut habis, proses beralih ke proses berikutnya dalam antrian.
- Priority Scheduling: Proses dengan prioritas tertinggi dijadwalkan terlebih dahulu.

Implementasi algoritma penjadwalan CPU dapat bervariasi tergantung pada kebutuhan sistem operasi dan tujuan kinerja yang diinginkan.

D. Sinkronisasi

Sinkronisasi dalam konteks pemrograman dan sistem komputer merujuk pada teknik atau mekanisme yang digunakan untuk mengelola akses bersama terhadap sumber daya oleh beberapa proses atau thread. Sinkronisasi diperlukan untuk memastikan bahwa operasi-operasi yang melibatkan data

bersama dilakukan secara aman dan terhindar dari konflik atau kondisi balapan (race condition).

Beberapa konsep terkait sinkronisasi melibatkan:

1. Mutex (Mutual Exclusion): Mutex adalah mekanisme sinkronisasi yang digunakan untuk memberikan kontrol eksklusif terhadap suatu sumber daya. Hanya satu proses atau thread yang dapat memegang mutex pada satu waktu. Jika satu proses atau thread sudah memegang mutex, yang lain harus menunggu hingga mutex dilepaskan.
2. Semaphore: Semaphore adalah variabel khusus yang digunakan untuk mengontrol akses ke sumber daya yang terbatas. Ini dapat memiliki nilai selain 0 atau 1 dan digunakan untuk mengatur jumlah proses atau thread yang dapat mengakses sumber daya secara bersamaan.
3. Monitor: Monitor adalah struktur data atau blok kode yang menyediakan metode untuk mengatur akses ke sumber daya bersama. Monitor dapat memastikan bahwa hanya satu thread dapat masuk ke dalam blok kritis pada satu waktu.
4. Condition Variables: Condition variables digunakan untuk memberi tahu thread tentang perubahan kondisi yang terjadi dalam program. Mereka sering digunakan bersama-sama dengan mutex untuk mengoordinasikan akses ke sumber daya.
5. Deadlock: Deadlock terjadi ketika dua atau lebih proses atau thread saling menunggu satu sama lain dan tidak dapat melanjutkan eksekusi. Sinkronisasi yang tepat harus dirancang untuk mencegah deadlock.
6. Race Condition: Race condition terjadi ketika dua atau lebih proses atau thread berusaha mengakses dan memodifikasi data bersama tanpa sinkronisasi yang benar. Ini dapat menghasilkan hasil yang tidak dapat diprediksi.

Sinkronisasi penting dalam pengembangan perangkat lunak untuk memastikan integritas data dan menghindari masalah konkurensi. Kesalahan dalam sinkronisasi dapat mengakibatkan bug yang sulit diprediksi dan sulit dideteksi. Oleh karena itu, pemrogram harus berhati-hati dalam merancang dan mengimplementasikan mekanisme sinkronisasi untuk memastikan konsistensi dan keandalan sistem.

BAB III

PENUTUP

A. Kesimpulan

Konsep-konsep ini saling terkait dan penting dalam pengembangan aplikasi dan sistem yang dapat berjalan secara efisien dan andal. thread memungkinkan konkurensi dan paralelisme, sedangkan penjadwalan dan penjadwal CPU mengelola eksekusi thread. sinkronisasi diperlukan untuk menghindari masalah konkurensi ketika thread atau proses berbagi sumber daya. desain dan implementasi yang baik dari konsep-konsep ini penting untuk mencapai kinerja dan keandalan sistem yang diinginkan.

DAFTAR PUSTAKA

https://socs.binus.ac.id/2020/12/13/thread-unit-pemanfaatan-cpu/

https://www.anbidev.com/penjadwalan-proses/

https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/

https://www.linkedin.com/pulse/understanding-process-synchronization-operating-systems-aritra-pain