PEMROSESAN PARALEL

Disusun Oleh:

NAMA	**:** FARCA RIZQI A.
NIM	**:** 09011282126082
KELAS	**:** SISTEM KOMPUTER 5B INDRALAYA
DOSEN PENGAMPU	**:** AHMAD HERYANTO, S.Kom., M.T..
	ADI HERMANSYAH, M.T..

PROGRAM STUDI SISTEM KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJAYA

TAHUN AJARAN 2023/2024

TUGAS:

1. Apa perbedaan antara multiprosessing dan multithreading?
2. Buat contoh program multiprosessing dan multithreading dan tunjukan setiap item perbedaan pada point 1?

Multiprocessing dan multithreading adalah dua pendekatan yang berbeda untuk mencapai konkurensi dalam pemrograman. Berikut adalah perbedaan utama antara keduanya:

1. **Unit Eksekusi**:
   - **Multiprocessing**: Dalam multiprocessing, program dibagi menjadi beberapa proses independen. Setiap proses memiliki memori dan sumber daya yang terpisah. Proses-proses ini berjalan secara parallel dan dapat menjalankan tugas-tugas mereka sendiri tanpa tergantung pada proses lain. Masing-masing proses memiliki ruang alamatnya sendiri.
   - **Multithreading**: Dalam multithreading, program memiliki satu proses utama yang berisi beberapa thread yang berbagi memori dan sumber daya yang sama. Thread adalah unit kecil dari proses yang berbagi memori dan konteks eksekusi. Thread-thread ini dapat berkomunikasi satu sama lain lebih mudah karena mereka berada dalam konteks yang sama.
2. **Overhead**:
   - **Multiprocessing**: Pembuatan dan pengelolaan proses memerlukan overhead yang lebih besar dibandingkan dengan pembuatan thread. Karena setiap proses memiliki memori dan sumber daya yang terpisah, ada overhead tambahan dalam hal manajemen memori dan komunikasi antar proses.
   - **Multithreading**: Pembuatan dan pengelolaan thread memiliki overhead yang lebih rendah karena mereka berbagi memori dan sumber daya yang sama. Namun, Anda perlu berhati-hati dalam menghindari kondisi balapan (race conditions) dan masalah sinkronisasi yang dapat timbul dalam multithreading.
3. **Komunikasi dan Sinkronisasi**:
   - **Multiprocessing**: Komunikasi antar proses biasanya dilakukan melalui mekanisme seperti antrean (queue), proses berbagi (shared memory), atau komunikasi jarak jauh (remote communication). Sementara itu, proses-proses ini tidak memerlukan sinkronisasi khusus karena mereka terpisah secara alami.
   - **Multithreading**: Komunikasi antar thread dalam satu proses dapat dilakukan dengan lebih mudah karena mereka berbagi memori yang sama. Namun, Anda harus menggunakan mekanisme sinkronisasi seperti mutex (lock) atau semafor (semaphore) untuk menghindari masalah seperti race conditions.
4. **Kemampuan Skalabilitas**:
   - **Multiprocessing**: Multiprocessing lebih cocok untuk tugas-tugas yang memerlukan pemrosesan berat secara parallel pada beberapa core atau CPU yang berbeda. Ini cocok untuk masalah-masalah terkait dengan CPU-bound.
   - **Multithreading**: Multithreading lebih cocok untuk tugas-tugas yang memerlukan I/O bound, seperti operasi file atau jaringan, di mana aplikasi dapat tetap aktif sambil menunggu operasi I/O selesai.

A. Multiprocessing

In [7]:
```
import multiprocessing

def worker_process(number):
    print(f'Proses {number} sedang berjalan')

if __name__ == '__main__':
    processes = []
    for i in range(5):
        process = multiprocessing.Process(target=worker_process, args=(i,))
        processes.append(process)
        process.start()

    for process in processes:
        process.join()

    print('Semua proses telah selesai.')
```

```
Semua proses telah selesai.
```

Dalam program ini, menggunakan modul **multiprocessing** Python untuk membuat lima proses yang berjalan secara parallel. Setiap proses menjalankan fungsi **worker_process**. Program ini akan mencetak pesan untuk setiap proses yang berjalan.

B.Multithreading

In [8]:
```
import threading

def worker_thread(number):
    print(f'Thread {number} sedang berjalan')

if __name__ == '__main__':
    threads = []
    for i in range(5):
        thread = threading.Thread(target=worker_thread, args=(i,))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()

    print('Semua thread telah selesai.')
```

```
Thread 0 sedang berjalan
Thread 1 sedang berjalan
Thread 2 sedang berjalan
Thread 3 sedang berjalan
Thread 4 sedang berjalan
Semua thread telah selesai.
```

Dalam program ini, menggunakan modul **threading** Python untuk membuat lima thread yang berjalan dalam satu proses. Setiap thread menjalankan fungsi **worker_thread**. Program ini akan mencetak pesan untuk setiap thread yang berjalan.
Dalam kedua contoh ini, menggunakan **start()** untuk memulai proses atau thread, dan **join()** untuk menunggu sampai semua proses atau thread selesai.

Perbedaannya

Unit Eksekusi:

Multiprosessing: Menggunakan proses sebagai unit eksekusi, setiap proses berdiri sendiri.
Multithreading: Menggunakan thread sebagai unit eksekusi, thread berbagi memori dalam proses yang sama.

Komunikasi:

Multiprosessing: Komunikasi antar proses lebih sulit dan memerlukan mekanisme khusus karena proses tidak berbagi memori.
Multithreading: Komunikasi antar thread lebih mudah karena thread berbagi memori, sehingga dapat berkomunikasi melalui variabel bersama.

Overhead:

Multiprosessing: Memiliki overhead yang lebih tinggi karena setiap proses memiliki salinan kode program, ruang memori, dan sumber daya yang terpisah.
Multithreading: Memiliki overhead yang lebih rendah karena thread berbagi sumber daya yang sama dalam proses tunggal, sehingga overheadnya lebih kecil.

Keamanan:

Multiprosessing: Lebih aman secara alami karena setiap proses beroperasi dalam lingkungan yang terpisah.

Multithreading: Memerlukan manajemen perangkat lunak yang cermat, seperti penguncian (locking), untuk menghindari konflik antar thread.

Skalabilitas:

Multiprosessing: Lebih cocok untuk tugas-tugas yang dapat dijalankan secara independen, cocok untuk mesin multiprosesor atau komputer dengan banyak inti.

Multithreading: Lebih cocok untuk tugas-tugas yang memerlukan akses kecepatan tinggi ke memori bersama, seperti operasi I/O.

Manajemen Sumber Daya:

Multiprosessing: Memungkinkan isolasi yang kuat antara proses, sehingga satu proses yang gagal tidak memengaruhi proses lain.

Multithreading: Thread harus berbagi sumber daya, yang dapat menyebabkan konflik dan kesulitan dalam manajemen sumber daya bersama.

Debugging:

Multiprosessing: Lebih mudah untuk melakukan debugging karena setiap proses memiliki ruang memori terpisah dan dapat diawasi secara independen.

Multithreading: Debugging dapat lebih rumit karena thread berbagi memori, sehingga lebih sulit untuk mengisolasi masalah.