Sistem Operasi

Catatan Kuliah #5

Alauddin Maulana Hirzan, M. Kom
0607069401

Manajemen Proses - *Konkurensi, Mutual Exclusion, dan Synchronization*

Konkurensi

Konkurensi

Apa itu *Konkurensi*?

Konkurensi dalam sistem operasi mengacu pada kemampuan sistem untuk mendukung pelaksanaan beberapa tugas atau proses secara bersamaan. Hal ini dapat dicapai melalui teknik seperti multitasking, multiprosesing, dan multithreading.

Konkurensi dapat memberikan beberapa manfaat pada sistem operasi, termasuk peningkatan daya tanggap dan hasil, pemanfaatan sumber daya yang lebih baik, dan dukungan yang lebih baik untuk aplikasi interaktif.

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts, Tenth Edition. Wiley.

Konkurensi

Manfaat *Konkurensi* #1

- Peningkatan Daya Tanggap: Konkurensi memungkinkan beberapa proses atau thread berjalan secara bersamaan, yang dapat membantu sistem operasi merespons permintaan pengguna dengan lebih cepat.
- Peningkatan Pemanfaatan Sumber Daya: Dengan konkurensi, sistem operasi dapat memanfaatkan sumber daya yang lebih baik, seperti CPU dan memori. Alih-alih harus menunggu satu proses selesai sebelum memulai proses lainnya, sistem operasi dapat tumpang tindih dengan eksekusi beberapa proses atau utas, memanfaatkan sumber daya yang tersedia dengan lebih baik.

Konkurensi

Manfaat *Konkurensi* #2

- Dukungan yang Lebih Baik untuk Aplikasi Interaktif: Konkurensi dapat memberikan dukungan yang lebih baik untuk aplikasi interaktif yang membutuhkan umpan balik langsung dari pengguna.
- Peningkatan Kinerja: Dengan menjalankan beberapa proses atau utas secara bersamaan, sistem operasi dapat meningkatkan kinerja sistem secara keseluruhan.

Konkurensi

Manfaat *Konkurensi* #2

- Dukungan yang Lebih Baik untuk Aplikasi Interaktif: Konkurensi dapat memberikan dukungan yang lebih baik untuk aplikasi interaktif yang membutuhkan umpan balik langsung dari pengguna.
- Peningkatan Kinerja: Dengan menjalankan beberapa proses atau utas secara bersamaan, sistem operasi dapat meningkatkan kinerja sistem secara keseluruhan.

Informasi

Hal ini telah mengarah pada pengembangan model konkurensi dan bahasa pemrograman baru yang secara khusus dirancang untuk memanfaatkan banyak core dan thread, seperti Go dan Rust.

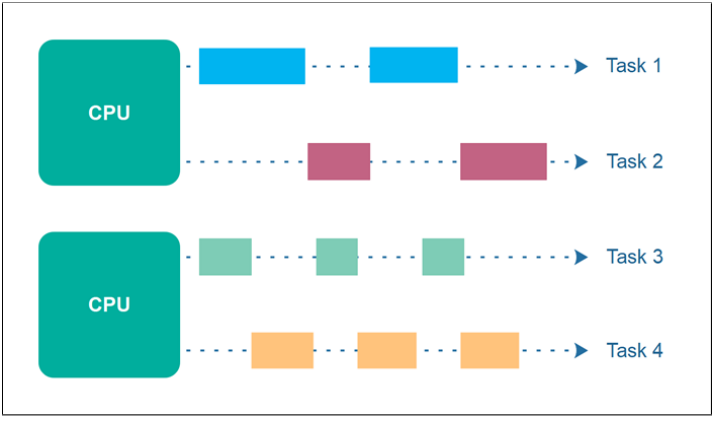
Konkurensi

Kerugian *Konkurensi* #2

- Kompleksitas: Konkurensi dapat menambah kompleksitas yang signifikan pada desain dan pengembangan perangkat lunak, terutama dalam hal menyinkronkan akses ke sumber daya bersama.
- Overhead: Overhead yang terkait dengan mekanisme sinkronisasi, seperti kunci dan semaphore, dapat menjadi signifikan.
- Kebuntuan: Kebuntuan dapat terjadi ketika beberapa proses atau thread saling menunggu untuk melepaskan sumber daya, dan tidak ada satupun dari mereka yang dapat membuat kemajuan.
- Skalabilitas: Skalabilitas sistem konkuren dapat dibatasi oleh faktor-faktor seperti perebutan sumber daya bersama dan overhead mekanisme sinkronisasi.

Konkurensi

Ilustrasi Konkurensi



Konkurensi

Konkurensi vs Paralelisme #1

Berbeda dengan Konkurensi, Paralelisme difokuskan pada pembagian tugas tunggal menjadi subtugas yang dapat dieksekusi secara paralel pada beberapa prosesor atau inti untuk meningkatkan kinerja dan kecepatan. Paralelisme membutuhkan banyak prosesor atau inti dan dapat diimplementasikan menggunakan teknik seperti algoritme paralel, bahasa pemrograman paralel, dan perangkat keras paralel.

Informasi

Secara keseluruhan, meskipun konkurensi dan paralelisme adalah konsep yang saling berkaitan, namun keduanya berbeda dalam hal fokus dan teknik yang digunakan untuk mencapai tujuan masing-masing.

Konkurensi

Konkurensi vs Paralelisme #2

Kapan pemanfaatannya?

- **Konkurensi** dapat sangat berguna ketika berhadapan dengan tugas-tugas yang memerlukan operasi input/output yang sering, seperti input pengguna atau komunikasi jaringan.
- **Paralelisme** dapat berguna saat menangani tugas yang dapat dibagi menjadi subtugas yang dapat dieksekusi secara independen dan paralel. Hal ini terutama berlaku untuk tugas yang membutuhkan sumber daya komputasi yang signifikan, seperti analisis data, simulasi, dan perhitungan ilmiah.

Konkurensi

Konkurensi vs Paralelisme #3

Konkurensi

- Server web yang menangani beberapa permintaan HTTP dari klien yang berbeda secara bersamaan
- Editor teks yang memungkinkan pengguna mengedit beberapa dokumen secara bersamaan
- Pemutar media yang memainkan musik di latar belakang saat pengguna mengerjakan tugas lain

Konkurensi

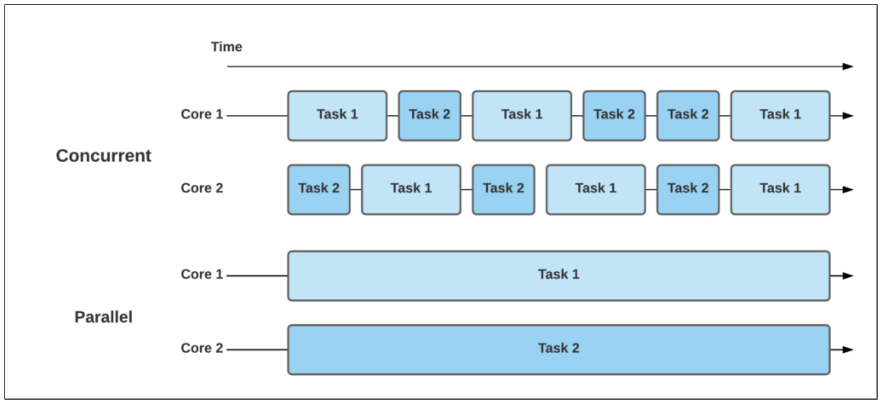
Konkurensi vs Paralelisme #4

Paralelisme

- Perangkat lunak pengolah gambar yang menggunakan beberapa inti untuk menerapkan filter dan efek pada gambar
- Alat analisis data yang membagi kumpulan data besar menjadi beberapa bagian dan memprosesnya secara paralel pada beberapa node dalam sebuah cluster
- Simulasi ilmiah yang menggunakan algoritme paralel untuk membagi simulasi besar menjadi subtugas yang lebih kecil yang dapat dieksekusi secara paralel pada beberapa prosesor

Konkurensi

Konkurensi vs Paralelisme #5



Mutual Exclusion

Mutual Exclusion

Apa itu **Mutual Exclusion**?

Mutual exclusion adalah konsep dalam sistem operasi yang mengacu pada kebutuhan untuk memastikan bahwa hanya satu proses dalam satu waktu yang dapat mengakses sumber daya bersama, seperti file, basis data, atau perangkat keras.

Mutual exclusion dapat diimplementasikan dengan menggunakan berbagai mekanisme sinkronisasi, seperti kunci, semaphore, dan monitor. Mekanisme-mekanisme ini memungkinkan sebuah proses untuk meminta akses eksklusif ke sumber daya bersama dan memblokir proses lain untuk mengaksesnya sampai proses pertama melepaskannya.

- "Operating System Concepts" by Abraham Silberschatz, Peter B. Galvin, and Greg Gagne

Mutual Exclusion

Hubungan **Mutual Exclusion** dan Konkurensi

Karena dengan konkurensi sistem operasi dapat menjalankan banyak proses sekaligus, maka dalam manajemen sumber dayanya perlu cara khusus. Oleh karena itu **Mutual Exclusion** digunakan untuk memastikan semua proses yang dijalankan sistem operasi tersebut memperoleh sumber daya yang ada di komputer.

Informasi
Mekanisme ini memastikan bahwa hanya satu proses pada satu waktu yang dapat mengakses sumber daya bersama

Mutual Exclusion

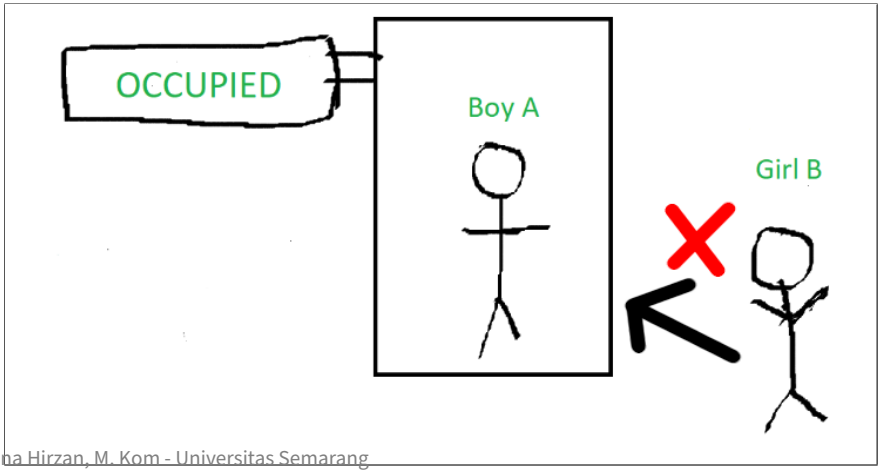
Contoh Mekanisme **Mutual Exclusion**

Mutual Exclusion dapat menggunakan mekanisme **Lock** untuk memastikan sumber daya bersama. Ketika sebuah proses ingin mengakses sumber daya, proses tersebut harus meminta kunci terlebih dahulu. Jika kunci saat ini dipegang oleh proses lain, proses yang meminta akan diblokir sampai kunci dilepaskan. Setelah kunci diperoleh, proses dapat mengakses sumber daya, dan ketika selesai, proses melepaskan kunci, sehingga proses lain dapat memperolehnya.

Semaphore memungkinkan proses untuk menunggu kondisi tertentu terpenuhi sebelum melanjutkan, sementara **Monitor** menyediakan cara bagi proses untuk mengakses sumber daya bersama sambil memberlakukan pengecualian timbal balik.

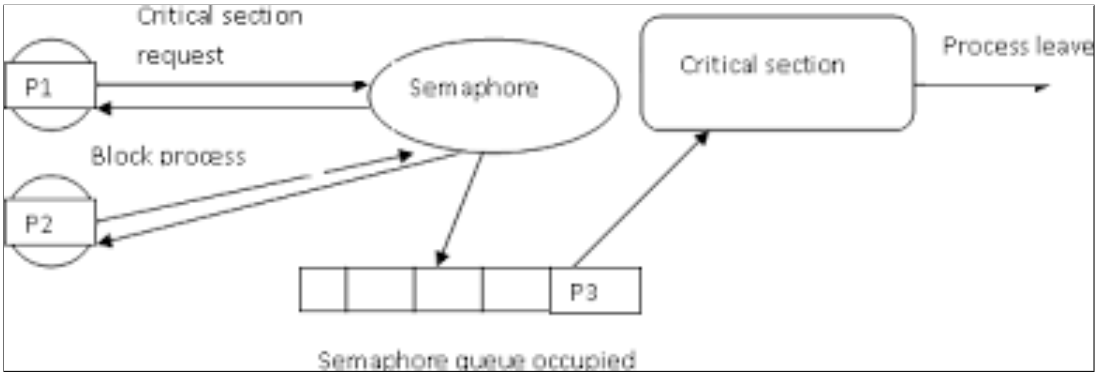
Mutual Exclusion

Mekanisme **Lock**



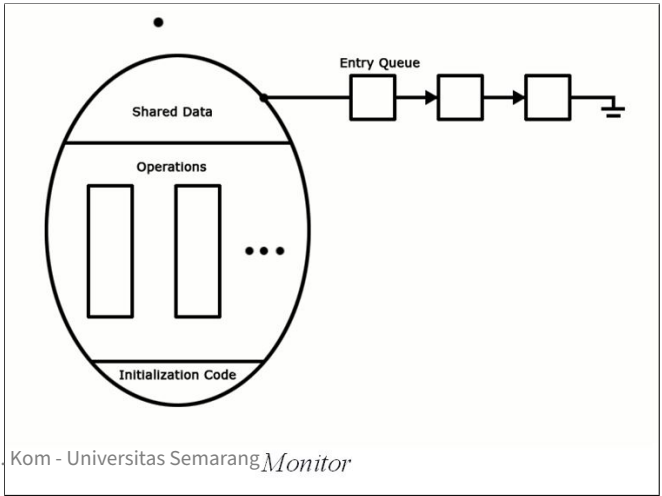
Mutual Exclusion

Mekanisme **Semaphore**



Mutual Exclusion

Mekanisme **Monitor**



Monitor

Mutual Exclusion

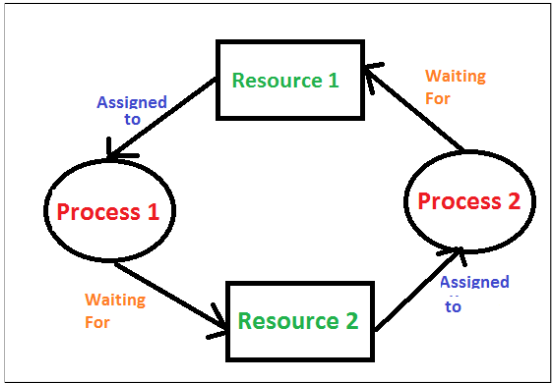
Resiko **Deadlock**

Risiko **deadlock** dapat muncul dalam sistem yang menggunakan banyak sumber daya, yang masing-masing hanya dapat dipegang oleh satu proses pada satu waktu, dan di mana proses dapat meminta sumber daya dalam urutan apa pun.

Kebuntuan dapat terjadi ketika proses memegang sumber daya dan meminta sumber daya tambahan yang saat ini dipegang oleh proses lain, menciptakan rantai melingkar dari proses yang menunggu.

Mutual Exclusion

Ilustrasi **Deadlock**



Mutual Exclusion

Mitigasi **Deadlock**

Deadlock Prevention melibatkan penataan sistem dengan cara yang membuat kebuntuan tidak mungkin terjadi.

Deadlock Avoidance melibatkan pemeriksaan dan penghindaran situasi kebuntuan secara dinamis.

Deadlock Detection and Recovery melibatkan pemeriksaan kebuntuan secara berkala dan mengambil tindakan untuk mengatasinya ketika terjadi.

Synchronization

Synchronization

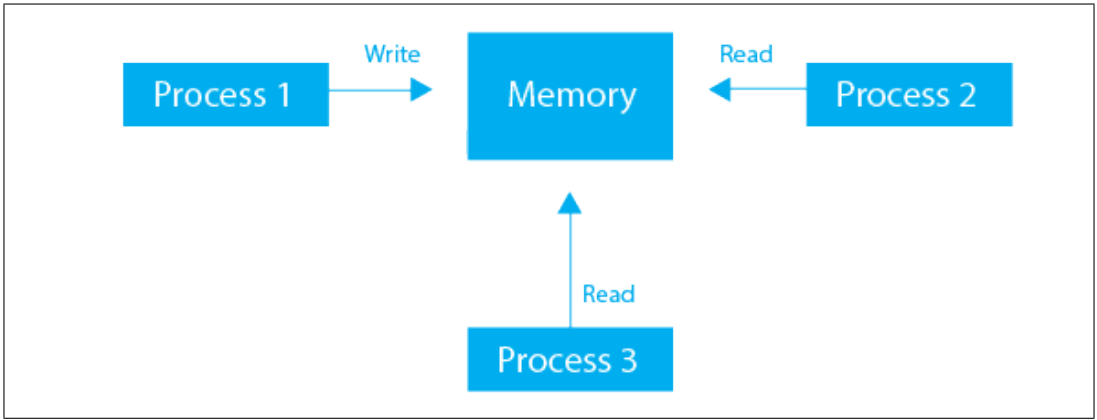
Mengenal Sinkronisasi

Sinkronisasi adalah konsep penting dalam sistem operasi yang mengacu pada koordinasi beberapa proses atau utas untuk memastikan bahwa mereka mengakses sumber daya bersama dengan cara yang aman dan teratur. Tanpa sinkronisasi yang tepat, akses bersamaan ke sumber daya bersama dapat menyebabkan kerusakan data, kondisi balapan, dan masalah lainnya.

Mekanisme sinkronisasi dapat digunakan untuk mengoordinasikan akses ke sumber daya bersama, membuat saluran komunikasi antar proses, dan mengoordinasikan pelaksanaan tugas yang bersamaan. Mekanisme ini juga dapat digunakan untuk mencegah masalah seperti kebuntuan dan kondisi balapan.

Synchronization

Ilustrasi **Sinkronisasi**



Synchronization

Cara Kerja Sinkronisasi

- **Meminta akses**: Proses atau thread yang ingin mengakses sumber daya bersama akan meminta akses ke mekanisme sinkronisasi yang terkait dengan sumber daya tersebut.
- **Memblokir atau menunggu**: Jika mekanisme sinkronisasi saat ini sedang dipegang oleh proses atau thread lain, proses atau thread yang meminta akan diblokir atau dimasukkan ke dalam kondisi menunggu hingga mekanisme tersebut tersedia.
- **Mengakses sumber daya**: Setelah mekanisme sinkronisasi tersedia, proses atau thread yang meminta akan diberikan akses ke sumber daya bersama.
- **Melepaskan sumber daya**: Ketika proses atau thread selesai menggunakan sumber daya bersama, ia akan melepaskan mekanisme sinkronisasi.


THANK YOU