# Proses

suhendi@nurulfikri.ac.id

# Proses

- Konsep Proses
- Penjadualan Eksekusi Proses
- Operasi pada Proses
- Proses yang saling Bekerjasama (Cooperating Processes)
- Komunikasi Antar Proses (Interprocess Communication)
- Komunikasi pada Sistem Client-Server

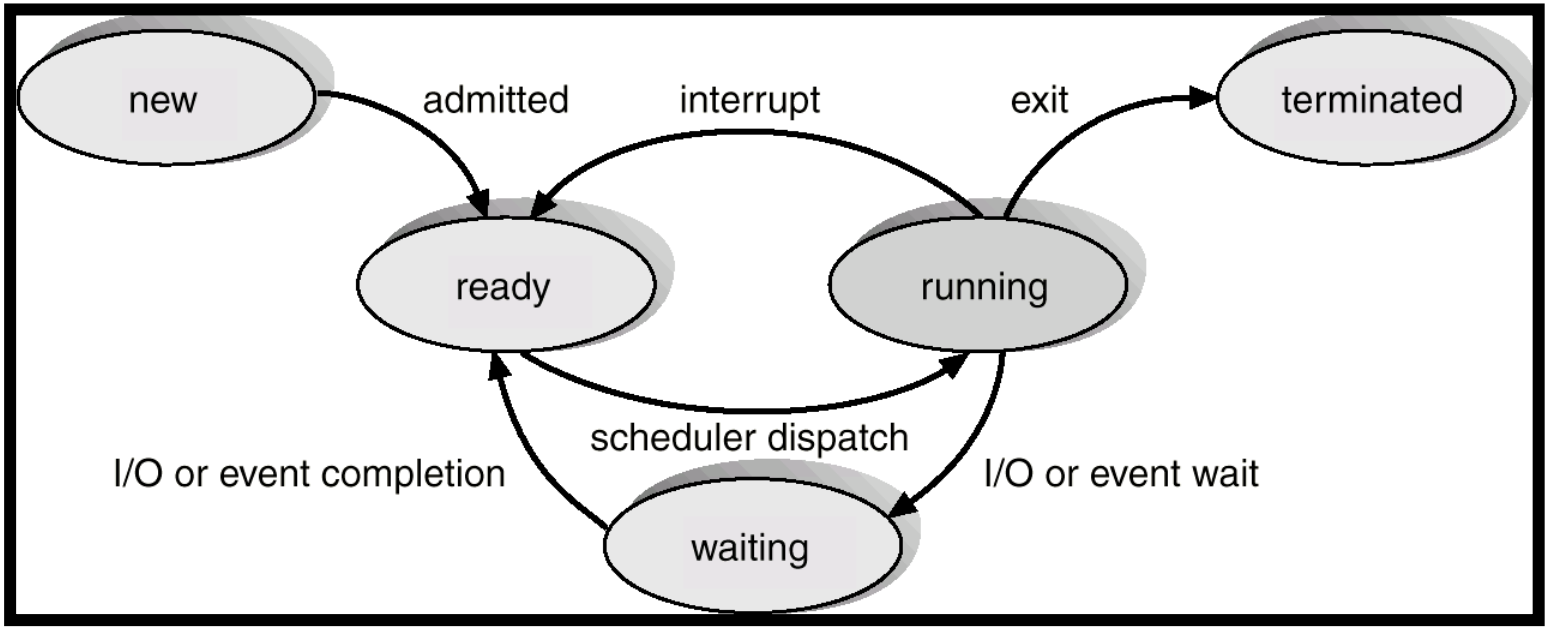
# Konsep Proses

- Sistem operasi menjalankan banyak dan **beragam** program :
  - Batch system – jobs
  - Time-shared systems – user programs atau tasks
  - Istilah pada buku teks: *job, task* dan *process* (dapat diartikan sama)
- Proses adalah program yang dieksekusi ;
  - Aktif (proses=>memori) vs pasif (program => file)
  - Instruksi pada program (code) akan dieksekusi secara berurut (sekwensial) sesuai dengan “line code” (stored program concept).
- Proses lebih dari “program code yang aktif”:
  - Melacak posisi instruksi (sequential execution): program counter
  - Menyimpan data sementara var., parameter, return value: stack
  - Menyimpan data (initial, global variable dll): data section
  - Menyimpan status proses (contoh, aktif, wait I/O request dll.)

# Status Proses

- Saat-saat proses dijalankan (executed) maka status dari proses akan berubah
  - Status proses tidak selamanya aktif menggunakan CPU).
  - Sering proses menunggu I/O complete => status wait, sebaiknya CPU diberikan kepada proses yang lain.
  - Mendukung multi-tasking – utilisasi CPU dan I/O
- Status proses (antara lain):
  - **new**: proses dibuat.
  - **running**: instruksi dieksekusi.
  - **waiting**: proses menunggu beberapa event yang akan terjadi
  - **ready**: proses menunggu jatah waktu dari prosessor
  - **terminated**: proses selesai dieksekusi.

# Diagram Status Proses

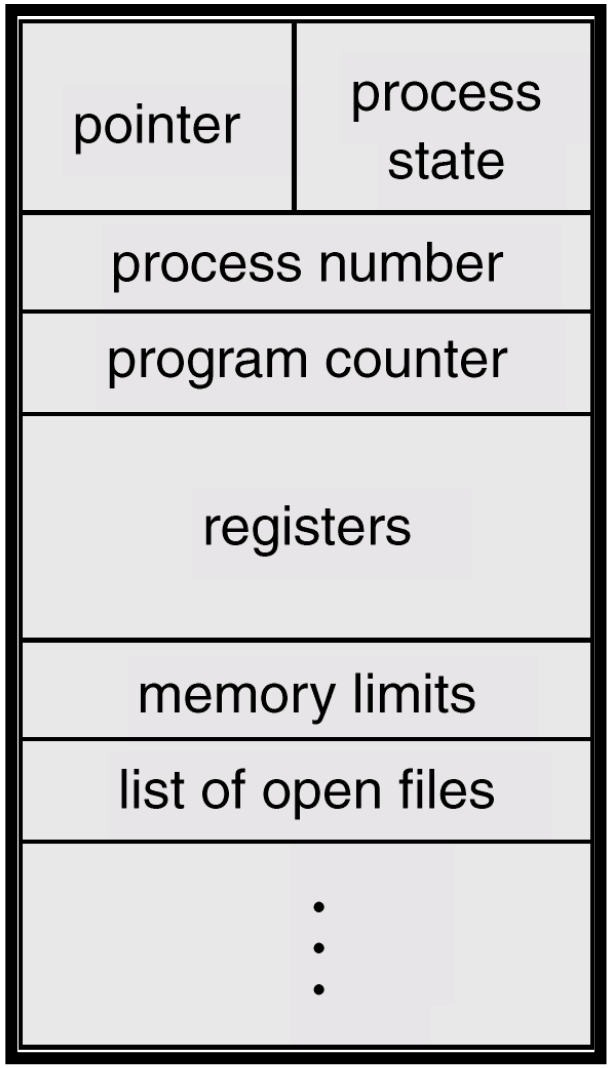


# Informasi Proses

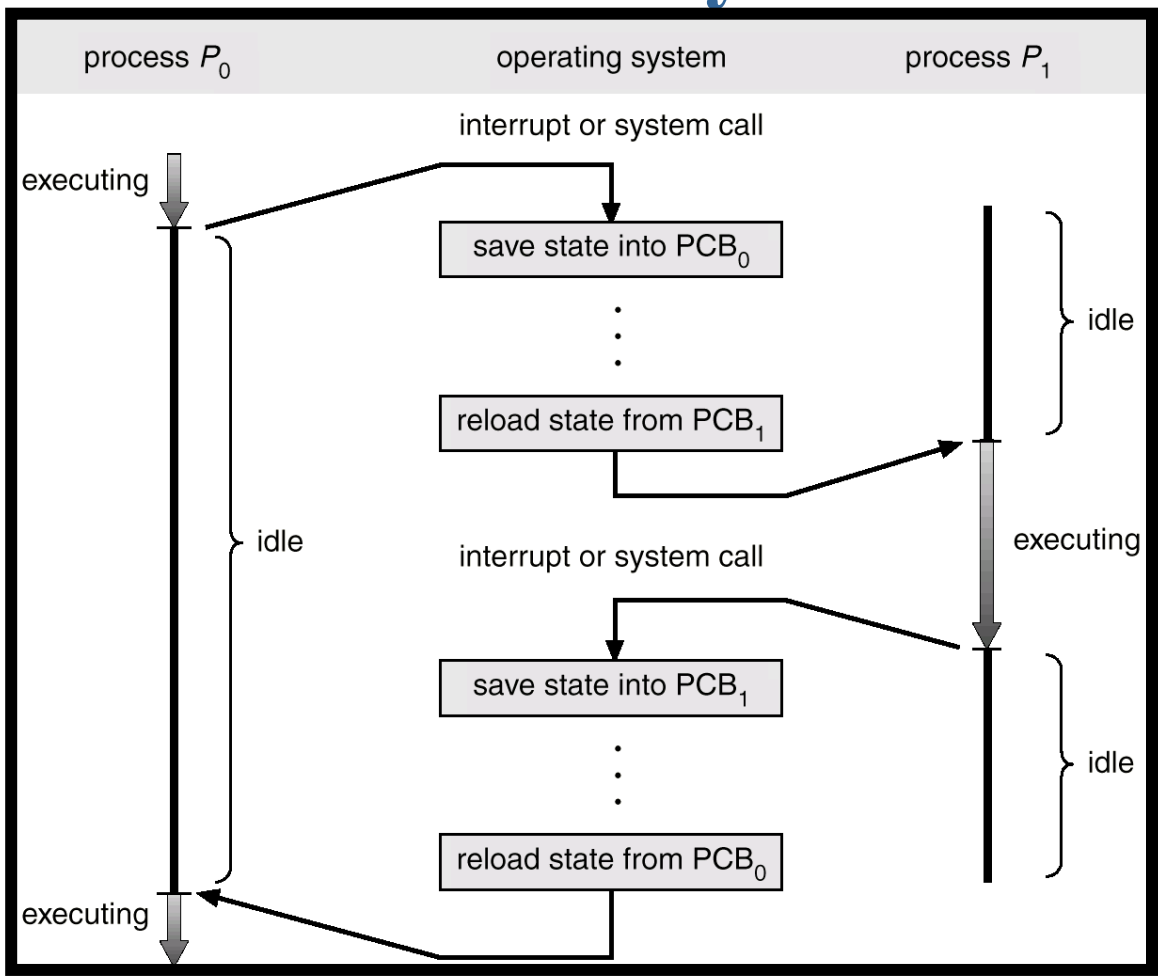
Dimanakah informasi proses disimpan?

- Data struktur dari OS dalam bentuk table :
  - Satu entry table/linked list => struktur data untuk menampung informasi satu proses (array of structure).
  - Setiap entry pada tabel proses menyimpan satu proses. Contoh: MINIX (src/kernel/proc.h) => struct proc { … };
- Informasi yang disimpan:
  - Informasi internal CPU: isi register-register, program counter, status CPU dll (umumnya dalam bentuk stack frame).
  - Identifikasi proses: nama proses, proses number/index, proses id.
  - Identifikasi proses: nama proses, proses number/index, proses id.
  - Accounting dan timer: user time, system time, alarm etc.
  - Resources: memory & file management.

# Process Control Block (PCB)



# CPU Switch Dari Satu Proses ke Proses Lainnya



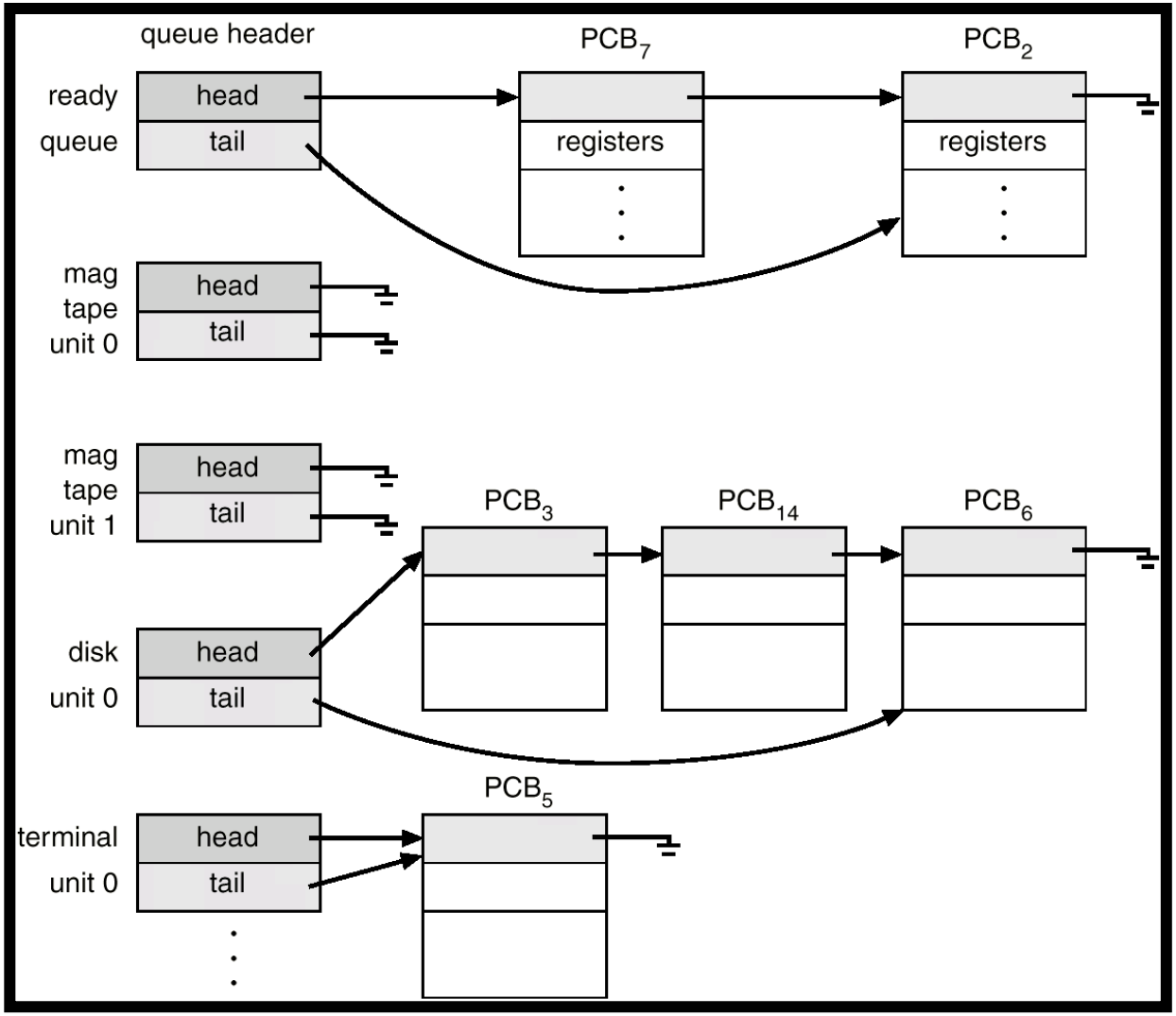
# Penjadualan Proses

- Apakah tujuan  dari multiprogramming?
  - “Maximize” pemakaian CPU secara efisien (jadwal dan giliran pemakaian CPU).

    => CPU digunakan oleh proses-proses terus menerus
- Apakah tujuan dari “time-sharing”?
  - Pemakaian CPU dapat di switch dari satu proses ke proses lain (concurrent process execution)

    => sesering mungkin, user dapat berinteraksi dengan sistim
- Bagaimana jika sistim prosesor tunggal?
  - “Hanya ada satu proses yang dapat dijalankan”
  - Proses lain menunggu sampai CPU dapat dijadwalkan (schedule) ke proses tsb

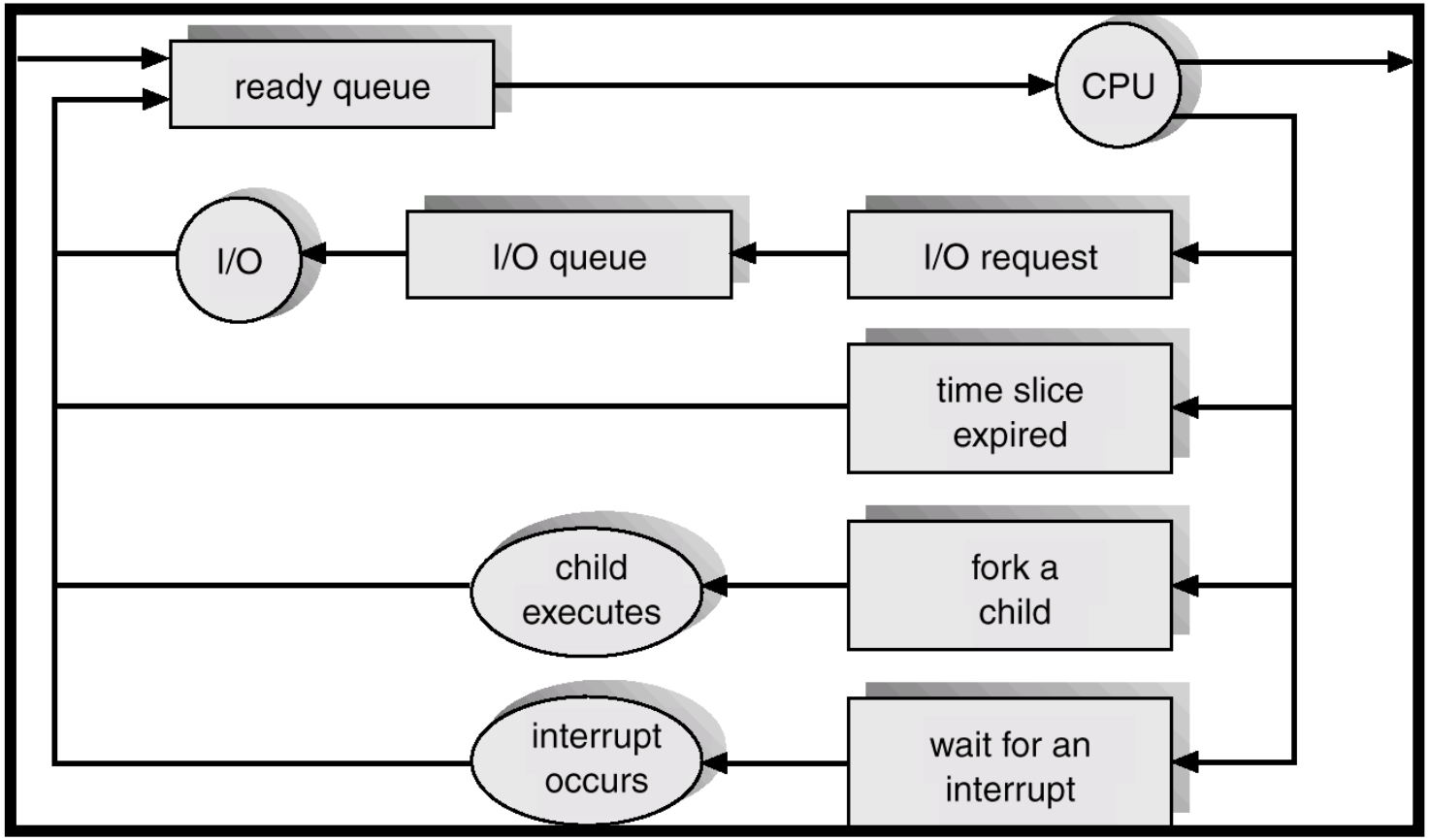
# Ready Queue dan I/O Device Queues



# Penjadualan Proses

- Proses dapat berubah status dan berpindah dari satu antrian ke antrian yang lain
  - Proses dengan status “ready” berada di ReadyQueue
    - Menunggu giliran/dipilih oleh scheduler => menggunakan CPU
  - Selama eksekusi (status “run”) events yang dapat terjadi:
    - I/O request => I/O wait berada pada DeviceQueue
    - Create “child” proses => Jalankan proses “child”, tunggu sampai proses selesai (wait)
    - Time slice expired => Waktu pemakaian CPU habis, interrupt oleh scheduler, proses akan berpindah ke ReadyQueue

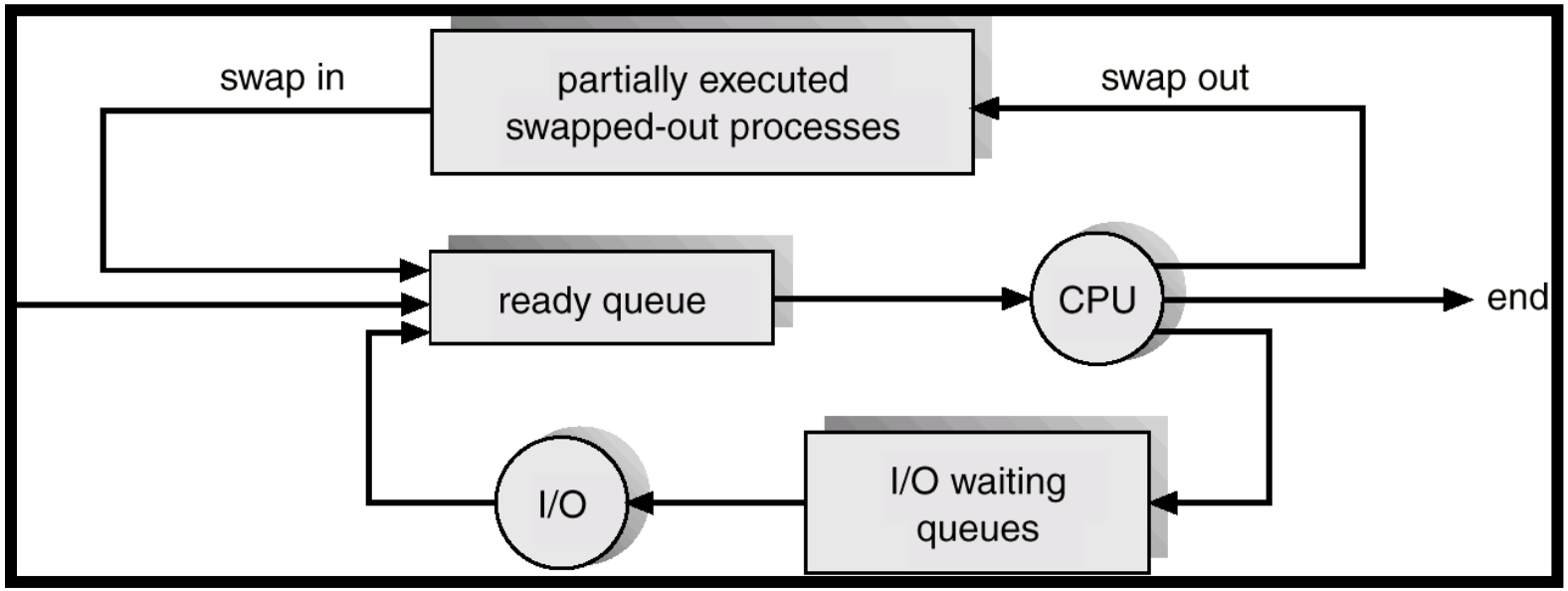
# Representasi Penjadualan Proses



# Penjadual / Schedulers

- Bagaimana schedulers memilih proses atau program (decision)?
  - Lebih dari satu proses atau program yang akan dijalankan?
- Long-term scheduler (or job scheduler) – memilih proses/program yang mana yang akan di load dan berada di **ready queue**.
  - Kemungkinan terdapat proses atau job baru.
  - Kemungkinan proses dipindahkan dari memori ke disk (swap out).
- Short-term scheduler (or CPU scheduler) – memilih proses yang mana yang berada di **ready queue** akan “run” (mendapatkan jatah CPU).

# Penjadualan Jangka Menengah



# Penjadual / Schedulers (Cont.)

- Long-term scheduler tidak sering (proses baru) (seconds, minutes) => (may be slow).
  - The long-term scheduler controls the *degree of multiprogramming* => *berapa banyak proses yang dapat aktif (berada di memori)*
- Short-term scheduler dijalankan sangat sering (milliseconds) => giliran pemakaian CPU dari proses- proses yang siap
  - Pada saat terjadi penggantian alokasi CPU dari satu proses ke proses lain:
    - Menyimpan informasi internal CPU dari proses yang akan digantikan (SAVE).
    - Meload kembali informasi internal CPU dari proses yang akan menggantikan.
  - Dikenal dengan istilah: context switch proses.

# Alih Konteks / Context Switch

- Jika Scheduler switch ke proses lain, maka sistim harus menyimpan “informasi” proses sekarang (supaya dapat dijalankan kembali)
- Load “informasi” dari proses baru yang berada di PCB
- Waktu Context-switch adalah overhead; sistem tidak melakukan pekerjaan saat terjadi switch.
  - Sangat tergantung pada waktu di hardware
  - OS modern mencari solusi untuk mengurangi overhead waktu switch proses

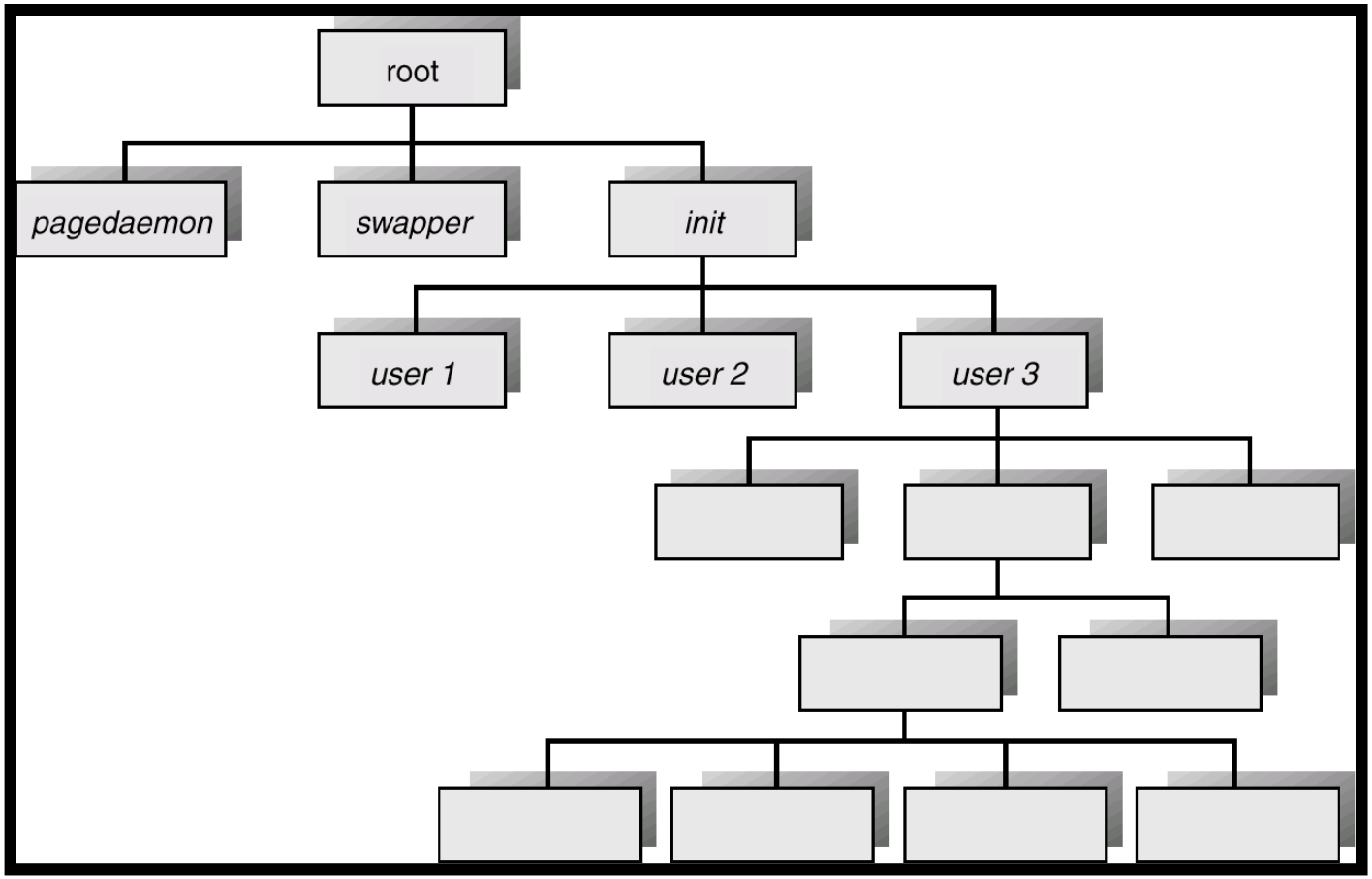
# Pembuatan Proses

- Umumnya proses dapat membuat proses baru (child process).
  - Child process dapat membuat proses baru.
  - Terbentuk “tree” dari proses.
- Pilihan hubungan antara parent dan child proses:
  - Resource sharing
    - Parent dan child berbagi resource
    - Children berbagi subset dari resource milik parents.
    - Parent dan child tidak berbagi resource.
  - Execution
    - Parent dan children melakukan eksekusi secara serempak.
    - Parent menunggu hingga children selesai.

# Pembuatan Proses (Cont.)

- Address space
  - Child menduplikasi parent.
  - Child memiliki program yang di load ke dalamnya.
- Contoh UNIX :
  - **fork** system call membuat proses baru
  - **execve (EXEC)** :
    - menjalankan program spesifik yang lain
    - nama program tersebut menjadi parameter dari system call
    - EXEC (sering di load sesudah menjalankan fork).
  - Tahapan pembuatan proses baru:
    - Periksa apakah masih terdapat ruang pada PCB.
    - Mencoba mengalokasikan memori untuk proses baru.
    - Mengisi informasi untuk proses baru: nama proses, id, copy data dari parent dll.
    - Mencantumkan informasi proses ke kernel OS.

# Proses Tree pada Sistem UNIX



# Terminasi Proses

- Proses dapat berakhir:
  - Eksekusi instruksi terakhir (atau keluar: exit system call).
  - OS yang akan melakukan dealokasi (memory, file resources).
- UNIX (MINIX):
  - Output signal dari child ke parent
  - Jika parent tidak menunggu (via **wait system call**), proses akan terminate tapi belum di release dari PCB (status: ZOMBIE).
  - Proses dengan status ZOMBIE (parent telah terminate), akan menjadi child dari proses “init”.
- Parent dapat menghentikan eksekusi proses child secara paksa.
  - Parent dapat mengirim signal (**abort, kill system call**).

# Kerjasama Proses

- Proses independent tidak mempengaruhi eksekusi proses yang lain
- Kerjasama proses dapat mempengaruhi  atau dipengaruhi oleh eksekusi proses yang lain
- Keuntungan kerjasama proses :
  - Sharing informasi
  - Meningkatkan kecepatan komputasi
  - Modularitas
  - Kemudahan

# Masalah Producer-Consumer

- Paradigma kerjasama proses – proses Producer menghasilkan informasi yang akan dikonsumsi oleh proses Consumer
  - Unbounded-buffer – tidak menggunakan batasan ukuran di buffer.
    - Consumer selalu dapat meminta item baru dan Producer selalu dapat menghasilkan item-item baru.
  - Bounded-buffer – menggunakan buffer dengan ukuran tertentu
    - Consumer harus menunggu jika buffer kosong dan Producer harus menunggu jika buffer penuh

# Bounded-Buffer – Solusi dari Shared Memory

- Shared data

```
#define BUFFER_SIZE 10
Typedef struct {
   . . .
} item;
item buffer[BUFFER_SIZE];
int in = 0;
int out = 0;
```

- Solution is correct, but can only use BUFFER_SIZE-1 elements

# Bounded-Buffer – Proses Producer

```
item nextProduced;

while (1) {
 while (((in + 1) % BUFFER_SIZE) == out)
       ; /* do nothing */
 buffer[in] = nextProduced;
 in = (in + 1) % BUFFER_SIZE;
}
```

# Bounded-Buffer – Proses Consumer

```
item nextConsumed;

while (1) {
  while (in == out)
        ; /* do nothing */
  nextConsumed = buffer[out];
  out = (out + 1) % BUFFER_SIZE;
}
```

# Interprocess Communication (IPC)

- Mekanisme proses untuk komunikasi dan sinkronisasi aksi
- Sistem Pesan – komunikasi proses satu dengan yang lain dapat dilakukan tanpa perlu pembagian data.
- IPC menyediakan dua operasi :
  - **send**(*message*) – pesan berukuran pasti atau variabel
  - **receive**(*message*)
- Jika P dan Q melakukan komunikasi, maka keduanya memerlukan :
  - Membangun jalur komunikasi diantara keduanya
  - Melakukan pertukaran pesan melaui send/receive
- Implementasi jalur komunikasi
  - physical (shared memory, hardware bus)
  - logical (logical properties)

# Komunikasi Langsung

- Proses harus diberi nama secara jelas :
  - **send** (*P, message*) – kirim pesan ke proses P
  - **receive**(*Q, message*) – terima pesan dari proses Q
- Properti jalur komunikasi
  - Jalur dibangun secara otomatis
  - Setiap jalur memiliki pasangan masing-masing dalam proses komunikasi
  - Jalur komunikasi tersebut biasanya directional

# Komunikasi Tidak Langsung

- Pesan dikirim dan diterima melalui mailboxes (yang ditunjuk sebagai port)
  - Proses
  - Processes can communicate only if they share a mailbox.
- Properti jalur komunikasi
  - Jalur komunikasi hanya dibangun jika proses di-share dalam mailbox
  - Jalur merupakan gabungan beberapa proses
  - Setiap pasangan proses dibagi ke dalam beberapa jalur komunikasi.

# Komunikasi Tidak Langsung

- Operasi
  - Membuat mailbox baru
  - Mengirim dan menerima pesan melalui mailbox
  - Menghapus/memusnahkan mailbox
- Primitive didefinisikan :

  **send**(*A, message*) – kirim pesan ke mailbox A

  **receive**(*A, message*) – terima pesan dari mailbox A

# Komunikasi Tidak Langsung

- Mailbox sharing
  - $P_1$, $P_2$, dan $P_3$ berbagi (share) mailbox A.
  - $P_1$, send; $P_2$ and $P_3$ receive.
  - Siapa yang mendapat pesan ?
- Solusi
  - Memperbolehkan suatu jalur yang merupakan gabungan lebih dari dua proses
  - Hanya meperbolehkan satu proses pada suatu waktu untuk mengeksekusi operasi receive .
  - Memperbolehkan sistem untuk memilih receiver. Sender diberitahu siapa yang menjadi receiver.

# Sinkronisasi

- Pesan yang disampaikan dapat di blok atau tidak (non-blocking)
- **Blocking** dikenal dengan synchronous.
- **Non-blocking** dikenal dengan **asynchronous**

# Buffering

- Antrian pesan yang dihubungkan dalam suatu jalur, diimplementasikan dengan tiga jalan :

1. Zero capacity – tidak ada pesan
   - Sender harus menunggu receiver (rendezvous).
2. Bounded capacity – memiliki panjang yang terbatas (finite length) dari *n* pesan.
   - Sender menunggu pada saat jalur penuh.
3. Unbounded capacity – memiliki panjang tidak terbatas (infinite length)
   - Sender tidak pernah menunggu.

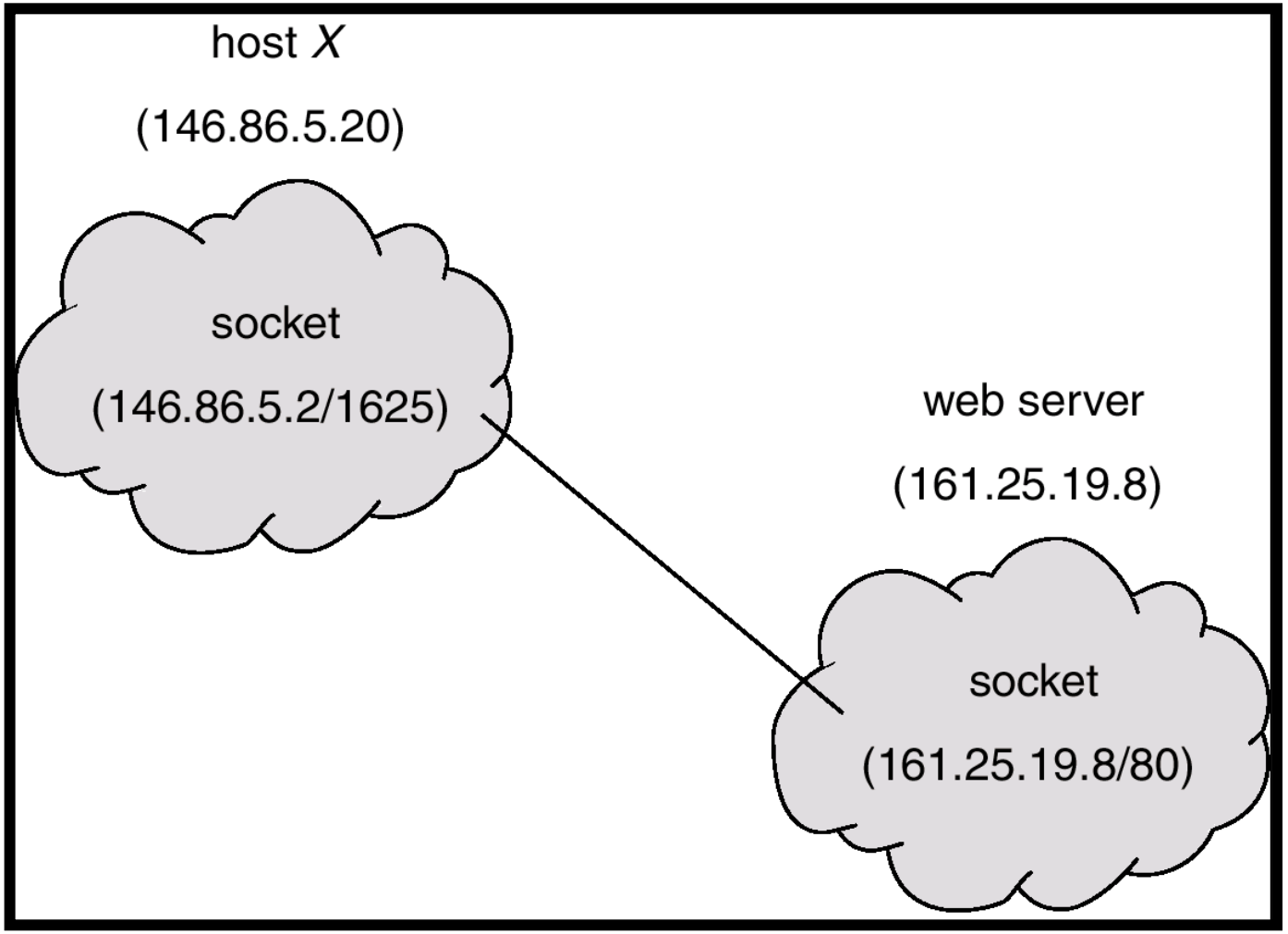
# Komunikasi Client-Server

- Sockets
- Remote Procedure Calls (RPC)
- Remote Method Invocation (Java)

# Sockets

- Suatu socket didefinisikan sebagai titik akhir (endpoint) komunikasi
- A socket is defined as an *endpoint for communication*.
- Gabungan IP address dan port
- Socket **161.25.19.8:1625** mengacu pada port **1625** pada host **161.25.19.8**
- Komunikasi berada diantara pasangan socket

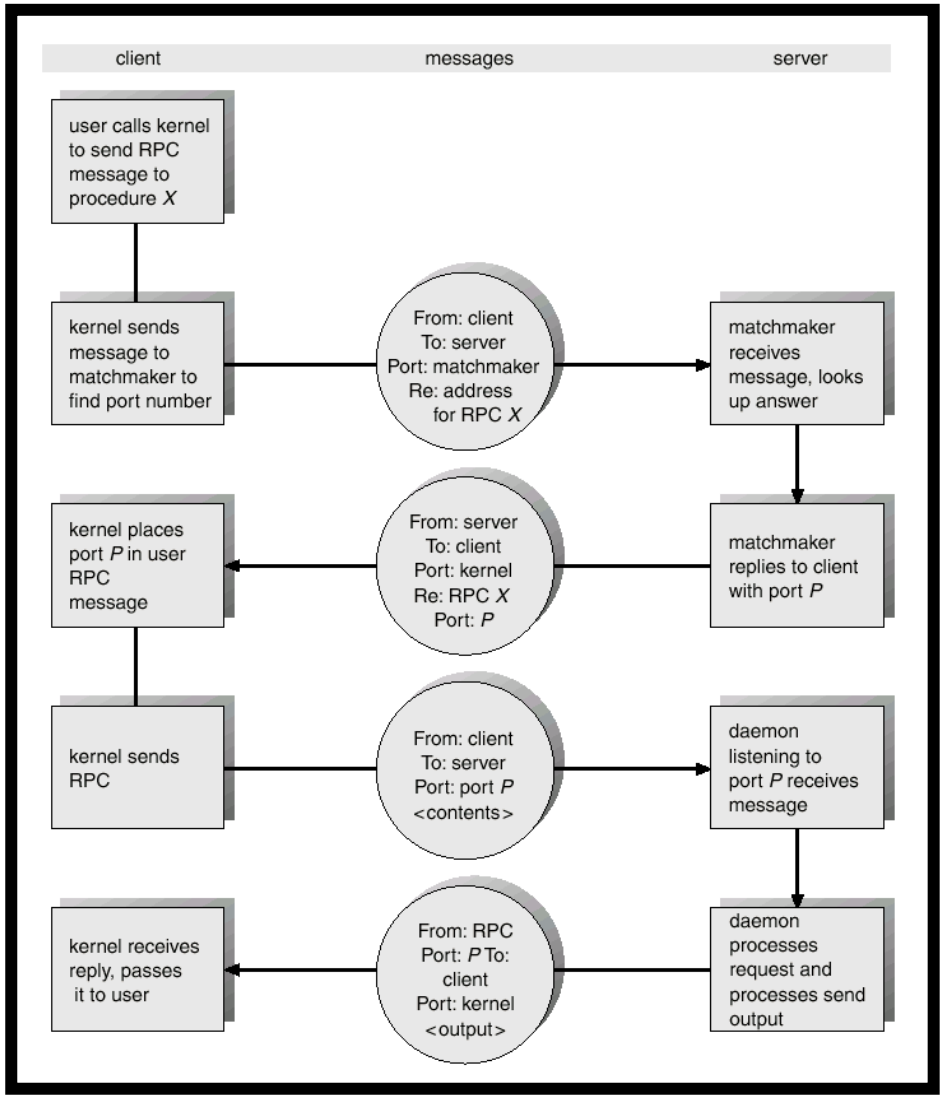
# Komunikasi Socket



# Remote Procedure Calls (RPC)

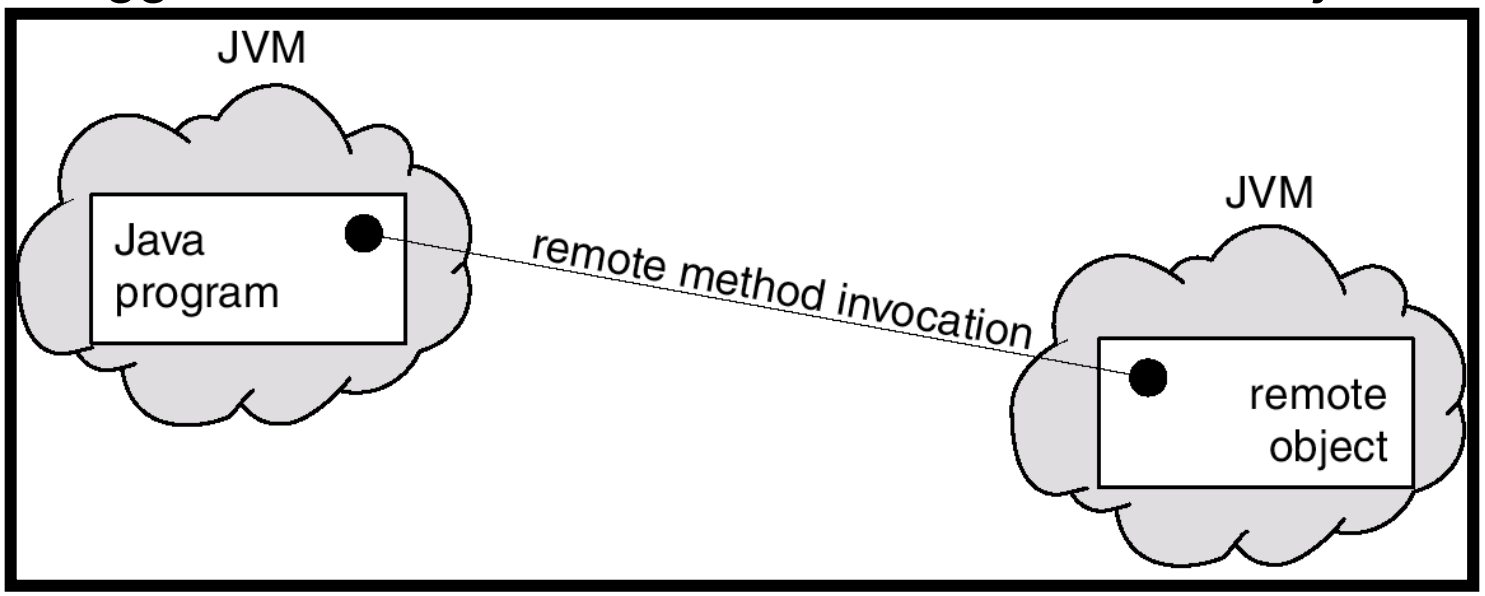
- Remote Procedure Call (RPC) adalah abstraksi pemanggilan prosedur diantara proses pada sistem jaringan
- **Stubs** – proxy sisi client untuk prosedur aktual pada server
- Stub sisi client ditempatkan di server dengan parameter *marshalls*.
- Stub sisi server menerima pesan, membongkarnya dengan parameter marshall dan menjalankan prosedur pada server.

# Eksekusi RPC

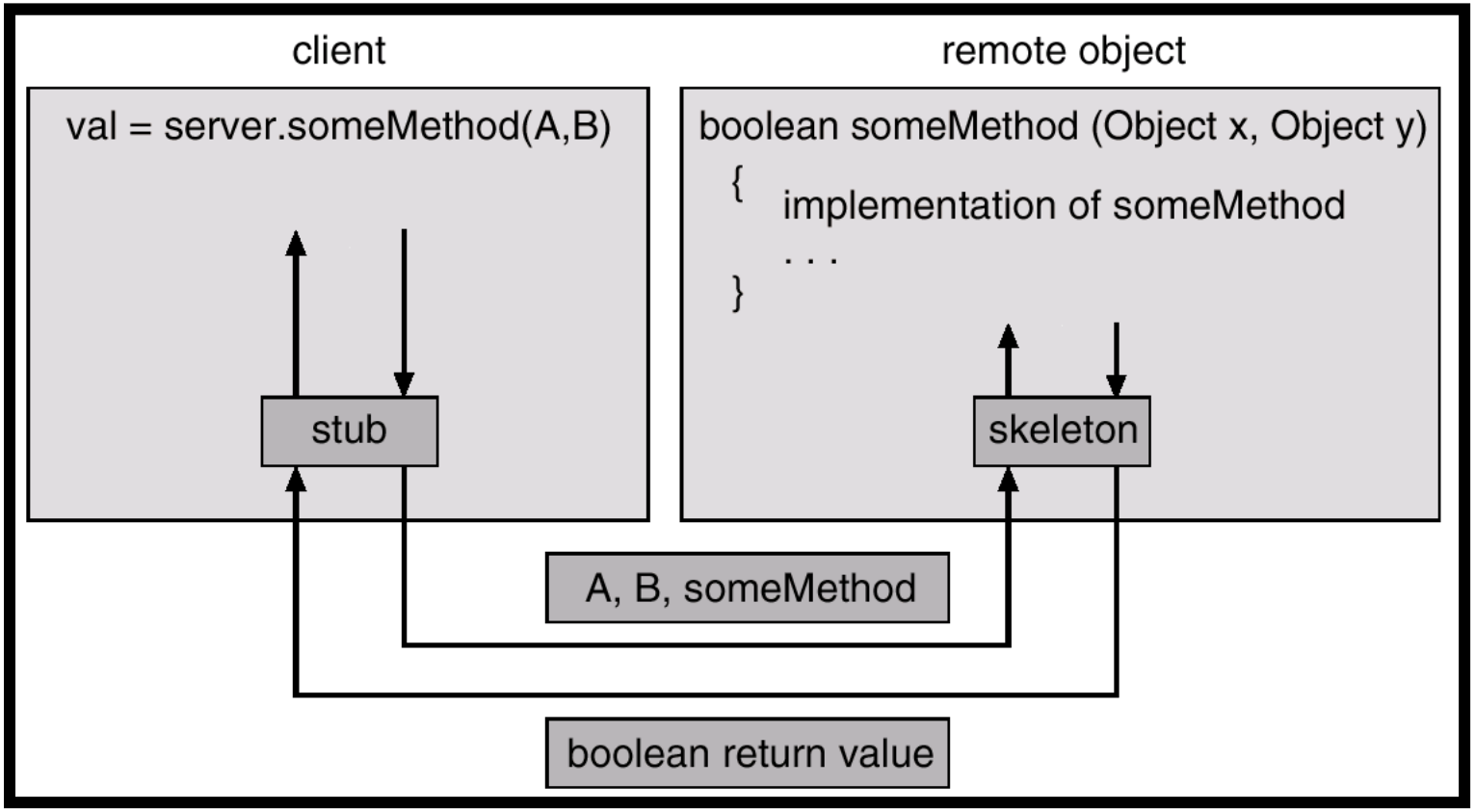


# Remote Method Invocation (RMI)

- Remote Method Invocation (RMI) adalah mekanisme pada JAVA yang hampir sama dengan RPC
- RMI membolehkan program JAVA pada satu mesin untuk menggunakan metode untuk melakukan remote objek.



# Parameter Marshall



# Referensi


1. Silberschatz,Galvin and Gagne ,Operating Systems Concepts, 2002
2. Kusnadi,S.T,M.Eng.Sc,Sistem Operasi,Penerbit Andi, 2008
3. Bambang Hariyanto,Ir.MT,Penerbit Informatika Bandung, 2007
4. Abas Ali Pangera, Dony Ariyus, Penerbit Andi,2010

# (B03-2003-01) Tabel Proses I

Berikut merupakan sebagian dari keluaran menjalankan perintah ``**top b n 1**" pada sebuah sistem GNU/Linux yaitu "**bunga.mhs.cs.ui.ac.id**" pada tanggal 10 Juni 2003 yang lalu.

```
16:22:04 up 71 days, 23:40,  8 users,  load average: 0.06, 0.02, 0.00
58 processes: 57 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  15.1% user,   2.4% system,   0.0% nice,  82.5% idle
Mem:    127236K total,   122624K used,     4612K free,     2700K buffers
Swap:   263160K total,     5648K used,   257512K free,    53792K cached

  PID USER   PRI NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
    1 root     0  0   112   72    56 S     0.0  0.0   0:11 init
    2 root     0  0     0    0     0 SW    0.0  0.0   0:03 kflushd
    4 root     0  0     0    0     0 SW    0.0  0.0 156:14 kswapd
   ...
14953 root     0  0   596  308   236 S     0.0  0.2  19:12 sshd
31563 daemon   0  0   272  256   220 S     0.0  0.2   0:02 portmap
 1133 user1   18  0  2176 2176  1752 R     8.1  1.7   0:00 top
 1112 user1    0  0  2540 2492  2144 S     0.0  1.9   0:00 sshd
 1113 user1    7  0  2480 2480  2028 S     0.0  1.9   0:00 bash
30740 user2    0  0  2500 2440  2048 S     0.0  1.9   0:00 sshd
30741 user2    0  0  2456 2456  2024 S     0.0  1.9   0:00 bash
30953 user3    0  0  2500 2440  2072 S     0.0  1.9   0:00 sshd
30954 user3    0  0  2492 2492  2032 S     0.0  1.9   0:00 bash
 1109 user3    0  0  3840 3840  3132 S     0.0  3.0   0:01 pine
...
 1103 user8    0  0  2684 2684  1944 S     0.0  2.1   0:00 tin
```

a. Jam berapakah program tersebut di atas dijalankan?
b. Berapa waktu sebelumnya (dari tanggal 10 Juni tersebut), server "bunga.mhs.cs.ui.ac.id" terakhir kali *(re)boot*?
c. Apakah yang dimaksud dengan "*load average*"?
d. Sebutkan nama dari sebuah proses di atas yang statusnya "*running*"!
e. Sebutkan nama dari sebuah proses di atas yang statusnya "*waiting*"!

# (B03-2003-02) Tabel Proses II

```
15:34:14 up 28 days, 14:40, 53 users,  load average: 0.28, 0.31, 0.26
265 processes: 264 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:   5.9% user,   1.8% system,   0.1% nice,  92.2% idle
Mem:    126624K total,   113548K used,    13076K free,      680K buffers
Swap:   263160K total,    58136K used,   205024K free,    41220K cached

PID   USER  PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
    1 root    8   0   460  420   408 S     0.0  0.3   0:56 init
    2 root    9   0     0    0     0 SW    0.0  0.0   0:02 keventd
.....
17353 user1   9   0  2500 2004  2004 S     0.0  1.5   0:00 sshd
17354 user1   9   0  1716 1392  1392 S     0.0  1.0   0:00 bash
17355 user1   9   0  2840 2416  2332 S     0.0  1.9   0:00 pine
12851 user2   9   0  2500 2004  2004 S     0.0  1.5   0:00 sshd
12852 user2   9   0  1776 1436  1436 S     0.0  1.1   0:00 bash
13184 user2   9   0  1792 1076  1076 S     0.0  0.8   0:00 vi
13185 user2   9   0   392  316   316 S     0.0  0.2   0:00 grep
22272 user3   9   0  2604 2592  2292 S     0.0  2.0   0:00 sshd
22273 user3   9   0  1724 1724  1396 S     0.0  1.3   0:00 bash
22283 user3  14   0   980  980   660 R    20.4  0.7   0:00 top
19855 user4   9   0  2476 2048  1996 S     0.0  1.6   0:00 sshd
19856 user4   9   0  1700 1392  1392 S     0.0  1.0   0:00 bash
19858 user4   9   0  2780 2488  2352 S     0.0  1.9   0:00 pine
```

Berikut merupakan sebagian dari keluaran hasil eksekusi perintah ``**top b n 1**'' pada sebuah sistem GNU/Linux yaitu "**bunga.mhs.cs.ui.ac.id**" beberapa saat yang lalu.

a. Berapakah nomer *Process Identification* dari program "*top*" tersebut?
b. Siapakah yang mengeksekusi program "*top*" tersebut?
c. Sekitar jam berapakah, program tersebut dieksekusi?
d. Sudah berapa lama sistem GNU/Linux tersebut hidup/menyala?
e. Berapa pengguna yang sedang berada pada sistem tersebut?
f. Apakah yang dimaksud dengan "*load average*"?
g. Apakah yang dimaksud dengan proses "*zombie*" ?

# (B03-2004-01) Tabel Proses III

Berikut merupakan sebagian dari keluaran hasil eksekusi perintah ``**top b n 1**'' pada sebuah sistem GNU/Linux yaitu "**rmsbase.vlsm.org**" beberapa saat yang lalu.

a. Berapakah nomor *Process Identification* dari program "*top*" tersebut?
b. Sekitar jam berapakah, program tersebut dieksekusi?
c. Apakah yang dimaksud dengan proses "*nice*" ?
d. Dalam sistem Linux, "*process*" dan "*thread*" berbagi "*process table*" yang sama. Identifikasi/tunjukkan (nomor *Process Identification*) dari salah satu *thread*. Terangkan alasannya!
e. Terangkan, mengapa sistem yang 46.6% idle dapat memiliki "*load average*" yang tinggi!

```
top - 17:31:56 up 10:14 min,  1 user,  load average: 8.64, 5.37, 2.57
Tasks:     95 total,     2 running, 93 sleeping,  0 stopped,   0 zombie
Cpu(s):  14.1% user,  35.7% system,   3.6% nice,  46.6% idle
Mem:    256712k total,   252540k used,     4172k free,    13772k buffers
Swap:   257032k total,     7024k used,   250008k free,   133132k cached

PID USER  PR NI  VIRT  RES  SHR S %CPU %MEM  TIME+  COMMAND
809 root  19 19  6780 6776 6400 S 42.2  2.6 1:02.47 rsync
709 root  20 19  6952 6952  660 R 29.3  2.7 1:46.72 rsync
710 root  19 19  6492 6484 6392 S  0.0  2.5 0:02.12 rsync
818 rms46 13  0   880  880  668 R  7.3  0.3 0:00.10 top
...
660 rms46  9  0  1220 1220  996 S  0.0  0.5 0:00.00 bash
661 rms46  9  0  1220 1220  996 S  0.0  0.5 0:00.01 bash
...
```

```
712 rms46   9   0  9256 9256 6068 S   0.0   3.6 0:06.82 evolution
781 rms46   9   0 16172  15m 7128 S   0.0   6.3 0:02.59 evolution-mail
803 rms46   9   0 16172  15m 7128 S   0.0   6.3 0:00.41 evolution-mail
804 rms46   9   0 16172  15m 7128 S   0.0   6.3 0:00.00 evolution-mail
805 rms46   9   0 16172  15m 7128 S   0.0   6.3 0:07.76 evolution-mail
806 rms46   9   0 16172  15m 7128 S   0.0   6.3 0:00.02 evolution-mail
766 rms46   9   0  5624 5624 4572 S   0.0   2.2 0:01.01 evolution-calen
771 rms46   9   0  4848 4848 3932 S   0.0   1.9 0:00.24 evolution-alarm
788 rms46   9   0  5544 5544 4516 S   0.0   2.2 0:00.55 evolution-addre
792 rms46   9   0  4608 4608 3740 S   0.0   1.8 0:01.08 evolution-execu
...
713 rms46   9   0 23580  23m  13m S   0.0   9.2 0:04.33 firefox-bin
763 rms46   9   0 23580  23m  13m S   0.0   9.2 0:00.57 firefox-bin
764 rms46   9   0 23580  23m  13m S   0.0   9.2 0:00.00 firefox-bin
796 rms46   9   0 23580  23m  13m S   0.0   9.2 0:00.18 firefox-bin
```

# (B03-2006-02) Tabel Proses IV

Berikut merupakan keluaran dari menjalankan ``**top b n 1**" pada sebuah sistem GNU/Linux yaitu "**telaga.cs.ui.ac.id**" (beberapa baris dihapus):

a. Ada berapa CPU pada sistem tersebut di atas?
b. Siapakah *user name* yang menjalankan program top tersebut?
c. Berapakah nomor *user ID* dari yang menjalankan program *top* tersebut?
d. Berapakah nomor *process ID* dari program *top* tersebut?
e. Siapakah parent dari proses *top* tersebut? Sebutkan nama program dan PID-nya!
f. Siapakah *grand parent* dari proses *top* tersebut? Sebutkan nama program dan PID-nya!
g. Gambarkan bagan "*Process Tree*" dari semua program tersebut di atas. Asumsikan, init (PID=1) merupakan root, serta *parent* dari semua proses yang tidak tercantum *parent*-nya ialah PID=1.

```
top - 11:31:54 up 40 days,  2:04,  9 users,  load average: 0.25, 0.43, 0.35
Tasks: 198 total,   1 running, 197 sleeping,   0 stopped,   0 zombie
 Cpu0 :   1.2% user,   1.4% system,  61.6% nice,  35.8% idle
 Cpu1 :   3.2% user,   0.9% system,  61.8% nice,  34.1% idle
 Cpu2 :   4.4% user,   1.1% system,  62.0% nice,  32.5% idle
 Cpu3 :   2.2% user,   0.6% system,  62.2% nice,  35.0% idle
Mem:   1032692k total,  1005108k used,    27584k free,     9776k buffers
Swap:   506008k total,   180172k used,   325836k free,   675336k cached

  PID  PPID   UID  USER   GROUP  PR NI  S  %CPU TIME COMMAND
    1     0     0  root   root    0  0  S  0.0  0:43 init
 5147  5141  1411  user2  staff   9  0  S  0.0  0:00 sshd
 5148  5147  1411  user2  staff   9  0  S  0.0  0:00 bash
 5290  5286 51018  user1  staff   9  0  S  0.0  0:00 sshd
 5291  5290 51018  user1  staff   9  0  S  0.0  0:00 bash
 5822  5148  1411  user2  staff   9  0  S  0.0  0:00 pine
 5979 24964  1030  user3  staff   9  0  S  0.0  0:00 ssh
 6207  5291 51018  user1  staff   9  0  S  0.0  0:00 mutt
 6439 24580  1248  user5  staff  13  0  R  2.7  0:00 top
23142 26022  1762  user4  staff   9  0  S  0.0  0:01 pine
24577 24575  1248  user5  staff   9  0  S  0.0  0:01 sshd
24580 24577  1248  user5  staff   9  0  S  0.0  0:00 bash
24963 24959  1030  user3  staff  10  0  S  1.3  0:01 sshd
24964 24963  1030  user3  staff   9  0  S  0.0  0:00 bash
26021 26015  1762  user4  staff   9  0  S  0.0  0:01 sshd
26022 26021  1762  user4  staff   9  0  S  0.0  0:00 bash
```