PROSES

Oleh : Rahmat Robi Waliyansyah, M.Kom.

Thread

- ***Thread*** adalah sebuah alur kontrol dari sebuah proses. Kontrol *thread* tunggal ini hanya memungkinkan proses untuk menjalankan satu tugas pada satu waktu.
- Banyak sistem operasi modern telah memiliki konsep yang dikembangkan agar memungkinkan sebuah proses untuk memiliki eksekusi *multi-threads*, agar dapat secara terus menerus mengetik dan menjalankan pemeriksaan ejaan didalam proses yang sama, maka sistem operasi tersebut memungkinkan proses untuk menjalankan lebih dari satu tugas pada satu waktu. Suatu proses yang *multithreaded* mengandung beberapa perbedaan alur kontrol dengan ruang alamat yang sama.
- *Thread* merupakan unit dasar dari penggunaan CPU, yang terdiri dari Thread_ID, *program counter*, *register set*, dan *stack*. Sebuah *thread* berbagi *code section*, *data section*, dan sumber daya sistem operasi dengan Thread lain yang dimiliki oleh proses yang sama. Thread juga sering disebut *lightweight process*. Sebuah proses tradisional atau *heavyweight process* mempunyai *thread* tunggal yang berfungsi sebagai pengendali.

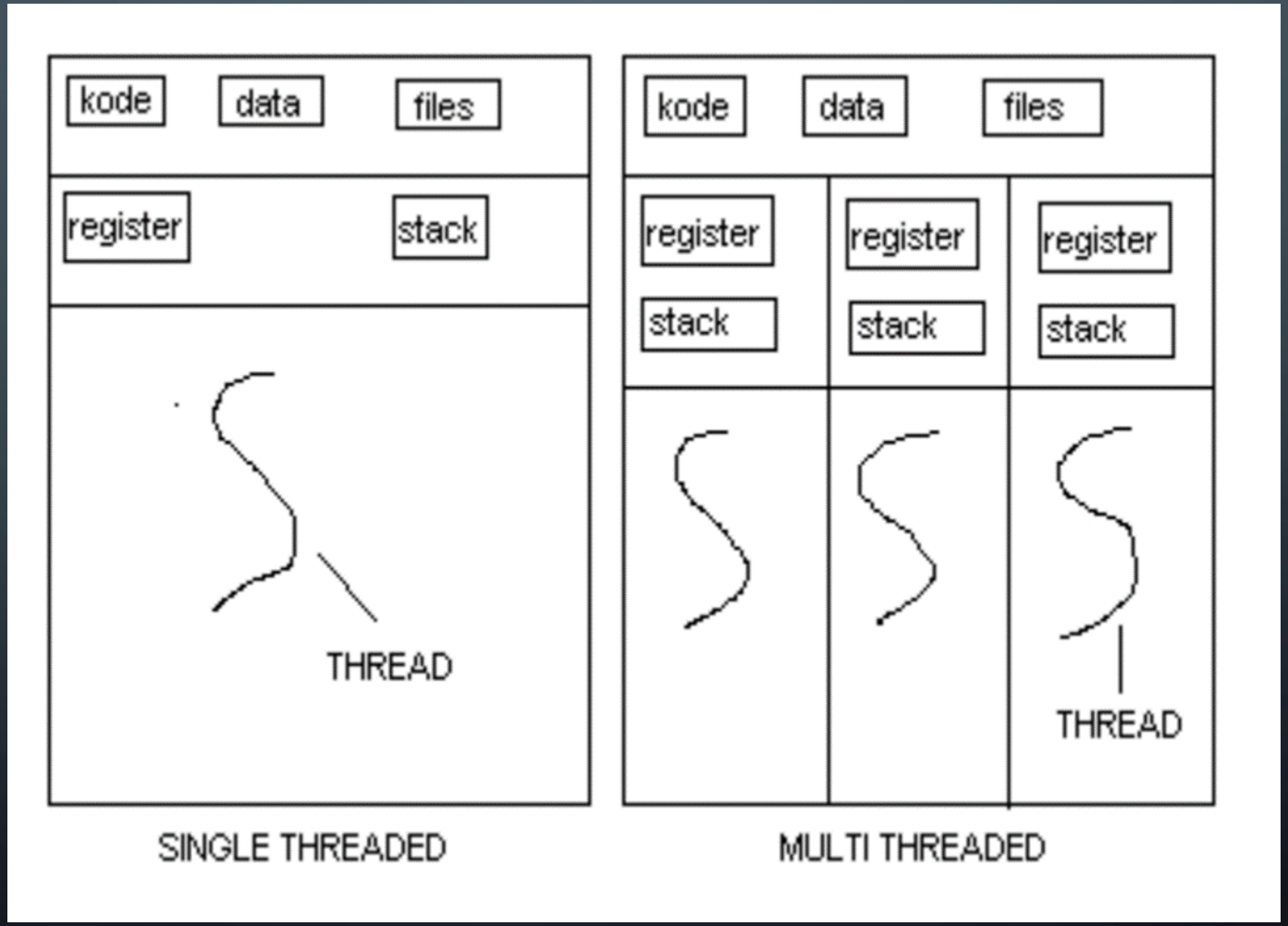
Perbedaan thread dengan proses

- Thread adalah bagian dari suatu proses
- Tiap proses mempunyai informasi status dan sumber daya sendiri, thread berbagi informasi status dan sumber daya dengan thread yang lain dalam satu proses
- Tiap proses mempunyai alamat yang berbeda, sedangkan thread berbagi alamat yang sama
- Pergantian antar threads lebih cepat daripada antar proses

CONTOH MULTITHREADING

- Web browser : sebuah *web browser* mempunyai *thread* untuk menampilkan gambar atau tulisan sedangkan *thread* yang lain berfungsi sebagai penerima data dari network.
- Web server : sebuah *web server* 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.

SINGLE & MULTITHREADED PROCESS



KEUTUNGAN THREAD

- Keuntungan dari program yang ***multithreading*** dapat dipisah menjadi empat kategori:

1. **Responsi**: Membuat aplikasi yang interaktif menjadi *multithreading* dapat membuat sebuah program terus berjalan meski pun 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. Keuntungan dari berbagi kode adalah mengizinkan sebuah aplikasi untuk mempunyai beberapa *thread* yang berbeda dalam lokasi memori yang sama

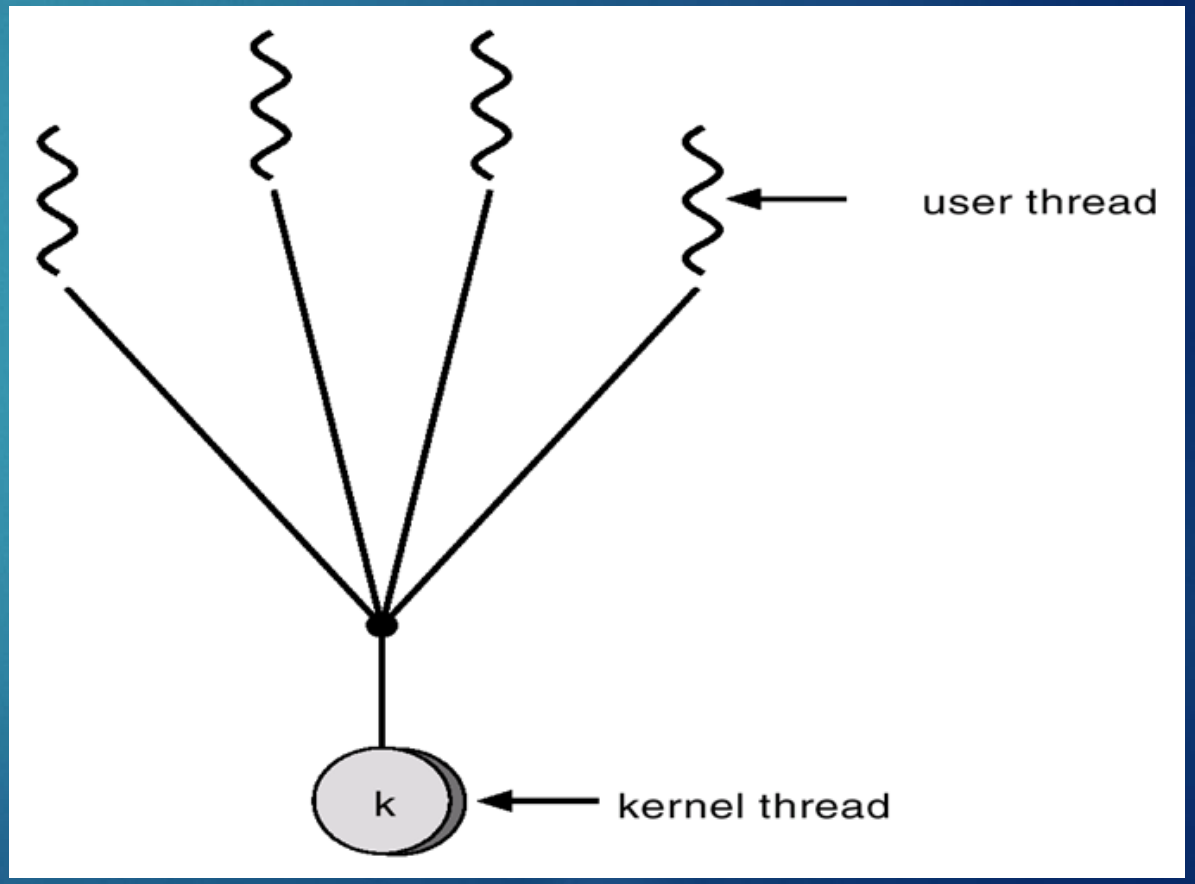
3. **Ekonomi**: dalam pembuatan sebuah proses banyak dibutuhkan pengalokasian memori dan sumber daya. Alternatifnya adalah dengan penggunaan *thread*, karena *thread* berbagi memori dan sumber daya proses yang memilikinya maka akan lebih ekonomis untuk membuat dan *context switch thread*. Akan susah 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*.
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*).

Macam-macam thread

- ***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.
- ***Kernel*** adalah sebuah perangkat lunak yang membuat komunikasi / mediator antara aplikasi komputer dan perangkat keras, yang menyediakan pelayanan sistem seperti pengaturan memori untuk proses-proses yang sedang berjalan, pengaturan file-file, input-output terhadap dan dari suatu device dan masih banyak lagi fungsi tambahan yang lainnya. Intinya adalah kernel merupakan suatu penghubung (antara software dan hardware).

Model-model *Multithreading*:

- **Model *Many-to-One*.** Model ini memetakan beberapa *thread* tingkatan pengguna ke sebuah *thread*. tingkatan kernel. Pengaturan *thread* dilakukan dalam ruang pengguna sehingga efisien. Hanya satu *thread* pengguna yang dapat mengakses *thread* kernel pada satu saat. Jadi *Multiple thread* tidak dapat berjalan secara paralel pada multiprosesor.

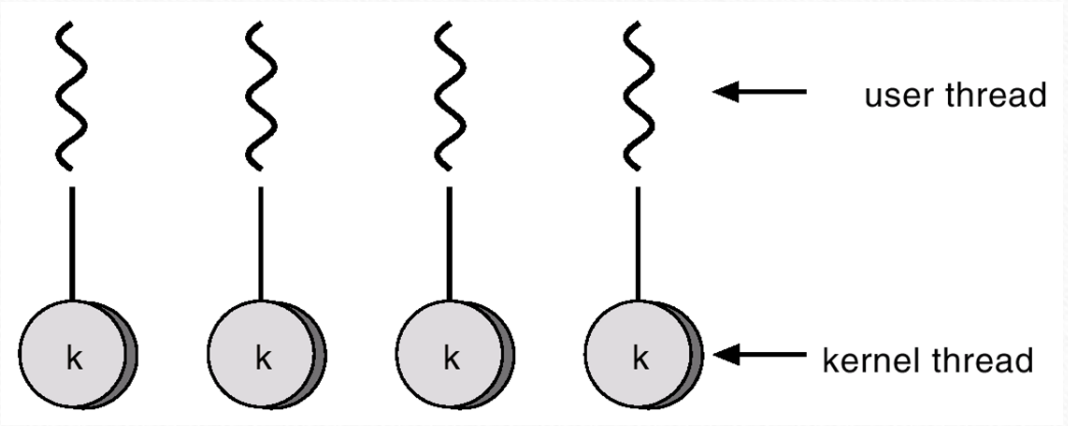


Dead Lock

- Suatu kondisi dimana 2 proses/lebih tidak dapat meneruskan eksekusinya.
- **Penyebab utama terjadinya deadlock** adalah terbatasnya sumber daya yang akan digunakan oleh proses-proses. Tiap proses berkompetisi untuk memperebuntukan sumber daya yang ada. Jadi deadlock berhubungan erat dengan tersedianya sumber daya dari komputer

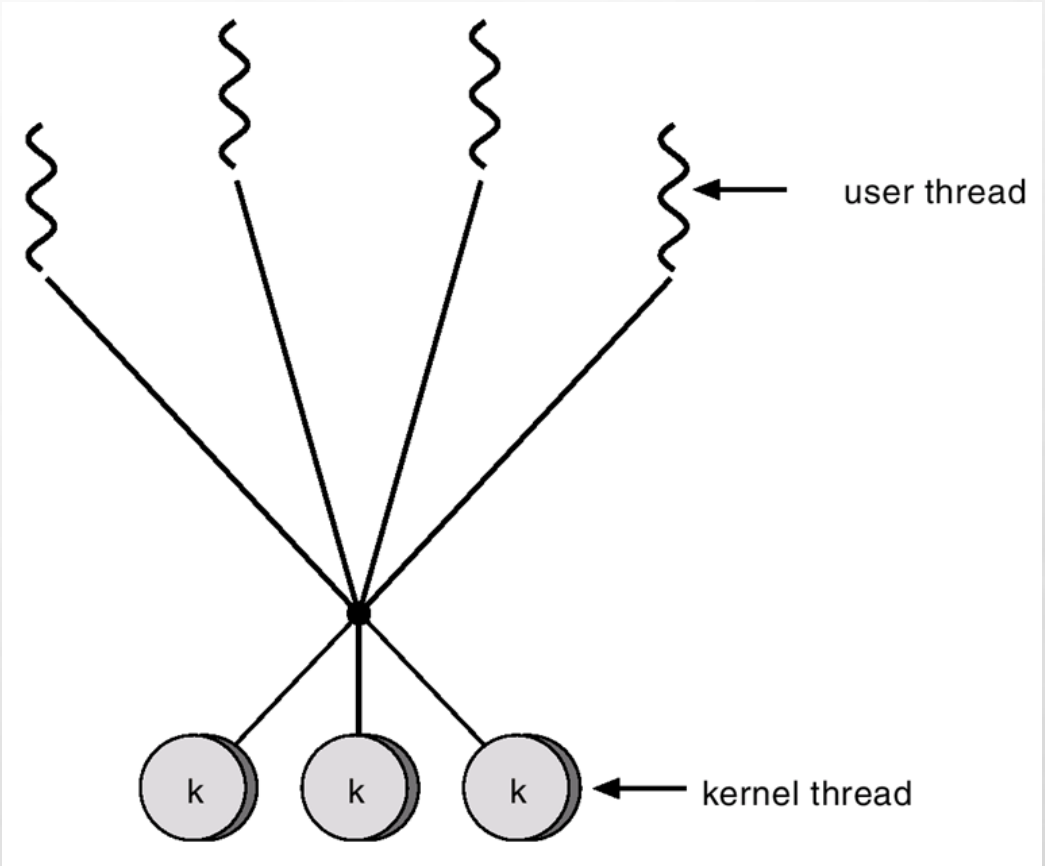
Model *One-to-One*

- **Model *One-to-One*.** Model ini memetakan setiap *thread* tingkatan pengguna ke setiap *thread*. Ia menyediakan lebih banyak *concurrency* dibandingkan model *Many-to-One*. Keuntungannya sama dengan keuntungan *thread* kernel. Kelemahan model ini ialah setiap pembuatan *thread* pengguna memerlukan tambahan *thread* kernel.



MODEL *MANY-TO-MANY*

- **MODEL *MANY-TO-MANY***. MODEL INI MEMULTIPLEKS BANYAK *THREAD* TINGKATAN PENGGUNA KE *THREAD* KERNEL YANG JUMLAHNYA SEDIKIT ATAU SAMA DENGAN TINGKATAN PENGGUNA. MODEL INI MENGIZINKAN *DEVELOPER* MEMBUAT *THREAD* SEBANYAK YANG IA MAU TETAPI *CONCURRENCY* TIDAK DAPAT DIPEROLEH KARENA HANYA SATU *THREAD* YANG DAPAT DIJADWALKAN OLEH KERNEL PADA SUATU WAKTU. KEUNTUNGAN DARI SISTEM INI IALAH KERNEL *THREAD* YANG BERSANGKUTAN DAPAT BERJALAN SECARA PARALEL PADA MULTIPROSESSOR.



Thread cancelation

- ***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*.
- Pemberhentian *target Thread* dapat dilakukan dengan 2 cara:

  1. ***Asynchronous cancellation.*** Suatu *thread* seketika itu juga membatalkan *target thread*.
  2. ***Deferred cancellation.*** Suatu *thread* secara periodik memeriksa apakah ia harus batal, cara ini memperbolehkan *target thread* untuk membatalkan dirinya secara terurut.

Client - Server

- **Server** adalah sebuah sistem komputer yang menyediakan jenis layanan tertentu dalam sebuah jaringan komputer. Server didukung dengan prosesor yang bersifat *scalable* dan RAM yang besar, juga dilengkapi dengan sistem operasi khusus, yang disebut sebagai sistem operasi jaringan atau ***network operating system*.** Server juga menjalankan perangkat lunak administratif yang mengontrol akses terhadap jaringan dan sumber daya yang terdapat di dalamnya, seperti halnya berkas atau alat pencetak (*printer*), dan memberikan akses kepada *workstation*anggota jaringan.
- **Client** adalah komputer yang diperbolehkan untuk masuk kedalam network dan mengambil/menggunakan segala sumber daya yang tersedia di dalam network.
- kata kuncinya adalah pada sistem client/server harus terdapat **satu atau beberapa server yang menyediakan layanan** dan **satu atau beberapa klien yang meminta layanan** tersebut.

Gambar Client - Server

MODEL CLIENT - SERVER

- Ada beberapa model client/server yang penting untuk diketahui. Dimulai dari arsitektur mainframe hingga arsitektur client/server.

1. **Arsitektur Mainframe**

   Pada arsitektur ini, terdapat sebuah komputer pusat (host) yang memiliki sumber daya yang sangat besar, baik memori, processor maupun media penyimpanan. Melalui komputer terminal, pengguna mengakses sumber daya tersebut. Komputer terminal hanya memiliki monitor/keyboard dan tidak memiliki CPU.

2. **Arsitektur File Sharing**

   Pada arsitektur ini komputer server menyediakan file-file yang tersimpan di media penyimpanan server yang dapat diakses oleh pengguna. Arsitektur file sharing memiliki keterbatasan, terutama jika jumlah pengakses semakin banyak serta ukuran file yang di shaing sangat besar. Hal ini dapat mengakibatkan transfer data menjadi lambat.

3. Arsitektur Client/Server

Karena keterbatasan sistem file sharing, dikembangkanlah arsitektur client/server. Salah satu hasilnya yaitu berupa software database server yang menggantikan software database berbasis file server. Dikenalkan pula RDBMS (Relational Database Management System). Dengan arsitektur ini, query data ke server dapat terlayani dengan lebih cepat karena yang ditransfer bukanlah file, tetapi hanyalah hasil dari query tersebut. RPC (Remote Procedure Calls) memegang peranan penting pada arsitektur client/server.

4. Model Two-tier

Model Two-tier terdiri dari tiga komponen yang disusun menjadi dua lapisan: *client* (yang meminta *service*) dan *server* (yang menyediakan *service*). Tiga komponen tersebut yaitu :

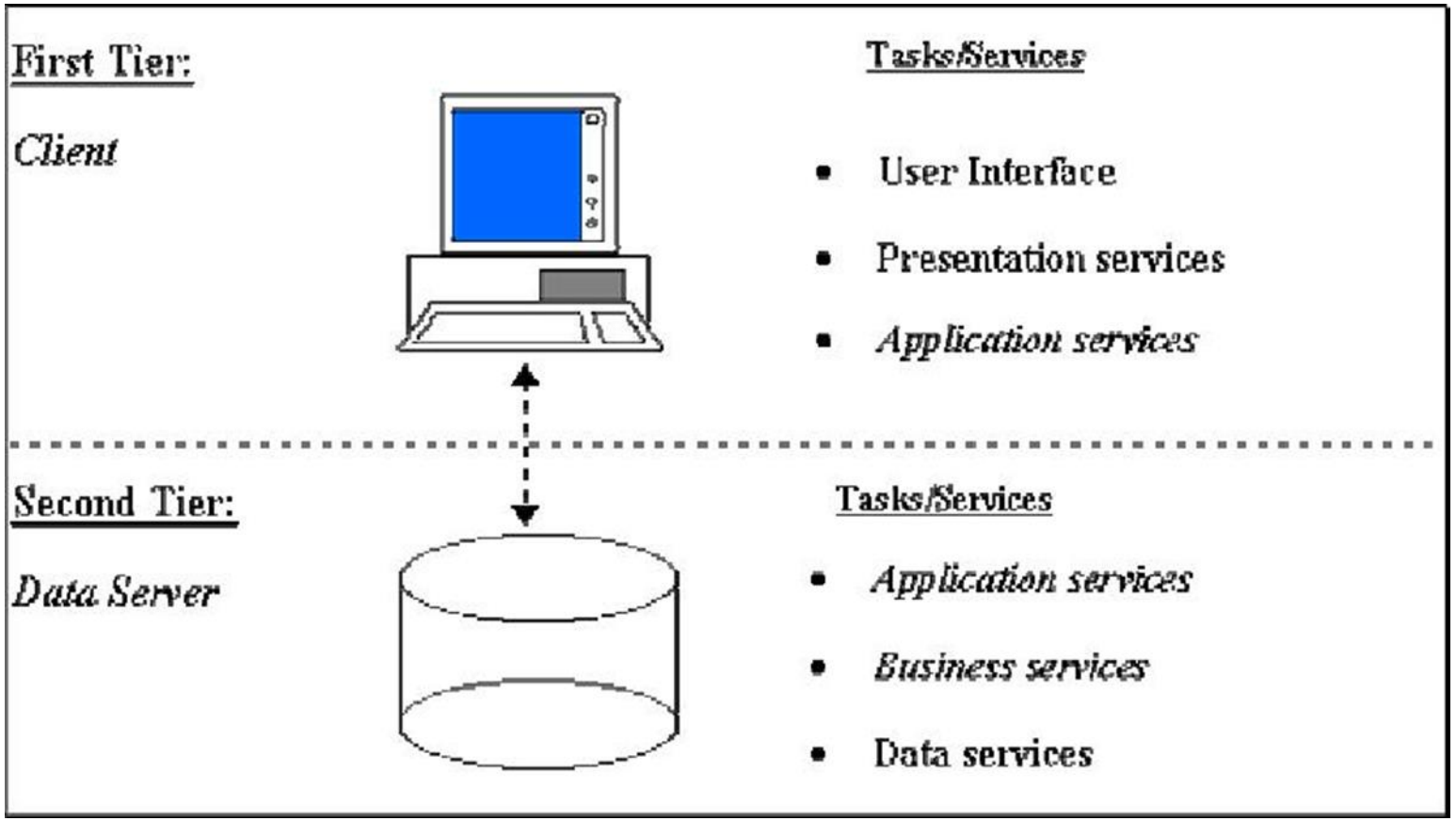
User Interface, yaitu antar muka program aplikasi yang berhadapan dan digunakan langsung oleh user.

Manajemen proses

Database

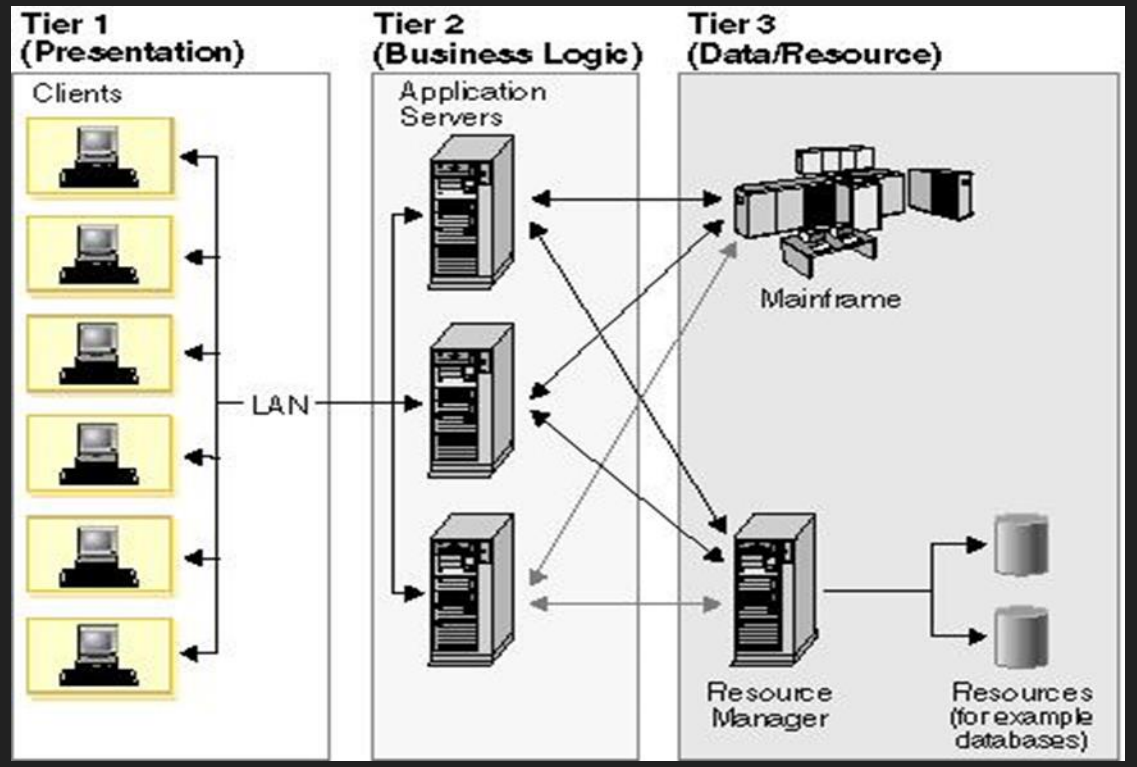
Model ini memisahkan peranan user interface dan database dengan jelas, sehingga terbentuk dua lapisan.

Gambar model two-tier



5. Model Three-tier

Pada model ini disisipkan satu layer tambahan diantara user interface tier dan database tier. Tier tersebut dinamakan middle-tier. Middle-Tier terdiri dari bussiness logic dan rules yang menjembatani query user dan database, sehingga program aplikasi tidak bisa mengquery langsung ke database server, tetapi harus memanggil prosedur-prosedur yang telah dibuat dan disimpan pada middle-tier. Dengan adanya server middle-tier ini, beban database server berkurang. Jika query semakin banyak dan/atau jumlah pengguna bertambah, maka server-server ini dapat ditambah, tanpa merubah struktur yang sudah ada. Ada berbagai macam software yang dapat digunakan sebagai server middle-tier. Contohnya MTS (Microsoft Transaction Server) dan MIDAS.



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.

- **Karakteristik dari Agen:**

1. ***Autonomy***: Agent dapat melakukan tugas secara mandiri dan tidak dipengaruhi secara langsung oleh user, agent lain ataupun oleh lingkungan (environment). Untuk mencapai tujuan dalam melakukan tugasnya secara mandiri, agent harus memiliki kemampuan kontrol terhadap setiap aksi yang mereka perbuat, baik aksi keluar maupun ke dalam
2. ***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.

3. ***Delegation***: Agent bergerak dalam kerangka menjalankan tugas yang diperintahkan oleh user. Fenomena pendelegasian (*delegation*) ini adalah **karakteristik utama** suatu program disebut agent.
4. ***Reactivity***: Kemampuan untuk bisa cepat beradaptasi dengan adanya perubahan informasi yang ada dalam suatu lingkungan. Lingkungan itu bisa mencakup: agent lain, user, informasi dari luar, dsb.
5. ***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. Untuk itu agent harus didesain memiliki tujuan (*goal*) yang jelas, dan selalu berorientasi kepada tujuan yang diembannya (*goal-oriented*).
6. ***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 agent software

1. Klasifikasi menurut Karakteristik yang Dimiliki
   a. ***Collaborative Agent***: *Agent* yang memiliki kemampuan melakukan kolaborasi dan koordinasi antar agent dalam kerangka *Multi Agent System* (MAS).
   b. ***Interface Agent***: *Agent* yang memiliki kemampuan untuk berkolaborasi dengan user, melakukan fungsi *monitoring* dan *learning* untuk memenuhi kebutuhan user.
   c. ***Mobile Agent***: Agent yang memiliki kemampuan untuk bergerak dari suatu tempat ke tempat lain, dan secara mandiri melakukan tugas ditempat barunya tersebut, dalam lingkungan jaringan komputer.
   d. ***Information dan Internet Agent***: *Agent* yang memiliki kemampuan untuk menjelajah internet untuk melakukan pencarian, pemfilteran, dan penyajian informasi untuk user, secara mandiri. Atau dengan kata lain, memanage informasi yang ada di dalam jaringan Internet

e. ***Reactive Agent***: *Agent* yang memiliki kemampuan untuk bisa cepat beradaptasi dengan lingkungan baru dimana dia berada.

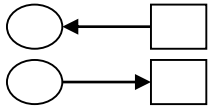
f. ***Hybrid Agent***: Kita sudah mempunyai lima klasifikasi *agent*. Kemudian *agent* yang memiliki katakteristik yang merupakan gabungan dari karakteristik yang sudah kita sebutkan sebelumnya adalah masuk ke dalam *hybrid agent*.

g. ***Heterogeneous Agent System***: Dalam lingkungan *Multi Agent System* (MAS), apabila terdapat dua atau lebih *hybrid agent* yang memiliki perbedaan kemampuan dan karakteristik, maka sistem MAS tersebut kita sebut dengan *heterogeneous agent system*.

2. Klasifikasi menurut Lingkungan Dimana Dijalankan

a. ***Desktop Agent***: *Agent* yang hidup dan bertugas dalam lingkungan *Personal Computer* (PC), dan berjalan diatas suatu *Operating System* (OS). Termasuk dalam klasifikasi ini adalah:

- *Operating System Agent*
- *Application Agent*
- *Application Suite Agent*



b. ***Internet Agent***: *Agent* yang hidup dan bertugas dalam lingkungan jaringan Internet, melakukan tugas memanage informasi yang ada di Internet. Termasuk dalam klasifikasi ini adalah:

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

Bahasa pemrograman yang digunakan

- Bahasa pemrograman yang dipakai untuk tahap implementasi dari software agent, sangat menentukan keberhasilan dalam implementasi agent sesuai dengan yang diharapkan, diantaranya yaitu :

1. ***Object-Orientedness:***

   Karena agent adalah berhubungan dengan obyek, bahkan beberapa peneliti menganggap agent adalah obyek yang aktif, maka juga agent harus diimplementasikan kedalam pemrograman yang berorientasi obyek (object-oriented programming language).

2. ***Platform Independence:***

   Seperti sudah dibahas pada bagian sebelumnya, bahwa agent hidup dan berjalan diberbagai lingkungan. Sehingga idealnya bahasa pemrograman yang dipakai untuk implementasi adalah yang terlepas dari platform, atau dengan kata lain program tersebut harus bisa dijalankan di platform apapun (platform independence).

3. ***Communication Capability:***

   Pada saat berinteraksi dengan agent lain dalam suatu lingkungan jaringan (network environment), diperlukan kemampuan untuk melakukan komunikasi secara fisik. Sehingga diperlukan bahasa pemrograman yang dapat mensupport pemrograman yang berbasis network dan komunikasi.

4.Security:

Faktor keamanan (security) adalah factor yang sangat penting dalam memilih bahasa pemrorgaman untuk implementasi software agent. Terutama untuk mobil agent, diperlukan bahasa pemrograman yang mensupport level-level keamanan yang bisa membuat agent bergerak dengan aman.

5.Code Manipulation:

Beberapa aplikasi software agent memerlukan manipulasi kode program secara runtime, sehingga diperlikan bahasa pemrograman untuk software agent yang dapat menangani masalah runtime tersebut.

- Dari karakteristik di atas dapat disimpulkan bahwa bahasa pemrograman yang layak untuk mengimplementasikan software agent adalah sebagai berikut :
  - ✓ Java
  - ✓ Telescript
  - ✓ Tcl/Tk, Safe-Tcl, Agent-Tcl