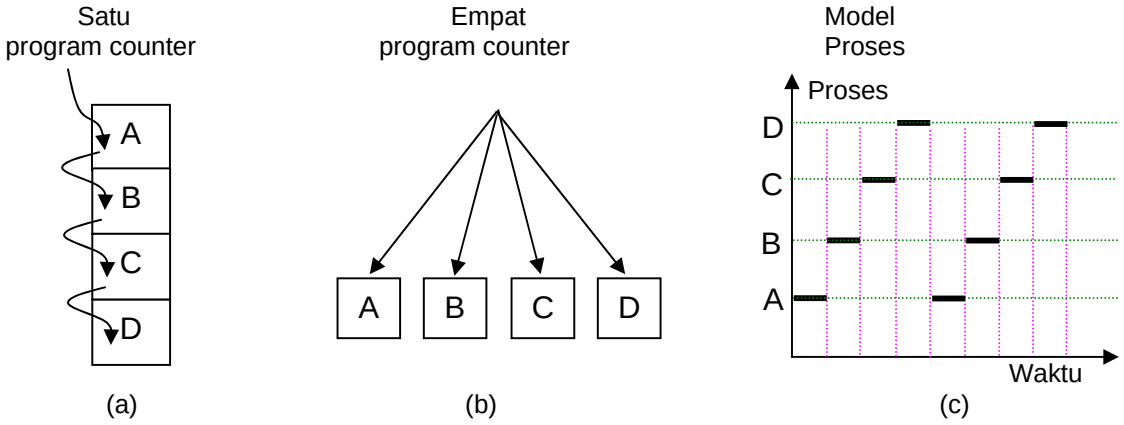
MANAJEMEN PROSES

Proses :
Adalah program yang sedang di jalankan atau software yang sedang dilaksanakan termasuk sistem operasi yang disusun menjadi sejumlah proses sequential.

Konsep dasar :
1. Multiprogramming
   Melakukan proses satu persatu secara bergantian dalam waktu yang sangat cepat / bersamaan (hardware level). Setiap proses mempunyai satu virtual CPU.
2. Pseudoparallelism
   Melakukan lebih dari satu pekerjaan dalam waktu yang bersamaan / pseudoparallelism (user level).

Model Proses :
1. Sequential Process / bergantian
2. Multiprogramming
3. CPU Switching → peralihan prosedur dalam mengolah 1 proses ke proses lainnya.

Secara konsep setiap proses mempunyai 1 virtual CPU, tetapi pada kenyataannya adalah multiprogramming. Maka akan lebih mudah menganggap kumpulan proses yang berjalan secara parallel.



Keterangan :
a. multiprogramming untuk 4 program di memori
b. model konseptual untuk 4 proses independent, sequential
c. hanya 1 program yang aktif dalam 1 waktu = pseudoparalel

Hirarki Proses
Pemanggilan proses oleh proses lain disebut parallel. Sistem operasi menyediakan apa yang dibutuhkan oleh proses. Umumnya proses diciptakan dan dihilangkan selama operasi berlangsung.

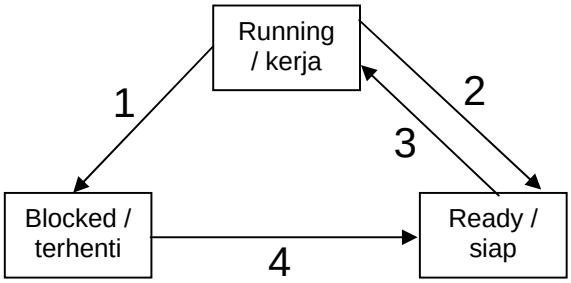
1. *Create & Destroy Proses*
   Sistem operasi yang mendukung konsep proses, harus menyediakan suatu cara untuk membuat (*create*) proses dan menghilangkan (*destroy*) proses.
2. *Fork System Call*
   Mekanisme untuk membuat (*create*) proses yang identik dengan proses yang memanggilnya.
   Contoh :
   Parent → children
   → children → children
   → children

Pada UNIX, parent dan child process running secara parallel.
Pada DOS, parent dan child process running secara bergantian (sequential). Contohnya : MSDOS sebagai parent dan program aplikasi sebagai child.

Process scheduler : untuk pengaturan eksekusi proses

3 Status proses / bagian keadaan proses :
1. Running / kerja, benar-benar menggunakan CPU pada saat itu (sedang mengeksekusi instruksi proses itu).
2. Blocked / terhenti, tidak dapat berjalan sampai kegiatan eksternal terlaksana (proses menunggu kejadian untuk melengkapi tugasnya)
   Bisa berupa proses menunggu : Selesainya operasi perangkat I/O; Tersedianya memori; Tibanya pesan jawaban
3. Ready / siap, proses siap dikerjakan tetapi menunggu giliran dengan proses lain yang sedang dikerjakan (bisa berjalan, sementara berhenti untuk memungkinkan proses lain dikerjakan)

Transisi Status



1. Proses di blok untuk melayani input karena sumber daya yang diminta belum tersedia / meminta layanan I/O sehingga menunggu kejadian muncul.
2. Penjadwalan mengambil proses lain.
3. Penjadwalan mengambil proses ini (baru).
4. Input telah tersedia.

Implementasi Proses :
- Untuk mengimplementasikan model proses, sistem operasi menggunakan suatu tabel / array yang disebut tabel proses dengan 1 entry per-proses.
- Setiap entry berisi tentang : status proses, program counter, stack pointer, alokasi memori, status file, informasi schedulling / penjadwalan informasi, dll dari status kerja ke status siap.

Contoh Tabel Proses :

Proses management	Memory management	File management
Register	Pointer to text segment	UMASK mask
Program counter	Pointer to data segment	Root directoy
Program status word	Pointer to bss segment	Working directory
Stack pointer	Exit status	File descriptiors
Process state	Signal status	Effective uid
Time when process started	Process id	Effective gid
CPU time used	Parent process	System call parameters
Children's CPU time	Process group	Various flag bits
Time of next alarm	Real uid	
Message queue pointers	Effective uid	
Pending signal bits	Real gid	
Process id	Effective gid	
Various flag bits	Bit maps for signals	
	Various flag bits	

- **Interupsi :** Kerja prosesor pada suatu proses terhenti oleh pensaklaran konteks.
- **Pensaklaran konteks :** perubahan kegiatan prosesor dari proses ke proses yang terjadi diantara proses sistem / proses aplikasi
- **Konteks :** kegiatan prosesor terhadap sesuatu hal, berasal dari sistem operasi, sistem bahasa dan sistem utilitas.
- **Blok kendali proses :** suatu bagian memori untuk mencatat keadaan proses, yang terbagi atas wilayah dimana setiap wilayah untuk mencatat informasi yang berbeda.

2 cara interupsi pada processor :

1. *Interupsi langsung*
   Berasal dari luar prosesor (peripheral / alat mengirim sinyal kepada prosesor untuk meminta pelayanan)
2. *Interupsi Tanya / Polling*
   Berasal dari prosesor (prosesor secara bergiliran mengecek apakah ada peripheral yang memerlukan pelayanan atau tidak)

- Interupsi dapat di-enable dan disable tergantung pada levelnya.
- Pembangkit interupsi dapat berasal dari :
  - *Program*, di dalam program telah dirancang pada bagian tertentu akan terjadi pensaklaran konteks, yang menimbulkan interupsi, contohnya pada saat penggunaan alat / prosesor secara bergantian.
  - *Prosesor*, prosesor sendiri dapat membangkitkan interupsi, yang biasa mengolah logika dan aritmatika. Jika melampoi ukuran tampung register di dalam prosesor, maka terjadi kekeliruan yang akan menginterupsi kerjanya sendiri dan menyerahkan kendali prosesor pada sistem operasi. Misalnya pembagian dengan bilangan nol.
  - *Satuan kendali*, tugas untuk melaksanakan interupsi terletak pada satuan kendali, sehingga satuan kendali dapat membangkitkan interupsi. Misalnya kekeliruan instruksi
  - *Kunci waktu / clock*, menggunakan interupsi berkala. Misalnya pada program looping yang tak terhingga, diinterupsi pada setiap selang waktu 60 detik.
  - *Peripheral I/O*, I/O jika akan bekerja memberitahukan pada prosesor dengan interupsi prosesor dan juga ketika pekerjaan selesai atau pada saat terjadi kekeliruan paritas.
  - *Memori*, karena terjadi kekeliruan, misalnya ketika prosesor ingin mencapai alamat memori yang terletak di luar bentangan alamat memori yang ada.
  - *Sumber daya lain*, misal dibangkitkan oleh operator sistem komputer yang mengerti cara interupsi.

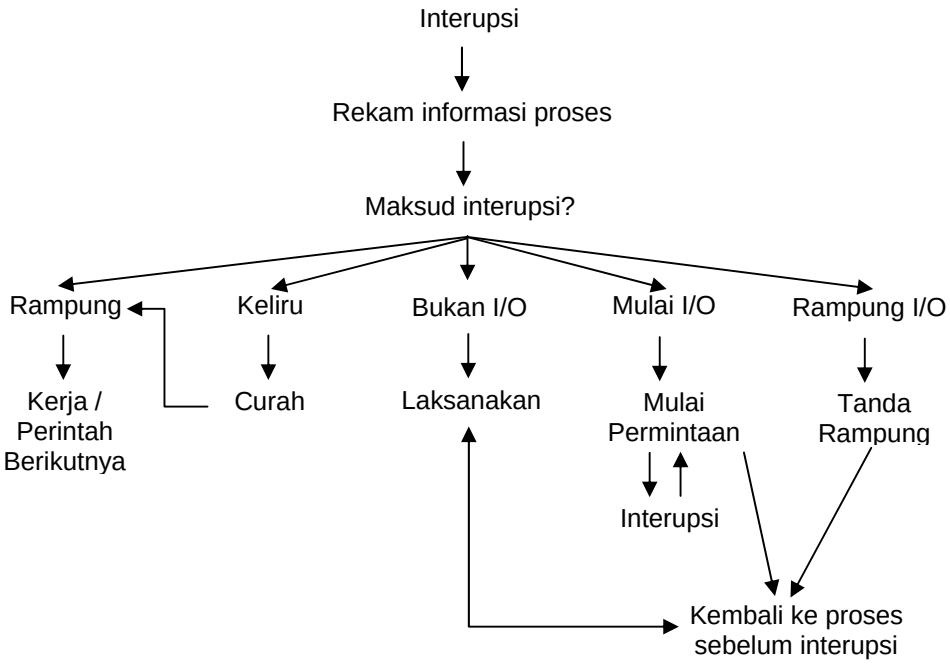
- **Interupsi vector** : Berisi alamat prosedur service interupsi
- **Penerimaan interupsi dan interupsi berganda** : ada kalanya interupsi ditolak oleh prosesor atau interupsi yang datang tidak hanya satu sehingga diperlukan prioritas.

Tindak lanjut interupsi :

1. *Penata interupsi / interrupt handler*
   jika terjadi interupsi, maka kendali prosesor diserahkan ke bagian penata interupsi pada sistem operasi, maka penata interupsi inilah yang melaksanakan interupsi.
   a. Instruksi yang sedang diolah oleh prosesor dibiarkan sampai selesai program.
   b. Penata interupsi merekam semua informasi proses ke dalam blok kendali proses.
   c. Penata interupsi mengidentifikasi jenis dan asal interupsi.
   d. Penata interupsi mengambil tindakan sesuai dengan yang dimaksud interupsi.
   e. Penata interupsi mempersiapkan segala sesuatu untuk pelanjutan proses yang diinterupsi.
2. *Penata keliru / error handler*
   yaitu interupsi karena kekeliruan pada pengolahan proses dan bagian pada sistem operasi yang menata kegiatan akibat kekeliruan adalah penata keliru.
   a. *Pemulihan*, komputer telah dilengkapi dengan sandi penemuan dan pemulihan kekeliruan, contohnya telah dilengkapi dengan sandi Hamming sehingga ketika menemukan kekeliruan sandi akan mengoreksi kekeliruan itu, proses pulih ke bentuk semula sebelum terjadi kekeliruan.

b. *Pengulangan*, mengatur agar proses yang membangkitkan interupsi keliru dikerjakan ulang, jika kekeliruan dapat diatasi maka proses akan berlangsung seperti biasa, jika tidak teratasi maka interupsi akan menempuh tindak lanjut keluar dari proses.
c. *Keluar dari proses*, penata keliru menyiapkan tampilan berita keliru dari monitor, setelah itu prosesor keluar dari proses, ini adalah tindakan terakhir jika tidak dapat menolong proses yang keliru tersebut.

Tindak lanjut interupsi menurut Peterson & Silberschatz :



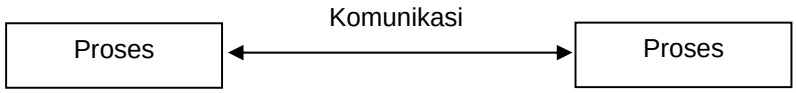
Keterangan :
1. Rampung
   Program selesai dilaksanakan oleh prosesor sehingga penyerahan kendali proses ke pekerjaan baru / perintah baru karena proses yang dikerjakan oleh prosesor telah selesai.
2. Keliru
   Jika menemukan kekeliruan akan ditampilkan, kemudian kendali prosesor diserahkan pada perintah berikutnya.
3. Permintaan bukan dari alat I/O
   Setelah interupsi selesai dilayani, kendali prosesor dikembalikan pada proses semula, misalnya interupsi berkala.
4. Permintaan dari alat I/O
   Setelah interupsi selesai dilayani, kendali prosesor dikembalikan pada proses sebelumnya, tetapi ada kalanya prosesor ikut campur dalam kerja alat I/O sehingga pengembalian kendali proses semula, tidak berlangsung segera / tunggu.
5. Rampung dari alat I/O
   Setelah interupsi selesai dilayani dan tanda rampung dicatat, kendali prosesor dikembalikan ke proses semula. Biasanya untuk alat I/O yang tidak diikut campuri oleh prosesor.

Catt:
1 & 2 tidak mengembalikan prosesor ke proses yang terinterupsi, sedangkan
3,4,5 mengembalikan prosesor ke proses yang terinterupsi.

Langkah-langkah yang dilakukan sistem operasi pada saat terjadi interupsi :
1. hardware memasukkan program counter, dl.l.
   memasukkan ke dalam stack pencacah program

2. Hardware memuatkan (load) program counter baru dari vector interrupsi
3. Prosedur bahasa rakitan menyimpan isi register
4. Prosedur bahasa rakitan men-set stack yang baru
5. Prosedur C menandai proses servis siap (ready)
6. Scheduler / penjadwalan menentukan proses mana yang akan jalan berikutnya
7. Prosedur C kembali ke modus bahasa rakitan
8. Prosedur bahasa rakitan memulai proses yang sedang dilaksanakan.

Komunikasi antar proses
(*Inter Process Communinication / IPC*) :

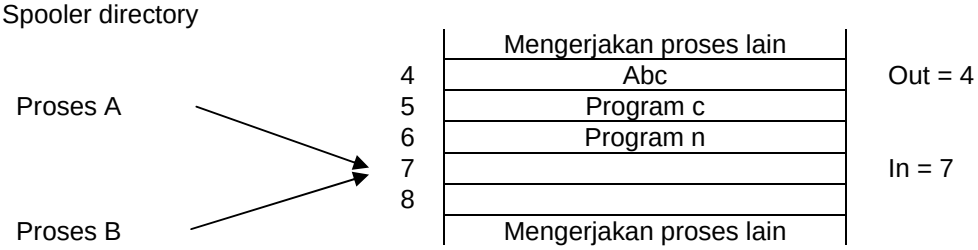


- Beberapa proses biasanya berkomunikasi dengan proses lainnya.
- Contohnya pada shell pipe line : output dari proses pertama harus diberikan kepada proses ke dua dan seterusnya.
- Pada beberapa sistem operasi, proses-proses yang bekerja bersama sering *sharing* (berbagi) media penyimpanan, dimana suatu proses dapat membaca dan menulis pada *shared storage* (main memory atau files)

Masalah – masalah pada IPC :

Race Condition :
Suatu kondisi dimana dua atau lebih proses mengakses *shared memory* / data pada saat yang bersamaan dan hasil akhirnya tidak sesuai dengan yang dikehendaki
Contoh rase condition :

- Print spooler
  Contoh : berupa kumpulan data-data yang akan di cetak.
- Spooler directory



- Proses A → cek slot input yang kosong (7) untuk mencetak suatu data dan stop
- Interupsi …….
- Proses B → meletakkan data yang akan di print pada slot kosong tersebut (7) dan stop (slot kosong berikutnya adalah 7+1=8)
- : prosesor mengerjakan proses lain
- Proses A dilanjutkan → meletakkan data yang akan di print di slot (7), sehingga meng-overwrite data proses B yang diletakkan di slot (7)
- Maka proses B tidak akan dilaksanakan, dan tidak akan terdeteksi terjadi kesalahan.

Untuk menghindari race condition maka harus diatur agar 2 proses yang mempunyai critical section yang sama tidak memasuki critical section pada saat yang bersamaan.

Critical Section / seksi kritis :
Bagian dari program yang mengakses shared memory, yang dapat menyebabkan terjadinya race condition.

4 kondisi untuk mencegah race condition :
a. Tidak ada 2 proses yang memasuki critical sectionnya secara bersamaan / simultan
b. Tidak ada asumsi yang dibuat yang berhubungan dengan kecepatan dan jumlah CPU
c. Tidak ada proses yang berjalan diluar critical section-nya yang dapat memblokir proses-proses lain
d. Tidak ada proses yang menunggu selamanya untuk masuk ke critical section-nya.

Mutual Exclusion (MuTex) With Busy Waiting :
Jika suatu proses sedang mengakses shared memory di critical sectionnya, tidak ada satu prosespun yang dapat memasuki critical section (mutual exclusion) dan menyebabkan masalah.

Jenis-jenis mutual exclusion :

1. Disabling interrupt / mematikan interupsi
Dengan cara mematikan interupsi yang masuk pada saat proses sedang berada pada critical section-nya. Cara ini kadang cukup berguna untuk kernel tetapi tidak untuk user. Dan cara inipun tidak terlalu baik untuk CPU yang jumlahnya lebih dari satu, dimana disable interrupt hanya mengenai CPU yang sedang menjalankan proses itu dan tidak berpengaruh terhadap CPU lain

2. Lock variables
Setiap proses yang akan mengakses ke critical section-nya harus meng-cek lock variable. Jika 0 berarti proses dapat memasuki critical section-nya dan jika 1 maka proses harus menunggu sampai lock variable = 0. Kelemahannya adalah 2 proses masih dapat memasuki critical section-nya pada saat yang bersamaan. Sewaktu satu proses meng-cek lock variable = 0, pada saat akan men-set 1 ada interupsi untuk melaksanakan proses lain yang juga ingin memasuki critical sectionnya, maka akan terjadi race condition.

3. Strict alternation
Dengan mengamati variable turn untuk menentukan siapa yang akan memasuki critical section-nya bukanlah ide yang baik jika proses lebih lambat dari yang lain.
Contohnya :

```
While (true)
{
        while (turn != 0)               /*wait*/;
        critical_section ( );
        turn = 1;
        noncritical_section ( );
}
while (true)
{
        while (turn != 1)               /*wait*/;
        critical_section ( );
        turn = 0;
    noncritical_section ( );
}
```

4. Peterson's Solution
Proses tidak akan diteruskan sampai while terpenuhi, bila interested[other] = TRUE, maka proses akan menunggu sampai FALSE.
Kelemahannya : jika proses memanggil enter_region-nya secara hampir bersamaan, yang disimpan di turn adalah data yang ditulis terakhir.
Contohnya :

```
# include "prototype.h"
# define FALSE          0
# define TRUE 1
# define N              2           /*banyaknya proses*/

int turn;
int interested [N];                 /*nilai awal di-set = 0 (false)*/
void enter_region(int process)      /*proses = 1 atau 0*/
```

```
{
int other;                              /*jumlah proses lainnya*/
other = 1 – process;                    /*proses lainnya*/
interested[process] = TRUE;             /*menunjukkan tertarik*/
turn = process;                         /*set flag*/
while (turn==process && interested[other] == TRUE)
}

void leave_region(int process)          /*proses yang selesai*/
{
    interested[process] = FALSE;        /*meninggalkan critical region*/
}
```

5. Test and Set Lock Instruction / Instruksi TSL

Dengan bantuan hardware, menentukan siapa yang berhak memasuki critical_region (section)
Contoh :

```
Enter_region :
    Tsl reg,flag                | copy flag ke reg dan set flag = 1
    Cmp reg,#0                  | apakah flag = 0
    Jnz enter_region            |jika <> 0 loop lagi
        Ret                     |return ke caller, masuk critical region

Leave_region :
        Mov flag, #0            |simpan 0 ke flag
        Ret                     |return ke caller
```

Proses harus memanggil ini pada saat yang tepat.
Kelemahan utama dengan busy waiting adalah menyita banyak waktu CPU dan problem inversi prioritas.

6. Sleep and Wake Up

Mekanismenya : proses akan di blok / tidur (sleep) apabila tidak bisa memasuki critical_section-nya dan akan dibangunkan (wake up) / ready apabila resource yang diperlukan telah tersedia.
SLEEP : sistem call membuat proses yang memanggil di blok (blocked)
WAKE UP : sistem call yang membuat proses yang memanggil menjasi ready
Contoh :
Procedure-Consumer Problem (bounded buffer)
Beberapa proses share buffer dengan ukuran tetap
- Jika buffer penuh producer sleep
- Jika buffer kosong consumer sleep
- Jika buffer mulai kosong producer wake up
- Jika buffer terisi consumer wake up

Masih ada kemungkinan terjadi race condition

7. Semaphore (Dijkstra, 1965)

Meng-cek, mengubah dan sleep — 1 instruksi yang
Mengubah dan wake up — tdk dpt dipisahkan
Instruksi tersebut sangat berguna untuk sinkronisasi.
Dapat diimplementasikan untuk memecahkan producer-consumer problem.
Mekanisme-nya menggunakan :
- variabel integer untuk menghitung jumlah wake up yang disimpan / tertunda
- bernilai 0 bila tidak ada wake up yang disimpan, bernilai positif bila ada wake up yang tertunda

Dua macam operasi terhadap semaphore :
1. DOWN(S) :

```
If S >= 0 then
        S := S-1;
        Else sleep (S)
End;
```

2. UP(S) :

```
S := S + 1;
If S <= 0 then wakeup(S)
End;
```

Operasi DOWN dan UP merupakan operasi yang bersifat Atomic (Atomic Action).

8. Event Counters (Reed and Kanodia, 1979)

Tiga operasi terhadap event counter (E) :
1. Read (E) : return current value of E
2. Advance (E) : Atomically increment E by 1
3. Wait until E has a value of v or more

9. Monitor

- Higher level synchronization primitive.
- Kumpulan prosedur, variabel dan struktur data yang dipaket menjadi satu modul atau paket.
- Proses bisa memanggil prosedur dalam monitor, tetapi tidak dapat mengakses langsung struktur data internal dari monitor.

10. Message Passing

Menggunakan 2 primitive :
1. send (destination, &message)
2. receive (source, &message)

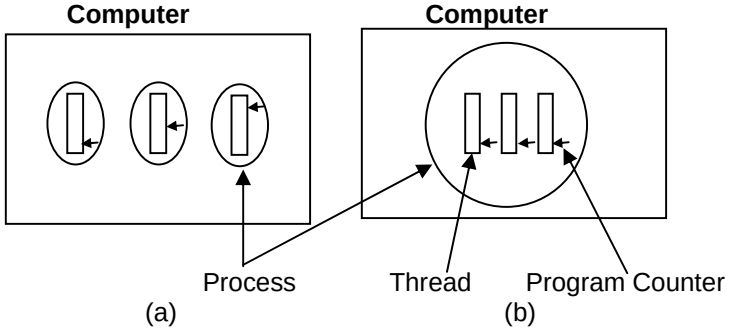
Beberapa isu pada message passing system : message lost; acknowledgement; domains; authentication; performance

Masalah Klasik IPC :

- *The Dining Philosopher Problem*
  - 5 philosophers yang kerjanya hanya makan dan berfikir
  - tersedia lima piring spaghetti dan lima sumpit
  - untuk makan dibutuhkan bua buah sumpit
  - problem-nya bagaimana cara menulis program agar setiap philosopher dapat berfikir dan makan tanpa harus saling menunggu ?
- *The Readers and Writers Problem*
  - Model akses database
  - Banyak proses berkompetisi untuk membaca dan menulis. Contohnya : airline reservation.
  - Beberapa proses boleh membaca pada saat yang sama
  - Bila suatu proses sedang menulis, tidak boleh ada proses lain yang mengakses database
  - Proses membaca mempunyai prioritas yang lebih tinggi daripada proses menulis

Proses dalam Sistem Terdistribusi

Thread



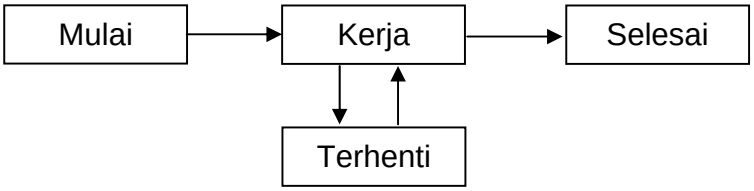
Gambar (a) :
- mempunyai : program counter, stack, register set, address space sendiri
- independent satu sama lain dan berkomunikasi lewat IPC yang disediakan sistem, seperti : semaphore, monitor, atau message

Gambar (b) :
- multiple threads of control (THREAD atau lightweight Process). Thread mirip seperti little-mini process. Setiap thread berjalan sekuensial, yang mempunyai program counter dan stack sendiri. Thread juga men-share CPU seperti proses.
- Thread dalam satu proses menempati address space yang sama, tidak ada proteksi penggunaan memori antar thread karena proses dimiliki oleh satu user.
- Thread dapat berada pada empat state yang berbeda, seperti process (running, blocked, ready, terminated)

Ada 3 model process pada server :
1. thread di ciptakan untuk dapat melakukan paralelisme yang dikombinasikan dengan eksekusi sekuensial dan blocking system calls
2. single treads server, menggunakan blocking system calls, tetapi kinerja sistem tidak baik
3. finite-state machine, kinerja baik dengan melakukan parallelisme, tetapi menggunakan nonblocking calls, sehingga sulit dalam memprogram

Status proses terhadap prosesor :
1. Status proses tanpa henti

2. Status proses sambil bekerja dan terhenti



3. Status proses dengan status siap