PROCESS DAN THREAD

Pertemuan 3

KONSEP PROSES

- Program yang sedang dieksekusi
- Proses tidak hanya sekedar suatu kode program (*text section*), melainkan meliputi beberapa aktivitas yang bersangkutan seperti *program counter* dan *stack*.
- Sebuah proses juga melibatkan *stack* yang berisi data sementara (parameter fungsi/metode, *return address*, dan variabel lokal) dan data section yang menyimpan variabel-variabel global.

KONSEP PROSES (CONT.)

- Proses adalah sebuah program yang dieksekusi yang mencakup *program counter*, register, dan variabel di dalamnya.
- Sistem Operasi mengeksekusi proses dengan dua cara yaitu *Batch System* yang mengeksekusi *jobs* dan *Time-shared System* yang mengatur pengeksekusian program pengguna (*user*) atau *tasks*.

KONSEP PROSES (CONT.)

- Sistem operasi *UNIX* mempunyai *system call fork* yang berfungsi untuk membuat proses baru
- Proses yang memanggil *system call fork* ini akan dibagi jadi dua, proses induk dan proses turunan yang identik.

TERMINASI PROSES

- Suatu proses diterminasi ketika proses tersebut telah selesai mengeksekusi perintah terakhir serta meminta sistem operasi untuk menghapus perintah tersebut dengan menggunakan *system call* exit.
- Proses dapat mengembalikan data keluaran kepada proses induk-nya melalui *system call* wait

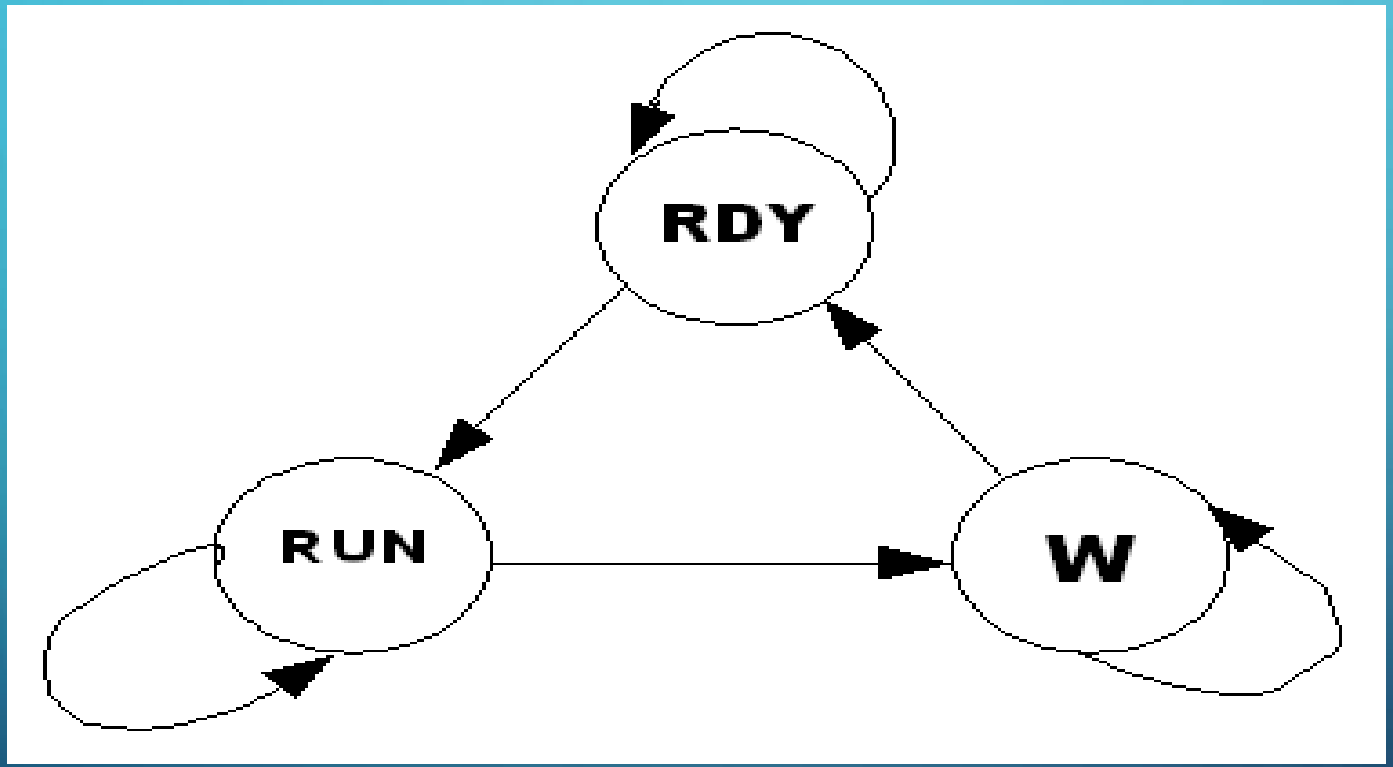
STATUS PROSES

- *Running*: status yang dimiliki pada saat instruksi-instruksi dari sebuah proses dieksekusi.
- *Waiting*: status yang dimiliki pada saat proses menunggu suatu sebuah *event* seperti proses M/K.
- *Ready*: status yang dimiliki pada saat proses siap untuk dieksekusi oleh prosesor.

STATUS PROSES (CONT.)

- *New*: status yang dimiliki pada saat proses baru saja dibuat.
- *Terminated*: status yang dimiliki pada saat proses telah selesai dieksekusi.

STATUS PROSES (CONT.)



RDY (Ready), RUN (Running), W (Wait).

PROCESS CONTROL BLOCK

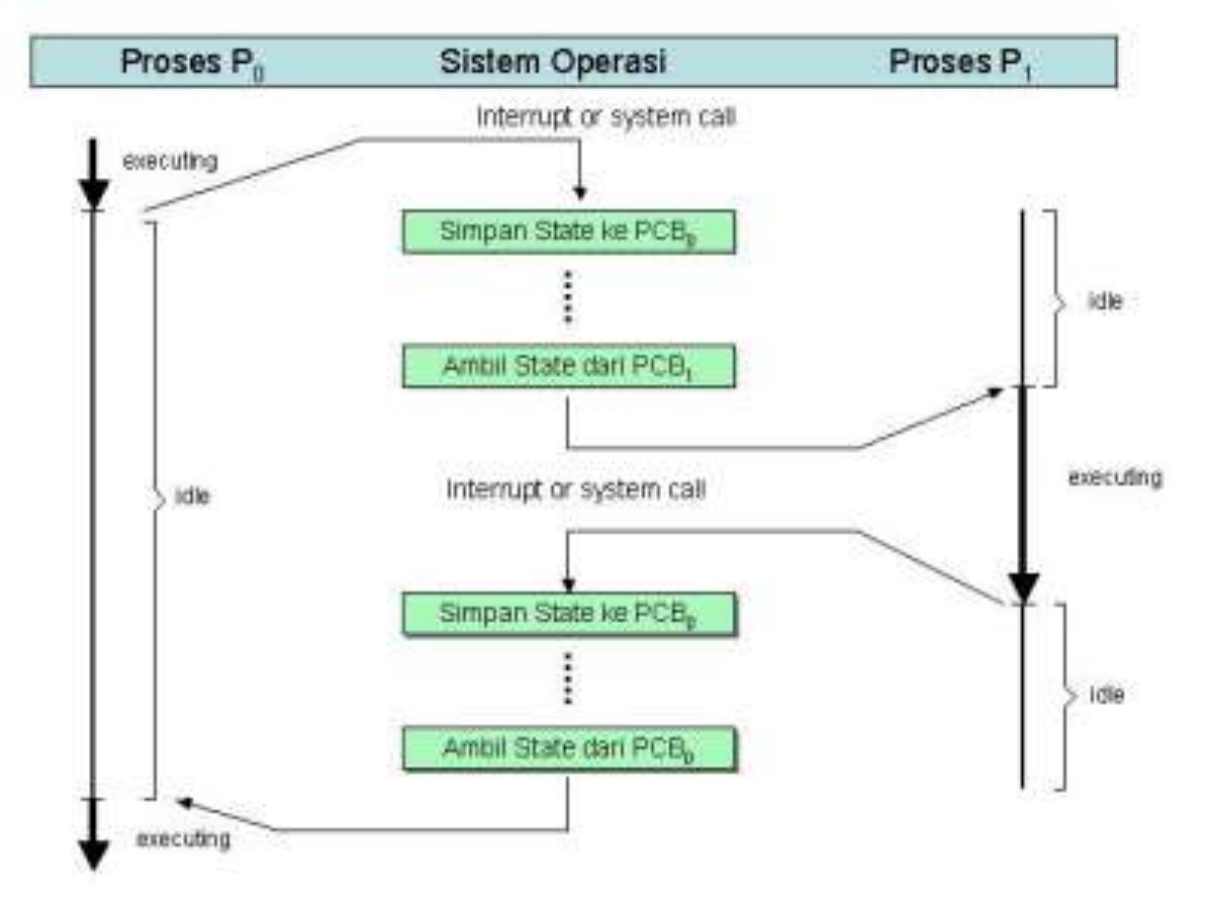
Setiap proses digambarkan dalam sistem operasi oleh sebuah ***process control block*** (PCB) – juga disebut sebuah *control block*.

Gambar Process Control Block

PROCESS CONTROL BLOCK (CONT.)

- PCB berisikan banyak bagian dari informasi yang berhubungan dengan sebuah proses yang spesifik, termasuk hal-hal di bawah ini:
  - ❑ Status Proses
  - ❑ Program counter
  - ❑ CPU Register
  - ❑ Informasi Manajemen Memori
  - ❑ Informasi pencatatan

PROCESS CONTROL BLOCK (CONT.)

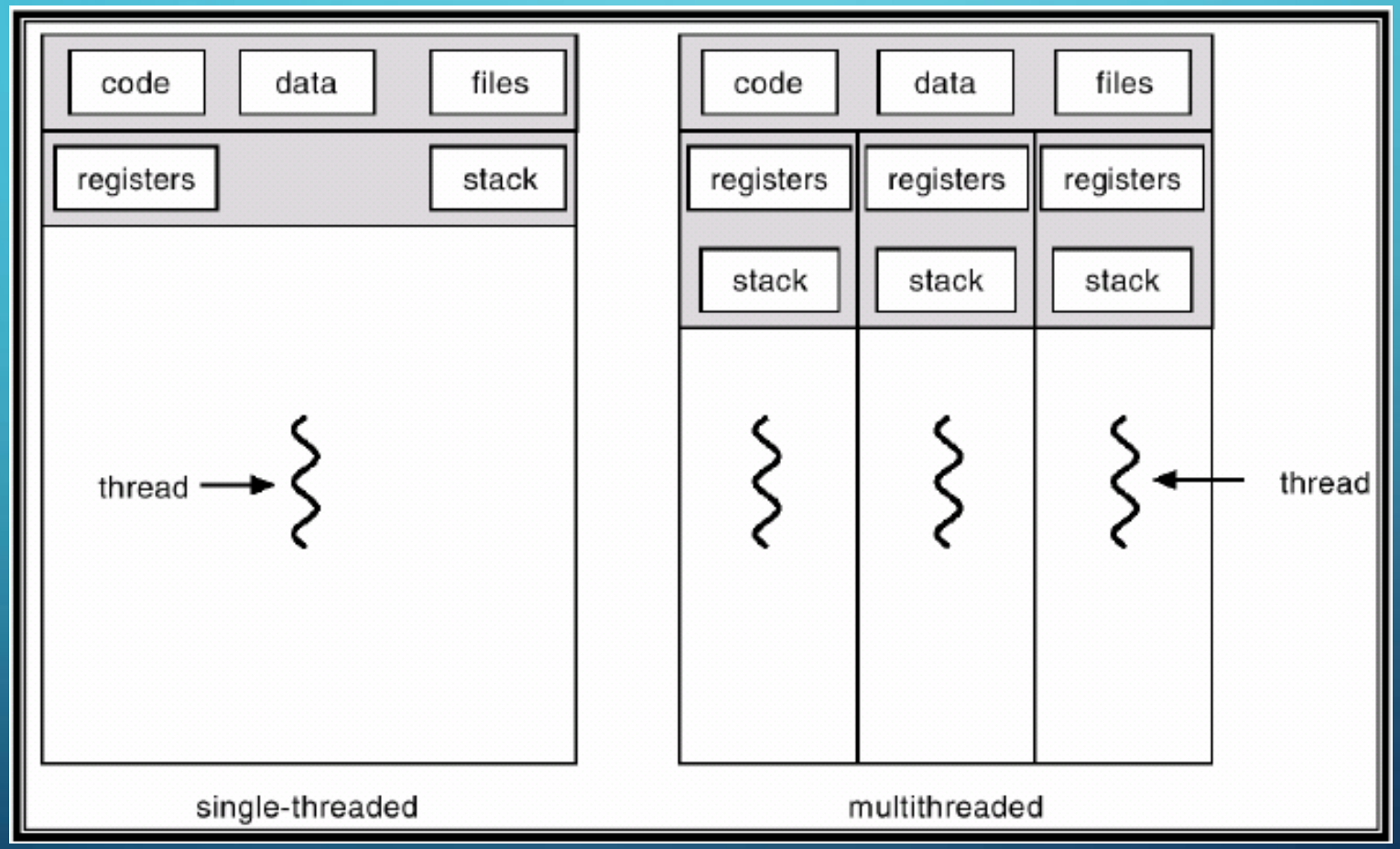


Gambar : Diagram menunjukkan beralih CPU dari proses ke proses

PROSES THREAD

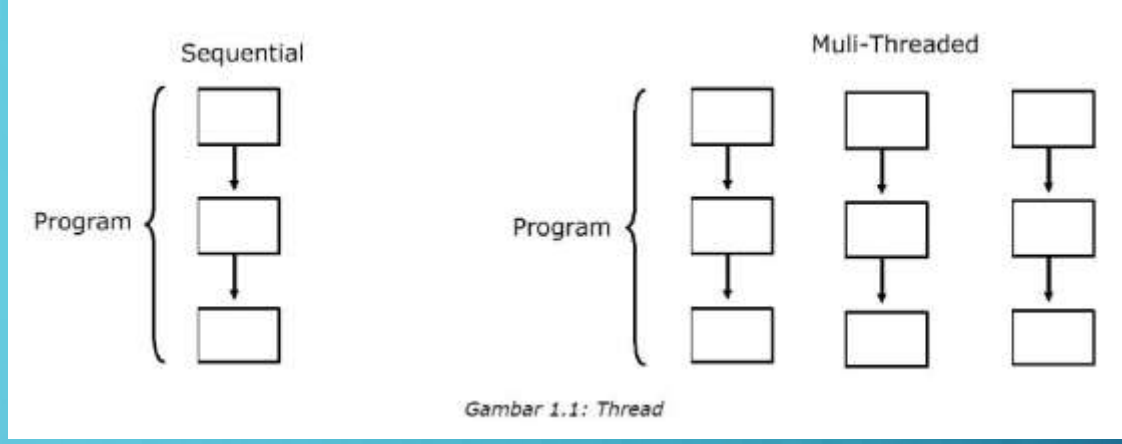
- *Thread* adalah sebuah alur kontrol dari sebuah proses. Kontrol *thread* tunggal ini hanya memungkinkan proses untuk menjalankan satu tugas pada satu waktu.

PROSES THREAD



Gambar : Proses thread

CONTOH MULTI-THREADING



Gambar 1.1: Thread

- Banyak perangkat lunak yang berjalan pada PC modern dirancang secara *multi-threading*.
- Sebuah aplikasi biasanya diimplementasi sebagai proses yang terpisah dengan beberapa *thread yang berfungsi sebagai pengendali*.
- Contohnya sebuah *web browser* mempunyai *thread* untuk menampilkan gambar atau tulisan sedangkan *thread* yang lain berfungsi sebagai penerima data dari network.

CONTOH MULTI-THREADING

- Situasi dimana sebuah aplikasi diperlukan untuk menjalankan beberapa tugas yang serupa
- Contohnya adaah sebuah *web server* yang dapat mempunyai ratusan klien yang mengaksesnya secara *concurrent*
- Kalau *web server berjalan sebagai proses yang hanya mempunyai thread tunggal maka ia hanya dapat melayani* satu klien pada pada satu satuan waktu.
- Bila ada klien lain yang ingin mengajukan permintaan maka ia harus menunggu sampai klien sebelumnya selesai dilayani.
- Solusinya adalah dengan membuat *web server menjadi multi-threading.*
- *Dengan ini maka sebuah web server akan membuat thread yang akan mendengar permintaan klien, ketika permintaan lain diajukan maka web server akan menciptakan thread lain yang akan melayani permintaan tersebut.*

THREAD DALAM PROCESS

- Perbedaan antara proses dengan *thread* tunggal dengan proses dengan *thread* yang banyak adalah proses dengan *thread* yang banyak dapat mengerjakan lebih dari satu tugas pada satu satuan waktu

KEUNTUNGAN THREAD

- Responsi
- Berbagi Sumber Daya
- Ekonomi
- **Utilisasi arsitektur *multiprocessor***

KEUNTUNGAN *THREAD*

1. **Responsi** : Membuat aplikasi yang interaktif menjadi *multithreading* dapat membuat sebuah program terus berjalan meskipun sebagian dari program tersebut diblok atau melakukan operasi yang panjang, karena itu dapat meningkatkan respons kepada pengguna. Sebagai contohnya dalam *web browser* yang *multithreading*, sebuah *thread* dapat melayani permintaan pengguna sementara *thread* lain berusaha menampilkan image.
2. **Berbagi sumber daya** : thread berbagi memori dan sumber daya dengan thread lain yang dimiliki oleh proses yang sama. Keuntungannya adalah mengizinkan sebuah aplikasi untuk mempunyai beberapa thread yang berbeda dalam lokasi memori yang sama.

KEUNTUNGAN *THREAD*

3. **Ekonomi** : dalam pembuatan sebuah proses banyak dibutuhkan pengalokasian memori dan sumber daya. Alternatifnya adalah dengan penggunaan *thread*, karena *thread* berbagi memori dan sumber daya proses yang memilikinya maka akan lebih ekonomis untuk membuat dan *context switch thread*. Akan susah untuk mengukur perbedaan waktu antara proses dan *thread* dalam hal pembuatan dan pengaturan, tetapi secara umum pembuatan dan pengaturan proses lebih lama dibandingkan *thread*. Pada Solaris, pembuatan proses lebih lama 30 kali dibandingkan pembuatan *thread*, dan *context switch* proses 5 kali lebih lama dibandingkan *context switch thread*.

KEUNTUNGAN *THREAD*

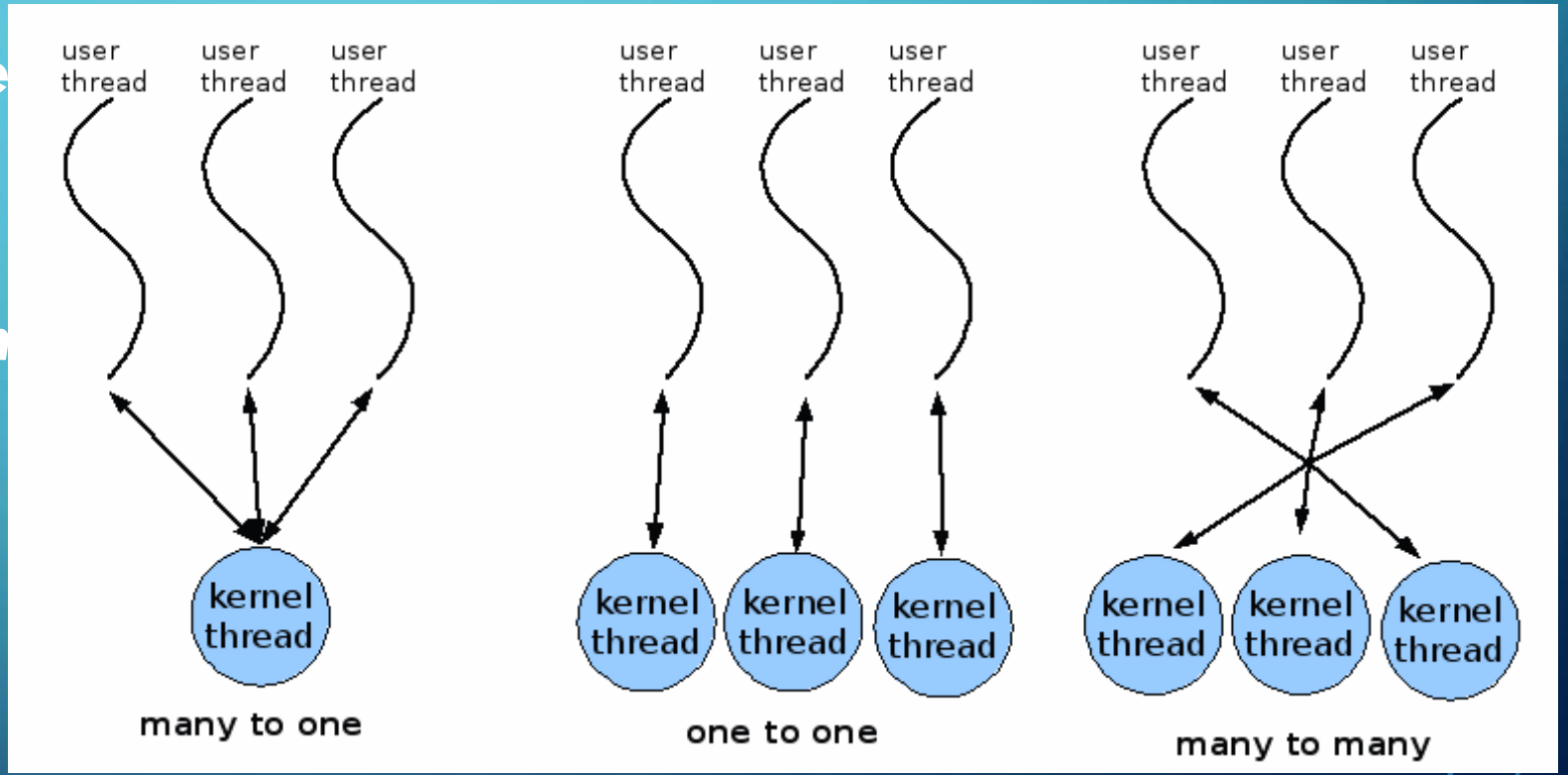
4. **Utilisasi arsitektur *multiprocessor*:** Keuntungan dari multithreading dapat sangat meningkat pada arsitektur *multiprocessor*, dimana setiap *thread* dapat berjalan secara pararel di atas processor yang berbeda. Pada arsitektur processor tunggal, CPU menjalankan setiap *thread* secara bergantian tetapi hal ini berlangsung sangat cepat sehingga menciptakan ilusi pararel, tetapi pada kenyataannya hanya satu *thread* yang dijalankan CPU pada satu-satuan waktu (satu-satuan waktu pada CPU biasa disebut *time slice* atau *quantum*).

MULTITHREADING

- ***Thread* pengguna:** *Thread* yang pengaturannya dilakukan oleh pustaka *thread* pada tingkatan pengguna. Karena pustaka yang menyediakan fasilitas untuk pembuatan dan penjadwalan *thread*, *thread* pengguna cepat dibuat dan dikendalikan.
- ***Thread* Kernel:** *Thread* yang didukung langsung oleh kernel. Pembuatan, penjadwalan dan manajemen *thread* dilakukan oleh kernel pada *kernel space*. Karena dilakukan oleh sistem operasi, proses pembuatannya akan lebih lambat jika dibandingkan dengan *thread* pengguna.

MODEL *MULTITHREADING*

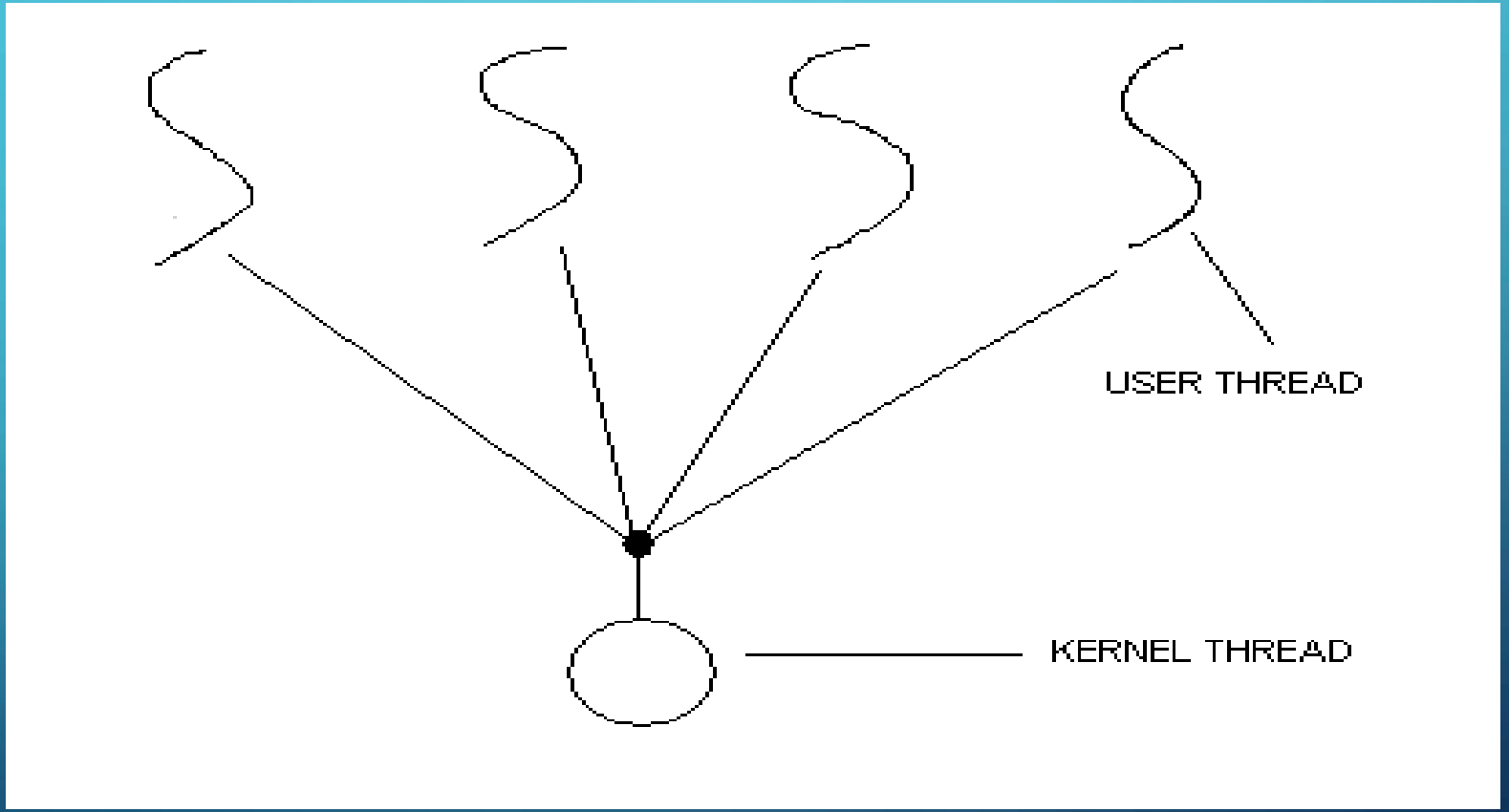
1. Model *Many-to-One*
2. Model *One-to-One.*
3. Model *Many-to-Many*



MULTITHREADING MODELS

1. **Many-to-One Model**
   - Memetakan banyak user-level *thread* ke satu kernel *thread*
   - Pengaturan *thread* dilakukan di *user space*
   - Efisien tetapi ia mempunyai kelemahan yang sama dengan user *thread*
   - Tidak dapat berjalan secara pararel pada *multiprocessor*

MULTITHREADING MODELS

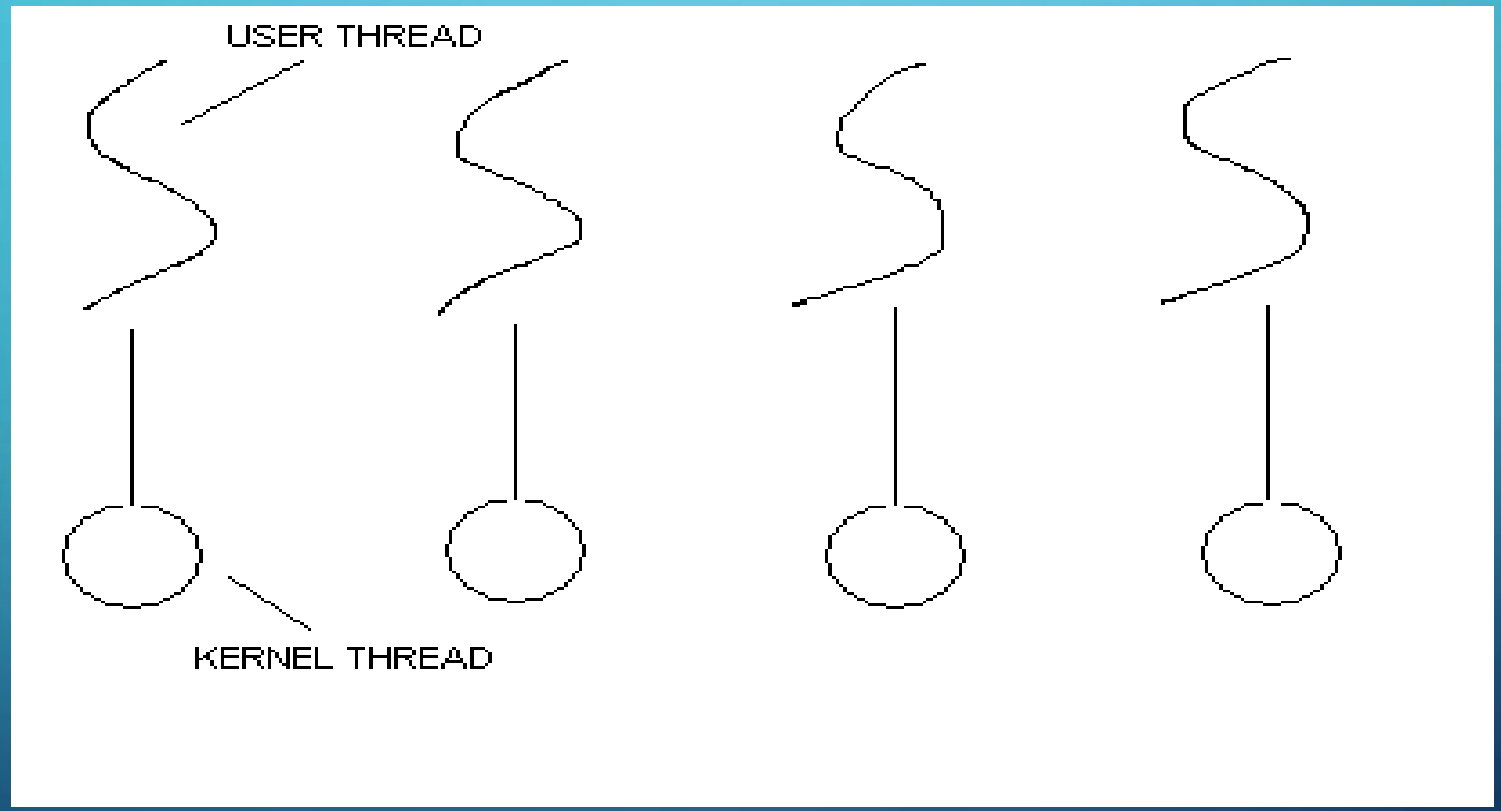


MULTITHREADING MODELS

2. *One-to-One Model*

- Memetakan setiap user *thread* ke kernel *thread*
- Menyediakan lebih banyak *concurrency* dibandingkan Many-to-One model
- Keuntungannya sama dengan keuntungan kernel *thread*
- Kelemahannya setiap pembuatan user *thread* membutuhkan pembuatan kernel *thread* yang dapat menurunkan performa dari sebuah aplikasi
- Sistem operasi yang mendukung One-to-One model adalah Windows NT dan OS/2

MULTITHREADING MODELS

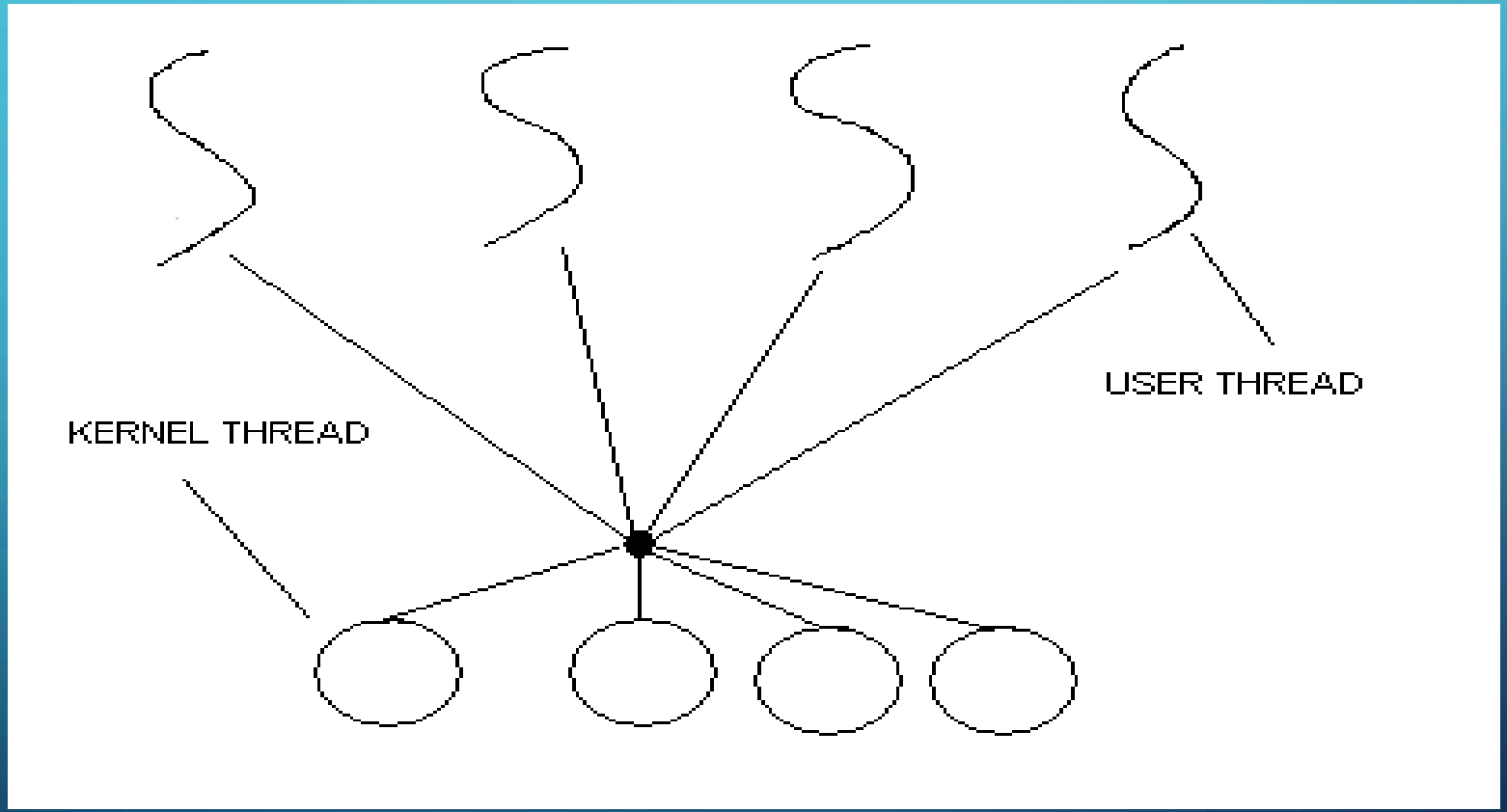


MULTITHREADING MODELS

3. Many-to-Many Model

- *multiplexes* banyak *user-level thread* ke kernel *thread* yang jumlahnya lebih kecil atau sama banyaknya dengan user-level *thread*
- Jumlah kernel *thread* dapat spesifik untuk sebagian aplikasi atau sebagian mesin
- Developer dapat membuat user *thread* sebanyak yang diperlukan, dan kernel *thread* yang bersangkutan dapat bejalan secara pararel pada *multiprocessor*.
- Ketika suatu *thread* menjalankan *blocking system call* maka kernel dapat menjadualkan *thread* lain untuk melakukan eksekusi.
- Sistem operasi yang mendukung model ini adalah Solaris, IRIX, dan Digital UNIX.

MULTITHREADING MODELS



PUSTAKA *THREAD*

- Pustaka *Thread* atau yang lebih familiar dikenal dengan *Thread Library* bertugas untuk menyediakan API (API (Application Programming Interface)) untuk *programmer* dalam menciptakan dan memanage *thread*. Ada dua cara dalam mengimplementasikan pustaka *thread* :

a. **Menyediakan API dalam level pengguna tanpa dukungan dari kernel sehingga pemanggilan fungsi tidak melalui *system call*. Jadi, jika kita memanggil fungsi yang sudah ada di pustaka, maka akan menghasilkan pemanggilan fungsi *call* yang sifatnya lokal dan bukan *system call*.**
b. **Menyediakan API di level kernel yang didukung secara langsung oleh sistem operasi. Pemanggilan fungsi *call* akan melibatkan *system call* ke kernel.**

- Ada tiga pustaka thread yang sering digunakan saat ini, yaitu: POSIX Pthreads, Java, dan Win32. Implementasi POSIX standard dapat dengan cara user level dan kernel level, sedangkan Win32 adalah kernel level. Java API thread dapat diimplementasikan oleh Pthreads atau Win32.

PEMBATALAN *THREAD* (*THREAD CANCELLATION*)

- *Thread Cancellation* ialah pembatalan *thread* sebelum tugasnya selesai. Umpamanya, jika dalam program Java hendak mematikan Java Virtual Machine (JVM). Sebelum JVM dimatikan, maka seluruh *thread* yang berjalan harus dibatalkan terlebih dahulu. Contoh lain adalah di masalah *search*. Apabila sebuah *thread* mencari sesuatu dalam *database* dan menemukan serta mengembalikan hasilnya, *thread* sisanya akan dibatalkan. *Thread* yang akan diberhentikan biasa disebut *target thread*.

PENJADWALAN *THREAD*

- Begitu dibuat, *thread* baru dapat dijalankan dengan berbagai macam penjadwalan. Kebijakan penjadwalanlah yang menentukan setiap proses, di mana proses tersebut akan ditaruh dalam daftar proses sesuai proritasnya dan bagaimana ia bergerak dalam daftar proses tersebut.

DEFINISI AGENT

Software Agent adalah entitas perangkat lunak yang didedikasikan untuk tujuan tertentu yang memungkinkan user untuk mendelegasikan tugasnya secara mandiri, selanjutnya software agent nantinya disebut agent saja. Agen bisa memiliki ide sendiri mengenai bagaimana menyelesaikan suatu pekerjaan tertentu atau agenda tersendiri. Agen yang tidak berpindah ke host lain disebut stationary agent.

DEFINISI AGENT (LANJUT)

Definisi agen yang lebih rinci, ditinjau dari sudut pandang sistem, adalah obyek perangkat lunak yang :

1. **Diletakan dalam lingkungan eksekusi**
2. **Memiliki sifat sebagai berikut :**
   a) **Reaktif, dapat merasakan perubahan dalam lingkungannya dan bertindak sesuai perubahan tersebut.**
   b) **Autonomous, mampu mengendalikan tindakannya sendiri**
   c) **Proaktif, mempunyai dorongan untuk mencapai tujuan**
   d) **Bekerja terus menerus sampai waktu tertentu**
3. **Dapat mempunyai sifat ortogonal sebagai berikut :**
   a. **Komunikatif, dapat berkomunikasi dengan agen yang lain.**
   b. **Mobile , dapat berpindah dari satu host ke host yang lain**
   c. **Learning, mampu menyesuaikan diri berdasarkan pengalaman sebelumnya**
   d. **Dapat dipercaya sehingga menimbulkan kepercayaan kepada end user.**

KARAKTERISTIK DARI AGENT

- **Autonomy** - Agent dapat melakukan tugas secara mandiri dan tidak dipengaruhi secara langsung oleh user, agent lain ataupun oleh lingkungan (environment).
- **Intelligence, Reasoning, dan Learning** - Setiap agent harus mempunyai standar minimum untuk bisa disebut agent, yaitu intelegensi (intelligence). Dalam konsep intelligence, ada tiga komponen yang harus dimiliki: internal knowledge base, kemampuan reasoning berdasar pada knowledge base yang dimiliki, dan kemampuan learning untuk beradaptasi dalam perubahan lingkungan.
- **Mobility dan Stationary** - Khusus untuk mobile agent, dia harus memiliki kemampuan yang merupakan karakteristik tertinggi yang dia miliki yaitu mobilitas. Berbeda dengan stationary agent. Tetapi keduanya tetap harus memiliki kemampuan untuk mengirim pesan dan berkomunikasi dengan agent lain.
- **Delegation** - Sesuai dengan namanya dan seperti yang sudah kita bahas pada bagian definisi, agent bergerak dalam kerangka menjalankan tugas yang diperintahkan oleh user. Fenomena pendelegasian (delegation) ini adalah karakteristik utama suatu program disebut agent.

KARAKTERISTIK DARI AGENT (LANJUT)

- **Reactivity** - Karakteristik agent yang lain adalah kemampuan untuk bisa cepat beradaptasi dengan adanya perubahan informasi yang ada dalam suatu lingkungan (enviornment). Lingkungan itu bisa mencakup: agent lain, user, informasi dari luar, dsb.
- **Proactivity dan Goal-Oriented** - Sifat proactivity boleh dibilang adalah kelanjutan dari sifat reactivity. Agent tidak hanya dituntut bisa beradaptasi terhadap perubahan lingkungan, tetapi juga harus mengambil inisiatif langkah penyelesaian apa yang harus diambil [Brenner et. al., 1998]. Untuk itu agent harus didesain memiliki tujuan (goal) yang jelas, dan selalu berorientasi kepada tujuan yang diembannya (goal-oriented).
- **Communication and Coordination Capability** - Agent harus memiliki kemampuan berkomunikasi dengan user dan juga agent lain. Masalah komunikasi dengan user adalah masuk ke masalah user interface dan perangkatnya, sedangkan masalah komunikasi, koordinasi, dan kolaborasi dengan agent lain adalah masalah sentral penelitian Multi Agent System (MAS).

KLASIFIKASI SOFTWARE AGENT

1. **Desktop Agent**

   Yaitu agent yang hidup dan bertugas dalam lingkungan Personal Computer (PC), dan berjalan diatas suatu Operating System (OS). Yang termasuk dalam klasifikasi ini adalah:

   - ✓ Operating System Agent
   - ✓ Application Agent
   - ✓ Application Suite Agent

KLASIFIKASI SOFTWARE AGENT (LANJ..)

2. **Internet Agent**

   Yaitu agent yang hidup dan bertugas dalam lingkungan jaringan Internet, melakukan tugasnya yaitu memanage informasi yang ada di Internet. Yang termasuk dalam klasifikasi ini adalah :

   - ✓ Web Search Agent
   - ✓ Web Server Agent
   - ✓ Information Filtering Agent
   - ✓ Information Retrieval Agent
   - ✓ Notification Agent
   - ✓ Service Agent
   - ✓ Mobile Agent

KARAKTERISTIK BAHASA PEMROGRAMAN

Bahasa pemrograman yang dipakai untuk tahap implementasi dari software agent, sangat menentukan keberhasilan dalam implementasi agent sesuai dengan yang diharapkan. Beberapa peneliti memberikan petunjuk tentang bagaimana karakteristik bahasa pemrorgaman yang sebaiknya di pakai.

- Object-Oriented
- Platform Independence
- Communication Capability
- Security
- Code Manipulation

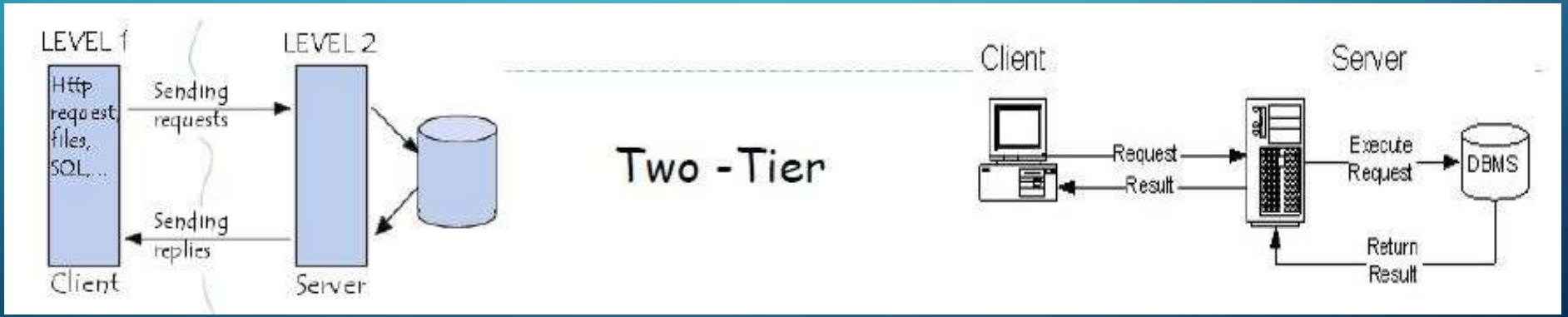
DEFINISI CLIENT SERVER

- Definisi client server menurut Budhi irawan (2005 : 30), Server adalah komputer database yang berada di pusat, dimana informasinya dapat digunakan bersama-sama oleh beberapa user yang menjalankan aplikasi di dalam komputer lokalnya yang disebut dengan Client.

MODEL CLIENT SERVER

- **Model Two Tier**

  Dalam model client/server, pemrosesan pada sebuah aplikasi terjadi pada client dan server.. Aplikasi ditempatkan pada computer client dan mesin database dijalankan pada server jarak-jauh. Aplikasi client mengeluarkan permintaan ke database yangmengirimkan kembali data ke client-nya. Model Two-tier terdiri dari tiga komponen yang disusun menjadi dua lapisan : client (yang meminta serice) dan server (yang menyediakan service).



MODEL CLIENT SERVER

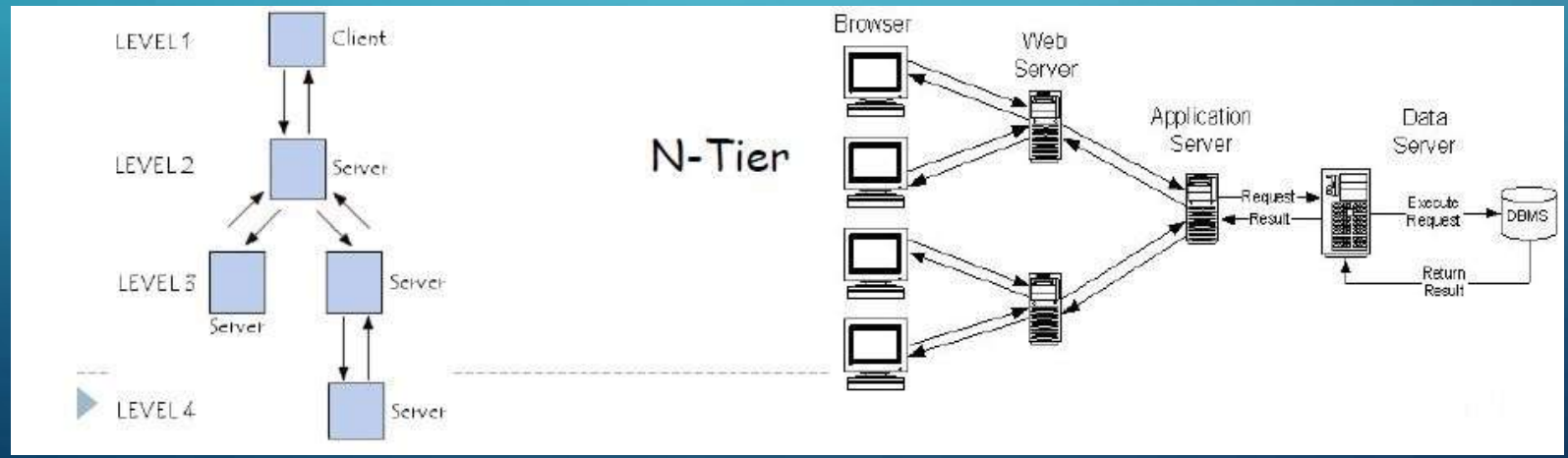
- **Model Three Tier**

  Pada arsitektur Three Tier ini terdapat Application Server yang berdiri di antara Client dan Database Server. Contoh dari Application server adalah IIS, WebSphere, dan sebagainya. Konsep model three-tier adalah model yang membagi fungsionalitas ke dalam lapisan-lapisan, aplikasi mendapatkan skalabilitas, keterbaharuan, dan keamanan.

MODEL CLIENT SERVER

- **Model Multi Tier**

  Arsitektur Multi Tier adalah suatu metode yang sangat mirip dengan Three Tier. Bedanya, pada Multi Tier akan diperjelas bagian UI (User Interface) dan Data Processing. Yang membedakan arsitektur ini adalah dengan adanya Business Logic Server. Database Server dan Bussines Logic Server merupakan bagian dari Data Processing, sedangkan Application Server dan Client/Terminal merupakan bagian dari UI.




TERIMA
KASIH
DAN
APA ADA
PERTANYAAN?