Sistem Operasi

Pertemuan 05

Alauddin Maulana Hirzan, S.Kom., M.Kom.
NIDN. 0607069401

Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang

Manajemen Thread

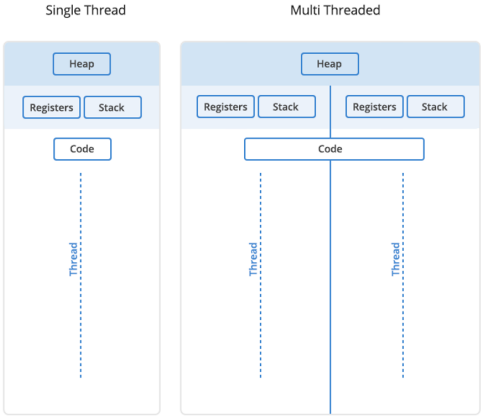
Definisi Thread

Definisi: Thread mengacu pada unit eksekusi terkecil dalam sebuah proses. Sebuah proses, secara sederhana, dapat dianggap sebagai sebuah program yang sedang dieksekusi, dan di dalam program ini, thread adalah entitas yang menjalankan instruksi.

Thread dalam proses yang sama berbagi ruang memori dan sumber daya yang sama, seperti file dan perangkat I/O, sementara setiap thread memiliki stack dan registernya sendiri.

Manajemen Thread

Definisi Thread



Manajemen Thread

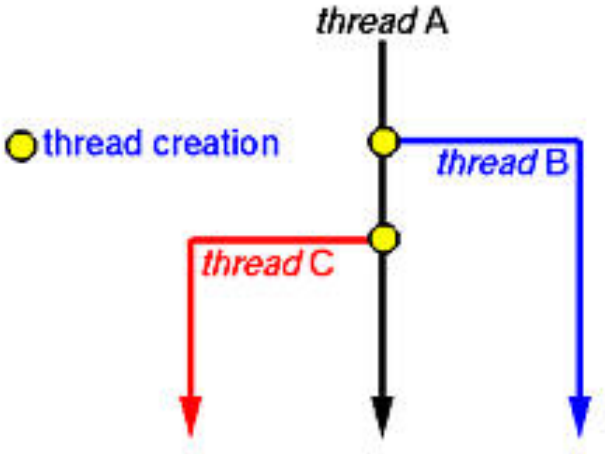
Cara Kerja Thread

1. Pembuatan Thread:
Thread dapat dibuat oleh sistem operasi atau oleh suatu proses. Sistem operasi menyediakan panggilan sistem atau API untuk membuat dan mengelola thread. Misalnya, dalam sistem seperti Unix, fungsi pthread_create () biasanya digunakan untuk membuat thread.

Ketika sebuah thread dibuat, thread tersebut berbagi sumber daya proses, termasuk memori, pegangan file, dan struktur kernel lainnya.

Manajemen Thread

Cara Kerja Thread



Manajemen Thread

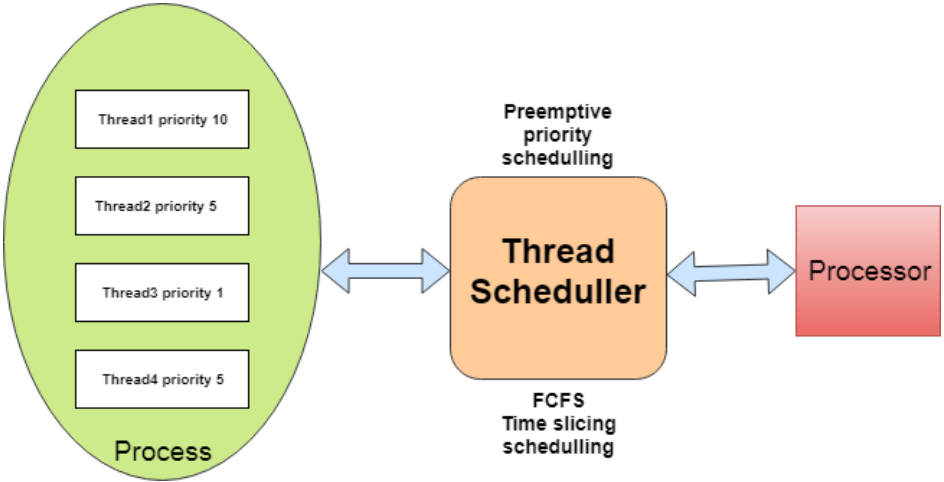
Cara Kerja Thread

2. Penjadwalan Thread:
Penjadwalan thread adalah proses menentukan thread mana yang harus dieksekusi berikutnya pada CPU. Penjadwal OS bertanggung jawab untuk memilih thread dari antrian siap dan mengirimkannya untuk dieksekusi pada CPU.

Algoritme penjadwalan thread meliputi **First Come First Serve (FCFS)**, **Round Robin**, **Shortest Job Next (SJN)**, dan **Penjadwalan Berbasis Prioritas**.

Manajemen Thread

Cara Kerja Thread



Manajemen Thread

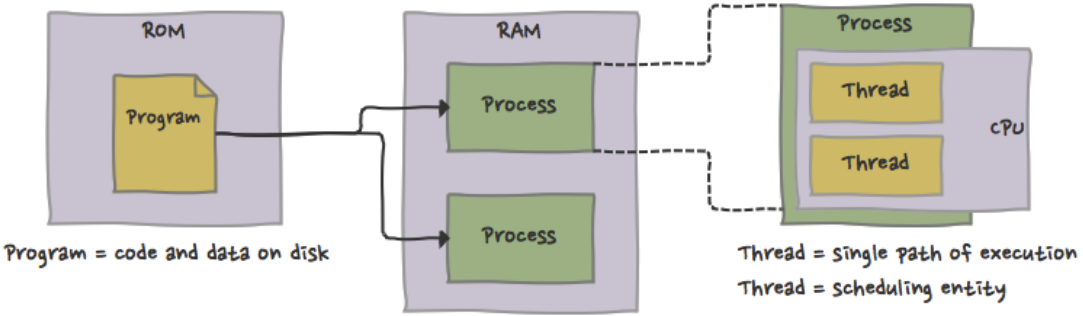
Cara Kerja Thread

3. Eksekusi Thread:
Ketika sebuah thread dijadwalkan untuk dieksekusi, penghitung program (PC) dimuat dengan alamat instruksi berikutnya yang akan dieksekusi. Thread mengeksekusi instruksi secara berurutan hingga mencapai operasi pemblokiran atau menyerahkan CPU secara sukarela.

Jika thread diblokir, thread akan berpindah ke status diblokir, dan thread lain dipilih untuk dieksekusi.

Manajemen Thread

Cara Kerja Thread



Manajemen Thread

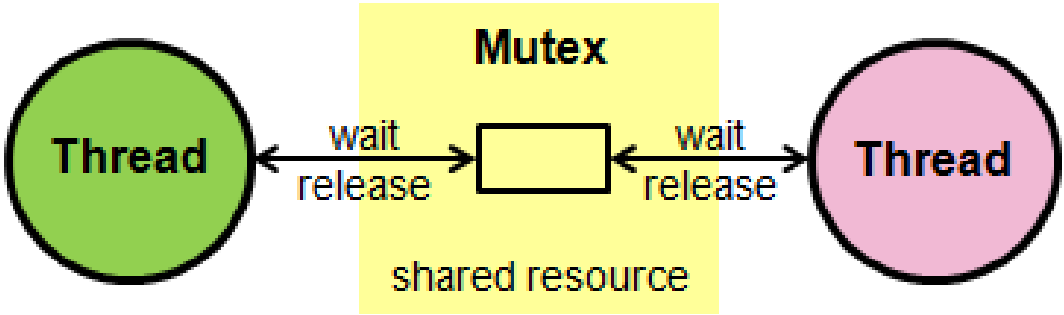
Cara Kerja Thread

4. Komunikasi dan Sinkronisasi Thread:
Thread dalam proses yang sama dapat berkomunikasi dan melakukan sinkronisasi satu sama lain melalui memori bersama. Mekanisme sinkronisasi thread seperti mutex, semaphore, dan variabel kondisi digunakan untuk mengoordinasikan akses ke sumber daya bersama dan mencegah kondisi balapan.

Sinkronisasi memastikan bahwa bagian kode yang penting dieksekusi secara atomik, tanpa gangguan dari thread lain.

Manajemen Thread

Cara Kerja Thread



Manajemen Thread

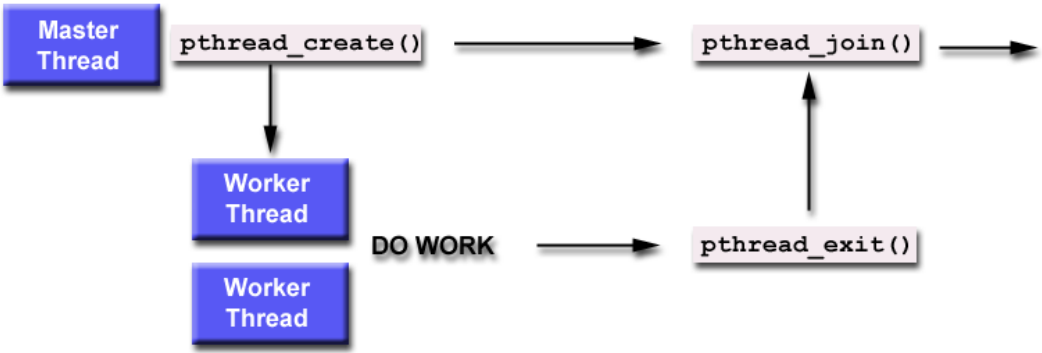
Cara Kerja Thread

5. Pengakhiran Thread:
Thread dapat diakhiri baik secara sukarela dengan memanggil fungsi exit atau secara tidak sukarela karena kesalahan atau selesainya tugas thread.

Ketika sebuah thread diakhiri, sumber dayanya akan didealokasikan, dan status penghentiannya akan tersedia untuk thread lain atau proses induk.

Manajemen Thread

Cara Kerja Thread



Manajemen Thread

Jenis Thread

Thread memiliki jenis:

- **Thread Tingkat Pengguna (User Level Thread)**:
  - Thread tingkat pengguna dikelola sepenuhnya oleh pustaka tingkat pengguna atau aplikasi tanpa dukungan kernel.
  - ULT ringan karena tidak memerlukan keterlibatan kernel untuk manajemen thread.
  - ULT biasanya lebih cepat dibuat dan dikelola dibandingkan dengan thread tingkat kernel.
  - ULT memiliki keterbatasan tidak dapat berjalan secara bersamaan di beberapa core.

Manajemen Thread

Jenis Thread

- **Thread Tingkat Kernel (Kernel-Level Threads, KLT):**
  - Thread tingkat kernel dikelola secara langsung oleh kernel sistem operasi.
  - Setiap thread diwakili oleh struktur data kernel yang terpisah, sehingga memungkinkan kernel untuk mengelolanya satu per satu.
  - KLT dapat berjalan secara bersamaan pada beberapa inti prosesor, sehingga mencapai paralelisme yang lebih baik dan memanfaatkan prosesor multi-inti secara efisien.
  - Perpindahan konteks antara KLT biasanya melibatkan lebih banyak overhead dibandingkan dengan ULT karena keterlibatan kernel.

Manajemen Thread

Jenis Thread

- **Thread Hibrida**:
  - Thread hibrida menggabungkan aspek-aspek ULT dan KLT.
  - Dalam model thread hybrid, setiap thread tingkat pengguna dipetakan ke satu atau lebih thread tingkat kernel.
  - Pustaka thread tingkat pengguna mengelola pemetaan antara thread tingkat pengguna dan tingkat kernel, memberikan fleksibilitas manajemen thread tingkat pengguna dengan manfaat kinerja dari thread tingkat kernel.
  - Pendekatan ini memungkinkan pemanfaatan prosesor multi-core yang lebih baik sambil tetap menyediakan pembuatan dan manajemen thread yang ringan.

Manajemen Thread

Jenis Thread

- **Thread Waktu Nyata**:
  - Thread waktu nyata dirancang untuk memenuhi persyaratan waktu tertentu, di mana memenuhi tenggat waktu sangat penting.
  - Thread ini sering digunakan dalam sistem waktu nyata seperti sistem kontrol, di mana respons yang tepat waktu sangat penting.
  - Thread waktu nyata biasanya diprioritaskan di atas thread non-waktu nyata untuk memastikan eksekusi yang tepat waktu.

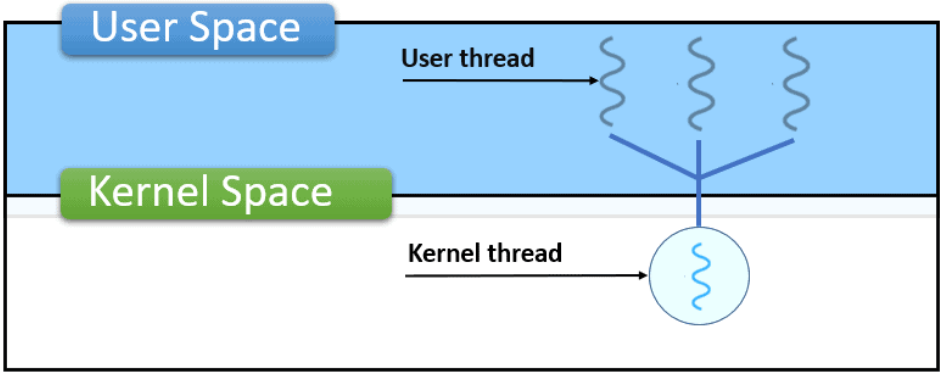
Manajemen Thread

Jenis Thread

- **Thread Daemon**:
  - Daemon thread adalah thread latar belakang yang berjalan di latar belakang dan tidak penting untuk fungsi program.
  - Thread ini biasanya digunakan untuk tugas-tugas seperti pengumpulan sampah, pemeliharaan berkala, atau pemrosesan latar belakang.
  - Thread daemon secara otomatis dihentikan ketika semua benang non-daemon dalam program telah keluar.

Manajemen Thread

Jenis Thread



Thread Pools

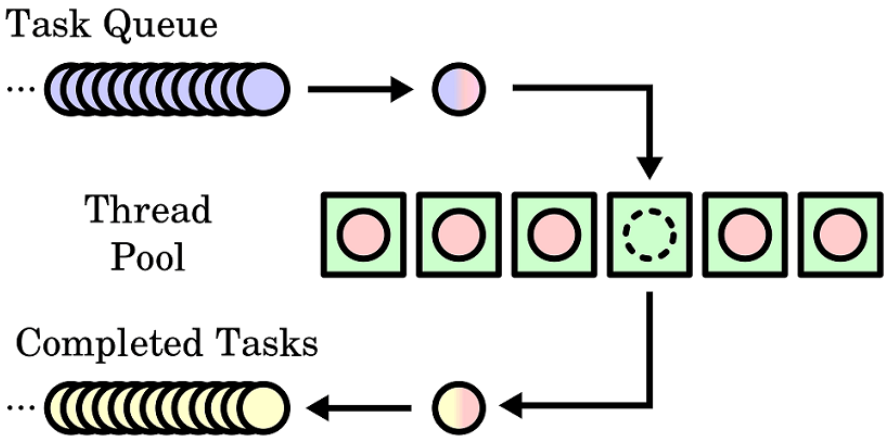
Definisi Thread Pools

Definisi:
Thread pool adalah konsep pemrograman dan mekanisme yang digunakan untuk mengelola dan mengontrol siklus hidup thread dalam aplikasi multithread. Alih-alih membuat thread baru setiap kali tugas perlu dieksekusi secara bersamaan, thread pool mempertahankan kumpulan thread yang dapat digunakan kembali.

Thread ini telah dialokasikan sebelumnya dan dikelola oleh thread pool, yang memberikan tugas sesuai kebutuhan.

Thread Pools

Definisi Thread Pools



Thread Pools

Cara Kerja Thread Pools

1. **Inisialisasi**: Saat kumpulan thread dibuat, biasanya akan menginisialisasi jumlah thread yang tetap atau secara dinamis menyesuaikan jumlahnya
2. **Pengiriman Tugas**: Tugas (juga dikenal sebagai pekerjaan atau unit kerja) dikirimkan ke kumpulan thread untuk dieksekusi. Tugas-tugas ini dapat berupa unit kerja apa pun yang dapat dieksekusi secara indepen-den
3. **Antrian Tugas**: Thread pool biasanya memiliki antrean tugas (juga disebut antrean kerja atau buffer tugas) untuk menampung tugas yang menunggu untuk dieksekusi

Thread Pools

Cara Kerja Thread Pools

4. **Alokasi Thread**: Kumpulan thread memberikan tugas dari antrean ke thread yang tersedia dalam kumpulan untuk dieksekusi. Jika semua thread sibuk, tugas akan menunggu dalam antrean sampai sebuah thread tersedia.
5. **Eksekusi**: Setiap thread dalam pool mengeksekusi tugas yang diberikan kepadanya secara berurutan. Setelah sebuah thread menyelesaikan sebuah tugas, thread tersebut akan kembali ke pool dan tersedia untuk eksekusi tugas lain.
6. **Pematian**: Ketika aplikasi atau sistem dimatikan, thread pool dapat dimatikan secara perlahan, melepaskan semua sumber dayanya dan mengakhiri thread-nya.

Thread Pools

Manfaat Thread Pools

1. **Manajemen Sumber Daya**: Thread pool secara efisien mengelola sumber daya sistem dengan menggunakan kembali thread, menghindari overhead pembuatan dan penghancuran thread.
2. **Kontrol Konkurensi**: Thread pool menyediakan pendekatan terstruktur untuk mengelola konkurensi dengan membatasi jumlah thread yang bersamaan.
3. **Penyeimbangan Beban**: Kumpulan thread dapat mendistribusikan tugas secara merata di antara thread yang tersedia, memastikan bahwa beban kerja seimbang di seluruh sistem.

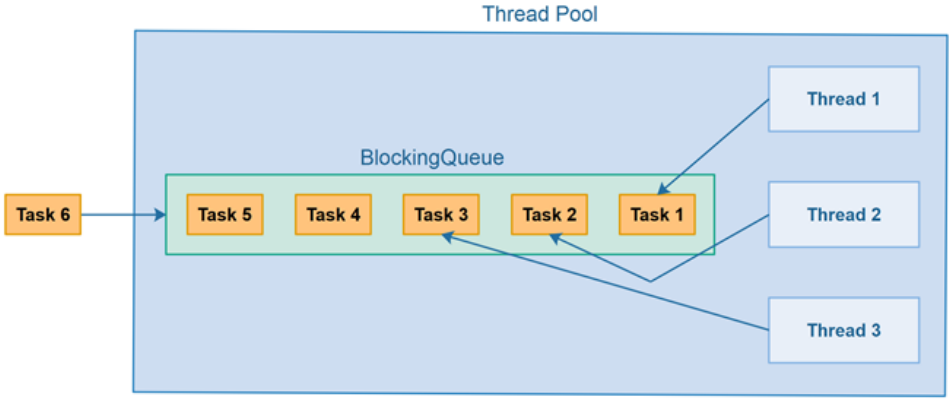
Thread Pools

Manfaat Thread Pools

4. **Peningkatan Daya Tanggap**: Dengan menyimpan kumpulan thread yang siap digunakan, thread pool dapat mengurangi waktu pengaktifan tugas
5. **Skalabilitas**: Thread pool dapat dikonfigurasi untuk menyesuaikan jumlah thread secara dinamis berdasarkan beban kerja atau kondisi sistem
6. **Kontrol atas Batas Sumber Daya**: Thread pool memungkinkan pengembang untuk menentukan jumlah maksimum thread

Thread Pools

Manfaat Thread Pools



CPU Bound vs. I/O Bound Threads

Definisi CPU Bound

CPU Bound menghabiskan waktunya untuk melakukan komputasi atau tugas pemrosesan yang sangat memanfaatkan sumber daya CPU. Tugas-tugas ini melibatkan perhitungan intensif, manipulasi data, atau operasi algoritmik yang kompleks.

Memiliki ciri-ciri:

1. **Pemanfaatan CPU yang tinggi**: Thread yang terikat pada CPU secara terus menerus menggunakan sumber daya CPU selama eksekusi.
2. **Waktu Tunggu Singkat**: Thread ini biasanya menghabiskan waktu minimal untuk menunggu sumber daya eksternal, seperti operasi I/O atau komunikasi jaringan.

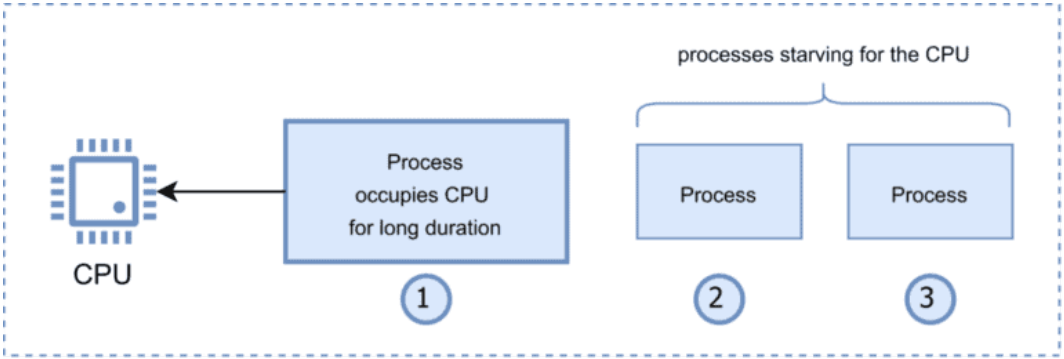
CPU Bound vs. I/O Bound Threads

Definisi CPU Bound

3. **Waktu Eksekusi yang Lama**: Tugas yang terikat pada CPU umumnya memiliki waktu eksekusi yang lebih lama karena kompleksitas komputasi yang terlibat.
4. **Operasi I / O Terbatas**: Karena fokusnya adalah pada komputasi, thread yang terikat CPU melakukan operasi I/O yang relatif lebih sedikit dibandingkan dengan jenis thread lainnya.
5. **Potensi Kelaparan Sumber Daya**: Jika tidak dikelola dengan benar, thread yang terikat CPU dapat membuat thread atau proses lain kekurangan sumber daya CPU, yang menyebabkan penurunan kinerja atau perlambatan sistem.

CPU Bound vs. I/O Bound Threads

Definisi CPU Bound



CPU Bound vs. I/O Bound Threads

Definisi I/O Bound

I/O Bound sebagian besar menghabiskan waktunya untuk menunggu operasi input/output (I/O) selesai, seperti membaca dari atau menulis ke file, mengakses database, atau berkomunikasi melalui jaringan.

Memiliki ciri-ciri:

1. **Waktu Tunggu I/O Tinggi**: Thread yang terikat I/O menghabiskan sebagian besar waktu eksekusi untuk menunggu operasi I/O selesai.
2. **Pemanfaatan CPU yang rendah**: Karena sebagian besar waktu dihabiskan untuk menunggu sumber daya eksternal, thread yang terikat I/O menggunakan sumber daya CPU lebih sedikit dibandingkan dengan thread yang terikat CPU.

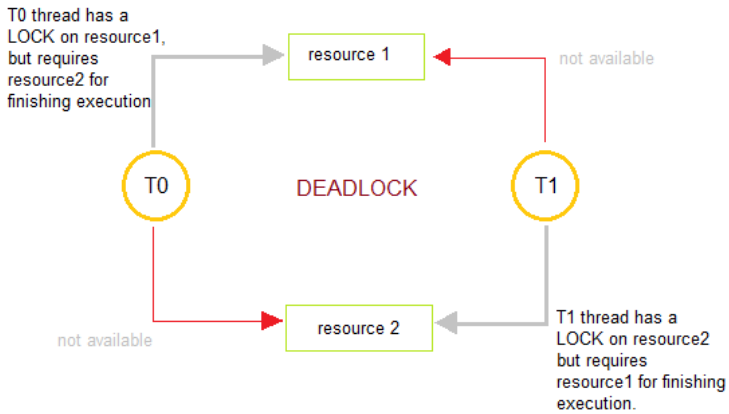
CPU Bound vs. I/O Bound Threads

Definisi I/O Bound

3. **Waktu Eksekusi Singkat**: Tugas yang terikat I/O biasanya memiliki waktu eksekusi yang lebih singkat setelah operasi I/O yang diperlukan selesai.
4. **Sering Memblokir**: Thread yang terikat I/O sering kali memblokir atau ditangguhkan saat menunggu operasi I/O selesai, sehingga memungkinkan thread lain untuk menggunakan sumber daya CPU untuk sementara waktu.
5. **Potensi Kebuntuan**: Karena seringnya terjadi pemblokiran, thread yang terikat I/O mungkin rentan terhadap situasi kebuntuan jika mekanisme sinkronisasi yang tepat tidak diterapkan.

CPU Bound vs. I/O Bound Threads

Definisi I/O Bound



Terima Kasih