

IKI-20230 Sistem Operasi

Kumpulan Soal Ujian 2002 – 2005

© 2002-2006 Rahmat M. Samik-Ibrahim (vLSM.org)

PDF: <http://rms46.vLSM.org/1/94.pdf>

OpenOffice.org: <http://rms46.vLSM.org/1/94.odt>

Silakan menggandakan dan mengedarkan berkas ini, tanpa mengubah nota hak cipta ini.

Pasangan Konsep I (2002-2005)

Terangkan dengan singkat, pasangan konsep berikut ini. Terangkan pula perbedaan atau/dan persamaan pasangan konsep tersebut.

- a) OS View: "**Resource Allocator**" vs. "**Control Program**"
- b) "**Graceful Degradation**" — "**Fault Tolerant**".
- c) Dual Mode Operation : "**User mode**" vs. "**Monitor mode**".
- d) Operating System Goal: "**Convenient**" vs. "**Efficient**"
- e) "**System Components**" vs. "**System Calls**".
- f) "**Operating System Components**" vs. "**Operating System Services**"
- g) "**Symetric Multiprocessing**" vs. "**Asymetric Multiprocessing**"
- h) "**Distributed Systems**" vs. "**Clustered Systems**".
- i) "**Client Server System**" vs. "**Peer-to-peer system**".
- j) "**Microkernels**" vs. "**Virtual Machines**"
- k) "**Random Access Memory**" vs. "**Magnetic Disk**".
- l) "**Hard Real-time**" vs "**Soft Real-time**"
- m) Job: "**Batch system**" vs. "**Time-Sharing System**"
- n) System Design: "**Mechanism**" vs. "**Policy**"
- o) Burst Cycle: "**I/O Burst**" vs. "**CPU Burst**"
- p) Process Bound: "**I/O Bound**" vs. "**CPU Bound**"
- q) "**Process State**" vs. "**Process Control Block**".
- r) "**Waiting Time**" vs. "**Response Time**"
- s) Process Type: "**Lightweight**" vs. "**Heavyweight**"
- t) Multithread Model: "**One to One**" vs. "**Many to Many**"
- u) Scheduling Process: "**Short Term**" vs. "**Long Term**"
- v) Scheduling Algorithm: "**FCFS (First Come First Serve)**" vs. "**SJF (Shortest Job First)**"
- w) "**Preemptive Shortest Job First**" vs. "**Non-preemptive Shortest Job First**".
- x) Inter Process Communication: "**Direct Communication**" vs. "**Indirect Communication**"
- y) "**Critical Section**" vs. "**Race Condition**".
- z) Process Synchronization: "**Monitor**" vs. "**Semaphore**"
- aa) "**Deadlock Avoidance**" vs. "**Deadlock Detection**".
- ab) "**Deadlock**" vs. "**Starvation**".
- ac) Address Space: "**Logical**" vs. "**Physical**"
- ad) Dynamic Storage Allocation Strategy: "**Best Fit**" vs. "**Worse Fit**"
- ae) Virtual Memory Allocation Strategy: "**Global**" vs. "**Local Replacement**"
- af) File Operations: "**Deleting**" vs. "**Truncating**"
- ag) Storage System: "**Volatile**" vs. "**Non-volatile**"
- ah) File Allocation Methods: "**Contiguous**" vs. "**Linked**"
- ai) Disk Management: "**Boot Block**" vs. "**Bad Block**"
- aj) I/O Data-Transfer Mode: "**Character**" vs. "**Block**"
- ak) I/O Access Mode: "**Sequential**" vs. "**Random**"
- al) I/O Transfer Schedules: "**Synchronous**" vs. "**Asynchronous**"
- am) I/O Sharing: "**Dedicated**" vs. "**Sharable**"

Pasangan Konsep II (2002-2005)

Terangkan dengan singkat, pasangan konsep berikut ini. Terangkan pula perbedaan atau/dan persamaan pasangan konsep tersebut.

- a) I/O direction: "**Read only**" vs. "**Write only**"
- b) "**I/O Structure**" vs. "**Storage Structure**"
- c) **Software License**: "**Free Software**" vs. "**Copyleft**"

Konsep Sistem Operasi (2005)

- a) Terangkan/jabarkan sekurangnya empat komponen utama dari sebuah Sistem Operasi.
- b) Terangkan/jabarkan peranan/pengaruh dari keempat komponen di atas terhadap sebuah Sistem Operasi Waktu Nyata (*Real Time System*).
- c) Terangkan/jabarkan peranan/pengaruh dari keempat komponen di atas terhadap sebuah Sistem Prosesor Jamak (*Multi Processors System*).
- d) Terangkan/jabarkan peranan/pengaruh dari keempat komponen di atas terhadap sebuah Sistem Operasi Terdistribusi (*Distributed System*).
- e) Terangkan/jabarkan peranan/pengaruh dari keempat komponen di atas terhadap sebuah Sistem Operasi Telepon Seluler (*Cellular Phone*).

Tabel Proses I (2003)

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.

```
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
3	root	19	19	0	0	0	SWN	0.0	0.0	0:02	ksoftirqd_CPU0
.....											
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

(sambungan Tabel Proses)

- 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*" ?

Tabel Proses II (2004)

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.

```
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
```

- 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!

GNU/Linux (2003)

- Sebutkan perbedaan utama antara kernel linux versi 1.X dan versi 2.X !
- Terangkan, apa yang disebut dengan "Distribusi (distro) Linux"? Berikan empat contoh distro!
- Berikut merupakan sebagian dari keluaran menjalankan perintah "**top b n 1**" pada server "bunga.mhs.cs.ui.ac.id" pada tanggal 10 Juni 2003 yang lalu.
- Jam berapakah program tersebut di atas dijalankan?
- Berapa waktu yang lalu (perkirakan/hitung dari tanggal 10 Juni tersebut), server "bunga.mhs.cs.ui.ac.id" terakhir kali (re)boot?
- Apakah yang dimaksud dengan "*load average*" ?
- Sebutkan nama dari sebuah proses di atas yang statusnya "running"!
- Sebutkan nama dari sebuah proses di atas yang statusnya "waiting"!

```
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

Kernel Linux 2.6.X (=KL26) (2004)

- Terangkan, apa yang dimaksud dengan Perangkat Lunak Bebas (PLB) yang berbasis lisensi GNU GPL (*General Public Licence*)!
- KL26 diluncurkan Desember 2003. Terangkan mengapa hingga kini (Januari 2005), belum juga dibuka cabang pengembangan Kernel Linux versi 2.7.X!
- KL26 lebih mendukung sistem berskala kecil seperti Mesin Cuci, Kamera, Ponsel, mau pun PDA. Terangkan, bagaimana kemampuan (*feature*) opsi tanpa MMU (*Memory Management Unit*) dapat mendukung sistem berskala kecil.
- KL26 lebih mendukung sistem berskala sangat besar seperti "*Enterprise System*". Terangkan sekurangnya dua kemampuan (*feature*) agar dapat mendukung sistem berskala sangat besar.
- KL26 lebih mendukung sistem interaktif seperti "*Work Station*". Terangkan sekurangnya satu kemampuan (*feature*) agar dapat mendukung sistem interaktif.

Rancangan Sistem (2002)

Rancang sebuah sistem yang secara *rata-rata*:

- sanggup melayani secara bersamaan (*concurrent*) hingga 1000 pengguna (*users*).
 - hanya 1% dari pengguna yang aktif mengetik pada suatu saat, sedangkan sisanya (99%) tidak mengerjakan apa-apa (*idle*).
 - kecepatan mengetik 10 karakter per detik.
 - setiap ketukan (ketik) menghasilkan "**response**" *CPU burst* dengan ukuran 10000 instruksi mesin.
 - setiap instruksi mesin dijalankan dalam 2 (dua) buah siklus mesin (*machine cycle*).
 - utilisasi CPU 100%.
- a) Gambarkan GANTT chart dari proses-proses tersebut di atas. Lengkapi gambar dengan yang dimaksud dengan *burst time* dan *response time*!
 - b) Berapa lama, durasi sebuah *CPU burst* tersebut?
 - c) Berapa lama, kasus terbaik (*best case*) *response time* dari ketikan tersebut?
 - d) Berapa lama, kasus terburuk (*worse case*) *response time* dari ketikan tersebut?
 - e) Berapa MHz. *clock-rate* CPU pada kasus butir tersebut di atas?

Penjadualan Proses I (2004)

Diketahui tiga (3) proses *preemptive* dengan nama berturut-turut $P_1(0)$, $P_2(2)$, dan $P_3(4)$. Angka dalam kurung menunjukkan waktu tiba ("*arrival time*"). Ketiga proses tersebut memiliki *burst time* yang sama yaitu 4 satuan waktu (*unit time*). Setiap memulai/peralihan proses, selalu diperlukan waktu-alih (*switch time*) sebesar satu (1) satuan waktu.

Berapakah rata-rata *turn-around time* dan *waiting time* dari ketiga proses tersebut, jika diimplementasikan dengan algoritma penjadualan:

- **Shortest Waiting First**: mendahulukan proses dengan *waiting time* terendah.
- **Longest Waiting First**: mendahulukan proses dengan *waiting time* tertinggi.

Jika kriteria penjadualan seri, dahulukan proses dengan nomor urut yang lebih kecil (umpama: P_1 akan didahulukan dari P_2). Jangan lupa membuat *Gantt Chart*-nya!

Penjadualan Proses II (2002)

Lima proses tiba secara bersamaan pada saat " t_0 " (awal) dengan urutan P_1 , P_2 , P_3 , P_4 , dan P_5 . Bandingkan (rata-rata) *turn-around time* dan *waiting time* dari ke lima proses tersebut di atas; jika mengimplementasikan algoritma penjadualan seperti FCFS, SJF, dan RR (Round Robin) dengan kuantum 2 (dua) satuan waktu. *Context switch* diabaikan.

- a) Burst time kelima proses tersebut berturut-turut (10, 8, 6, 4, 2) satuan waktu.
- b) Burst time kelima proses tersebut berturut-turut (2, 4, 6, 8, 10) satuan waktu.

Penjadualan Proses III (2001)

Diketahui lima (5) PROSES dengan nama berturut-turut:

- $P_1(0, 9)$
- $P_2(2, 7)$
- $P_3(4, 1)$
- $P_4(6, 3)$
- $P_5(8, 2)$

Angka dalam kurung menunjukkan: ("*arrival time*", "*burst time*"). Setiap peralihan proses, selalu akan diperlukan waktu-alih (*switch time*) sebesar satu (1) satuan waktu (*unit time*).

(Penjadualan Proses III)

- a) Berapakah rata-rata **turnaround time** dan **waiting time** dari kelima proses tersebut, jika diimplementasikan dengan algoritma penjadualan FCFS (First Come, First Serve)?
- b) Bandingkan **turnaround time** dan **waiting time** tersebut, dengan sebuah algoritma penjadualan dengan ketentuan sebagai berikut:
 - **Pre-emptive**: pergantian proses dapat dilakukan kapan saja, jika ada proses lain yang memenuhi syarat. Namun durasi setiap proses dijamin minimum dua (2) satuan waktu, sebelum boleh diganti.
 - Waktu alih (**switch-time**) sama dengan diatas, yaitu sebesar satu (1) satuan waktu (unit time).
 - Jika proses telah menunggu ≥ 15 satuan waktu:
 - dahulukan proses yang telah menunggu paling lama
 - lainnya:
 - dahulukan proses yang menunggu paling sebentar.
 - Jika kriteria yang terjadi seri:
 - dahulukan proses dengan nomor urut yang lebih kecil (umpama: P1 akan didahulukan dari P2).

Status Proses (2003)

- a) Gambarkan sebuah model bagan status proses (process state diagram) dengan minimum lima (5) status.
- b) Sebutkan serta terangkan semua nama status proses (process states) tersebut.
- c) Sebutkan serta terangkan semua nama kejadian (event) yang menyebabkan perubahan status proses.
- d) Terangkan perbedaan antara proses "**I/O Bound**" dengan proses "**CPU Bound**" berdasarkan bagan status proses tersebut.

Proses (2005)

Silakan menelusuri program C berikut ini. Diasumsikan bahwa PID dari program tersebut (baris 17) ialah 5000, serta tidak ada proses lain yang terbentuk kecuali dari "fork()" program ini.

- a) Tuliskan keluaran dari program tersebut.
- b) Ubahlah **MAXLEVEL** (baris 04) menjadi "5"; lalu kompail ulang dan jalankan kembali! Tuliskan bagian keluaran dari modifikasi program tersebut.
- c) Jelaskan asumsi pemilihan PID pada butir "b" di atas!

```
01 #include <sys/types.h>
02 #include <stdio.h>
03 #include <unistd.h>
04 #define MAXLEVEL 4
05 char* turunan[] = {"", "pertama", "kedua", "ketiga", "keempat", "kelima"};
06
07
08 main()
09 {
10     int     idx     = 1;
11     int     putaran = 0;
12     int     deret0  = 0;
13     int     deret1  = 1;
14     int     tmp;
15     pid_t  pid;
16
```

```

17 printf("PID INDUK %d\n", (int) getpid());
18 printf("START deret Fibonacci... %d... %d...\n", deret0, deret1);
20 while (putaran < MAXLEVEL)
21 {
22     tmp=deret0+deret1;
23     deret0=deret1;
24     deret1=tmp;
25
26     pid = fork();           /* FORK      */
27
28     if (pid > 0)           /* Induk?  */
29     {
30         wait(NULL);
31         printf("INDUK %s selesai menunggu ", turunan[idx]);
32         printf("PID %d...\n", (int) pid);
33         putaran++;
34     } else if (pid==0) {   /* Turunan? */
35         printf("Deret Fibonacci selanjutnya... %d...\n", deret1);
36         idx++;
37         exit (0);
38     } else {              /* Error?   */
39         printf("Error...\n");
40         exit (1);
41     }
42 };
43 exit (0);
44 }

```

Deadlock I (2005)

- Terangkan/jabarkan secara singkat, keempat kondisi yang harus dipenuhi agar terjadi *Deadlock*! Gunakan graf untuk menggambarkan keempat kondisi tersebut!
- Terangkan/jabarkan secara singkat, apakah akan selalu terjadi *Deadlock* jika keempat kondisi tersebut dipenuhi?!

Deadlock II (2003)

Diketahui:

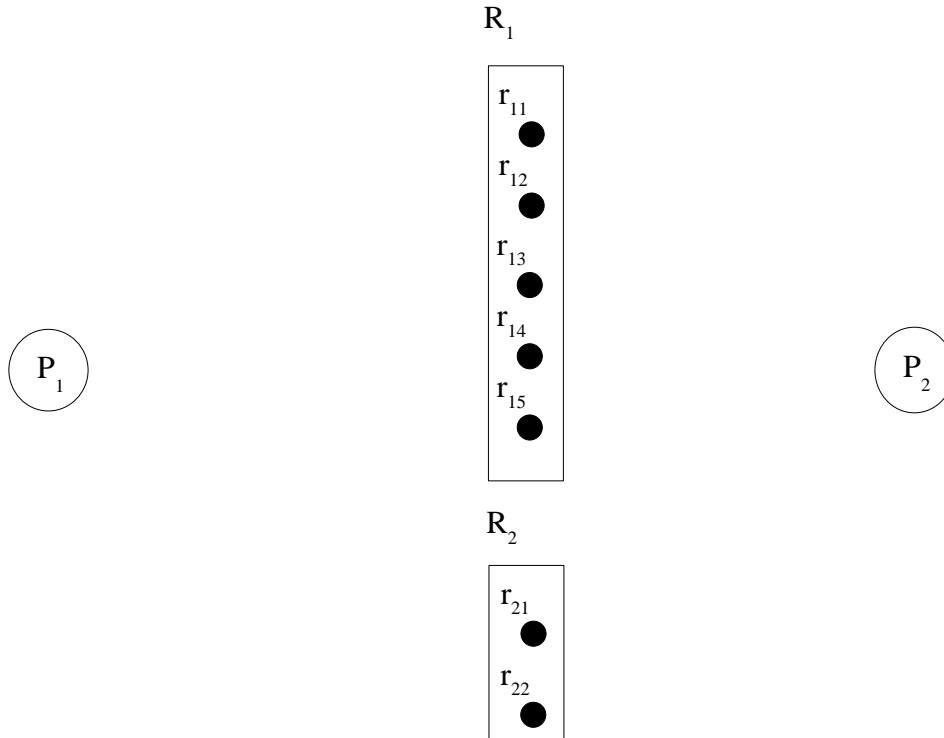
- set P yang terdiri dari dua (2) proses; $P = \{ P_1, P_2 \}$.
- set R yang terdiri dari dua (2) sumber-daya (*resources*); dengan berturut-turut lima (5) dan dua (2) *instances*; $R = \{ R_1, R_2 \} = \{ \{r_{11}, r_{12}, r_{13}, r_{14}, r_{15} \}, \{r_{21}, r_{22} \} \}$.
- Plafon (jatah maksimum) sumber-daya untuk masing-masing proses ialah:

	R ₁	R ₂
P ₁	5	1
P ₂	3	1

- Pencegahan *deadlock* dilakukan dengan **Banker's Algorithm**.
- Alokasi sumber-daya yang **memenuhi** kriteria *Banker's Algorithm* di atas, akan diprioritaskan pada proses dengan indeks yang lebih kecil.
- Setelah mendapatkan semua sumber-daya yang diminta, proses akan mengembalikan **SELURUH** sumber-daya tersebut.
- Pada saat T₀, "**Teralokasi**" serta "**Permintaan**" sumber-daya proses ditentukan sebagai berikut:

	TERALOKASI		PERMINTAAN	
	R ₁	R ₂	R ₁	R ₂
P ₁	2	0	2	1
P ₂	2	0	1	1

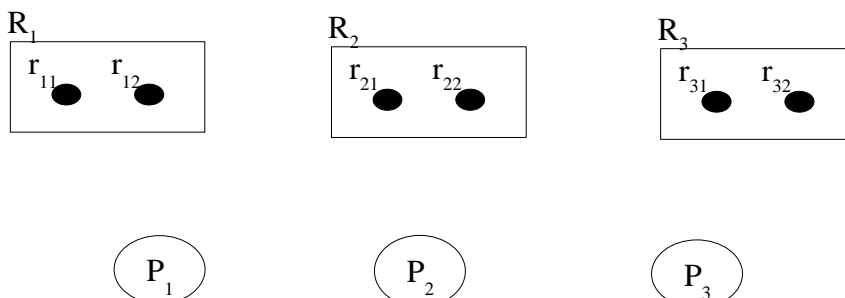
Gambarkan *graph* pada urutan T₀, T₁,... dan seterusnya, hingga semua permintaan sumber-daya terpenuhi dan dikembalikan. Sebutkan, jika terjadi kondisi "*unsafe*"!



Deadlock III (2003)

Diketahui:

- set P yang terdiri dari tiga (3) proses; $P = \{ P_1, P_2, P_3 \}$.
- set R yang terdiri dari tiga (3) *resources*; masing-masing terdiri dari dua (2) *instances*; $R = \{ R_1, R_2, R_3 \} = \{ \{r_{11}, r_{12}\}, \{r_{21}, r_{22}\}, \{r_{31}, r_{32}\} \}$.
- Prioritas alokasi sumber daya (*resource*) akan diberikan pada proses dengan indeks yang lebih kecil.
- Jika tersedia:** permintaan alokasi sumber daya pada T_N akan dipenuhi pada urutan berikutnya (T_{N+1}).
- Proses yang telah dipenuhi semua permintaan sumber daya (*resources*) pada T_M; akan melepaskan semua sumber daya tersebut pada urutan berikutnya (T_{M+1}).
- Pencegahan *deadlock* dilakukan dengan menghindari *circular wait*.
- Pada saat T₀, set E₀ = { } (atau kosong), sehingga gambar *graph*-nya sebagai berikut:



Jika set E pada saat T_1 menjadi:

$E_1 = \{ P_1 \rightarrow R_1, P_1 \rightarrow R_2, P_2 \rightarrow R_1, P_2 \rightarrow R_2, P_3 \rightarrow R_1, P_3 \rightarrow R_2, P_3 \rightarrow R_3 \}$,

gambaran *graph* pada urutan T_1, T_2, \dots serta (E_2, E_3, \dots) berikutnya hingga semua permintaan sumberdaya terpenuhi dan dikembalikan.

Problem Reader/Writer I (2001)

Perhatikan berkas "ReaderWriterServer.java" berikut ini (*source-code* terlampir):

- Ada berapa *object class* "Reader" yang terbentuk? Sebutkan nama-namanya!
- Ada berapa *object class* "Writer" yang terbentuk? Sebutkan nama-namanya!
- Modifikasi kode program tersebut (cukup baris terkait), sehingga akan terdapat 6 (enam) "Reader" dan 4 (empat) "Writer".
- Modifikasi kode program tersebut, dengan menambahkan sebuah (satu!) *object thread* baru yaitu "*janitor*". Sang "*janitor*" berfungsi untuk membersihkan (*cleaning*). Setelah membersihkan, "*janitor*" akan tidur (*sleeping*). Pada saat bangun, "*janitor*" kembali akan membersihkan. Dan seterusnya... Pada saat "*janitor*" akan membersihkan, tidak boleh ada "*reader*" atau "*writer*" yang aktif. Jika ada, "*janitor*" harus menunggu. Demikian pula, "*reader*" atau "*writer*" harus menunggu "*janitor*" hingga selesai membersihkan.

Problem Reader/Writer II (2002)

Perhatikan berkas "ReaderWriterServer.java" berikut ini, yang merupakan gabungan "ReaderWriterServer.java", "Reader.java", "Writer.java", "Semaphore.java", "Database.java", oleh Gagne, Galvin, dan Silberschatz. Terangkan berdasarkan berkas tersebut:

- akan terbentuk berapa *thread*, jika menjalankan program class "ReaderWriterServer" ini? Apa yang membedakan antara sebuah *thread* dengan *thread* lainnya?
- mengapa: jika ada "*Reader*" yang sedang membaca, tidak ada "*Writer*" yang dapat menulis; dan mengapa: jika ada "*Writer*" yang sedang menulis, tidak ada "*Reader*" yang dapat membaca?
- mengapa: jika ada "*Reader*" yang sedang membaca, boleh ada "*Reader*" lainnya yang turut membaca?
- modifikasi kode program tersebut (cukup mengubah baris terkait), sehingga akan terdapat 5 (lima) "*Reader*" dan 4 (empat) "*Writer*"!

Modifikasi kode program tersebut (cukup mengubah *method* terkait), sehingga pada saat RAJA (*Reader* 0) ingin membaca, tidak boleh ada RAKYAT (*Reader* lainnya) yang sedang/akan membaca. **JANGAN MEMPERSULIT DIRI SENDIRI:** jika RAJA sedang membaca, RAKYAT boleh turut membaca.

```
001 // Gabungan ReaderWriterServer.java Reader.java Writer.java
002 //           Semaphore.java Database.java
003 // (c) 2000 Gagne, Galvin, Silberschatz
004
005 public class ReaderWriterServer {
006     public static void main(String args[]) {
007         Database server = new Database();
008         Reader[] readerArray = new Reader[NUM_OF_READERS];
009         Writer[] writerArray = new Writer[NUM_OF_WRITERS];
010         for (int i = 0; i < NUM_OF_READERS; i++) {
011             readerArray[i] = new Reader(i, server);
012             readerArray[i].start();
013         }
014         for (int i = 0; i < NUM_OF_WRITERS; i++) {
015             writerArray[i] = new Writer(i, server);
016             writerArray[i].start();
017         }
018     }
```

```

019     private static final int NUM_OF_READERS = 3;
020     private static final int NUM_OF_WRITERS = 2;
021 }
022
023 class Reader extends Thread {
024     public Reader(int r, Database db) {
025         readerNum = r;
026         server = db;
027     }
028     public void run() {
029         int c;
030         while (true) {
031             Database.napping();
032             System.out.println("reader " + readerNum + " wants to read.");
033             c = server.startRead();
034             System.out.println("reader " + readerNum +
035                 " is reading. Reader Count = " + c);
036             Database.napping();
037             System.out.print("reader " + readerNum + " is done reading. ");
038             c = server.endRead();
039         }
040     }
041     private Database server;
042     private int readerNum;
043 }
044
045 class Writer extends Thread {
046     public Writer(int w, Database db) {
047         writerNum = w;
048         server = db;
049     }
050     public void run() {
051         while (true) {
052             System.out.println("writer " + writerNum + " is sleeping.");
053             Database.napping();
054             System.out.println("writer " + writerNum + " wants to write.");
055             server.startWrite();
056             System.out.println("writer " + writerNum + " is writing.");
057             Database.napping();
058             System.out.println("writer " + writerNum + " is done writing.");
059             server.endWrite();
060         }
061     }
062     private Database server;
063     private int writerNum;
064 }
065
066 final class Semaphore {
067     public Semaphore() {
068         value = 0;
069     }
070     public Semaphore(int v) {
071         value = v;
072     }
073     public synchronized void P() {
074         while (value <= 0) {
075             try { wait(); }
076             catch (InterruptedException e) { }
077         }
078         value--;
079     }

```

```

080     public synchronized void V() {
081         ++value;
082         notify();
083     }
084     private int value;
085 }
086
087 class Database {
088     public Database() {
089         readerCount = 0;
090         mutex = new Semaphore(1);
091         db = new Semaphore(1);
092     }
093     public static void napping() {
094         int sleepTime = (int) (NAP_TIME * Math.random() );
095         try { Thread.sleep(sleepTime*1000); }
096         catch(InterruptedException e) {}
097     }
098     public int startRead() {
099         mutex.P();
100         ++readerCount;
101         if (readerCount == 1) {
102             db.P();
103         }
104         mutex.V();
105         return readerCount;
106     }
107     public int endRead() {
108         mutex.P();
109         --readerCount;
110         if (readerCount == 0) {
111             db.V();;
112         }
113         mutex.V();
114         System.out.println("Reader count = " + readerCount);
115         return readerCount;
116     }
117     public void startWrite() {
118         db.P();
119     }
120     public void endWrite() {
121         db.V();
122     }
123     private int readerCount;
124     Semaphore mutex;
125     Semaphore db;
126     private static final int NAP_TIME = 15;
127 }
128
129 // The Class java.lang.Thread
130 // When a thread is created, it is not yet active; it begins to run when method
131 // "start" is called. Invoking the "start" method causes this thread to begin
132 // execution; by calling the "run" method.
133 // public class Thread implements Runnable {
134 //     ...
135 //     public void run();
136 //     public void start()
137 //         throws InterruptedException;
138 //     ...
139 // }

```

Problem Reader/Writer III (2004)

Perhatikan berkas program java pada halaman berikut ini.

- Berapa jumlah *thread class* Reader yang akan terbentuk?
- Berapa jumlah *thread class* Writer yang akan terbentuk?
- Perkirakan bagaimana bentuk keluaran (*output*) dari program tersebut!
- Modifikasi program agar "*nap*" rata-rata dari class Reader lebih besar daripada class Writer.

```
001 /*****
002 * Gabungan/Modif: Factory.java Database.java RWLock.java Reader.java
003 * Semaphore.java SleepUtilities.java Writer.java
004 * Operating System Concepts with Java - Sixth Edition
005 * Gagne, Galvin, Silberschatz Copyright John Wiley & Sons - 2003.
006 */
007
008 public class Factory
009 {
010     public static void main(String args[])
011     {
012         System.out.println("INIT Thread...");
013         Database server = new Database();
014         Thread readerX = new Thread(new Reader(server));
015         Thread writerX = new Thread(new Writer(server));
016         readerX.start();
017         writerX.start();
018         System.out.println("Wait...");
019     }
020 }
022 // Reader // ****
023 class Reader implements Runnable
024 {
025     public Reader(Database db) { server = db; }
026
027     public void run() {
028         while (--readercounter > 0)
029         {
030             SleepUtilities.nap();
031             System.out.println("readerX: wants to read.");
032             server.acquireReadLock();
033             System.out.println("readerX: is reading.");
034             SleepUtilities.nap();
035             server.releaseReadLock();
036             System.out.println("readerX: done...");
037         }
038     }
039
040     private Database server;
041     private int readercounter = 3;
042 }
043
044 // Writer // ****
045 class Writer implements Runnable
046 {
047     public Writer(Database db) { server = db; }
048
049     public void run() {
050         while (writercounter-- > 0)
051         {
052             SleepUtilities.nap();
053             System.out.println("writerX: wants to write.");
054             server.acquireWriteLock();
055             System.out.println("writerX: is writing.");
```

```

056         SleepUtilities.nap();
057         server.releaseWriteLock();
058         System.out.println("writerX: done...");
059     }
060 }
061
062 private Database server;
063 private int     writercounter = 3;
064 }
065 // Semaphore // *****
066 class Semaphore
067 {
068     public Semaphore()          { value = 0; }
069     public Semaphore(int val) { value = val; }
070     public synchronized void acquire() {
071         while (value == 0) {
072             try { wait(); }
073             catch (InterruptedException e) { }
074         }
075         value--;
076     }
077
078     public synchronized void release() {
079         ++value;
080         notifyAll();
081     }
082     private int value;
083 }
084
085 // SleepUtilities // *****
086 class SleepUtilities
087 {
088     public static void nap() { nap(NAP_TIME); }
089
090     public static void nap(int duration) {
091         int sleeptime = (int) (duration * Math.random() );
092         try { Thread.sleep(sleeptime*1000); }
093         catch (InterruptedException e) {}
094     }
095     private static final int NAP_TIME = 3;
096 }
097
098 // Database // *****
099 class Database implements RWLock
100 {
101     public Database()          { db = new Semaphore(1); }
102     public void acquireReadLock() { db.acquire(); }
103     public void releaseReadLock() { db.release(); }
104     public void acquireWriteLock() { db.acquire(); }
105     public void releaseWriteLock() { db.release(); }
106     Semaphore db;
107 }
108 // An interface for reader-writer locks. // *****
109 interface RWLock
110 {
111     public abstract void acquireReadLock();
112     public abstract void releaseReadLock();
113     public abstract void acquireWriteLock();
114     public abstract void releaseWriteLock();
115 }

```

Bounded Buffer (2003)

Perhatikan berkas "**BoundedBufferServer.java**" pada halaman berikut:

- a) Berapakah ukuran penyangga (*buffer*) ?
- b) Modifikasi program (sebutkan nomor barisnya) agar ukuran penyangga (*buffer*) menjadi 6.
- c) Tuliskan/perkirakan keluaran (*output*) 10 baris pertama, jika menjalankan program ini.
- d) Jelaskan fungsi dari ketiga *semaphore* (mutex, full, empty) pada program tersebut.
- e) Tambahkan (sebutkan nomor barisnya) sebuah thread dari class **Supervisor** yang berfungsi:
 - i. pada awal dijalankan, melaporkan ukuran penyangga (*buffer*).
 - ii. secara berkala (acak), melaporkan jumlah pesan (*message*) yang berada dalam penyangga (*buffer*).
- f) *Semaphore* mana yang paling relevan untuk modifikasi butir "e" di atas?

```
001 // Authors: Greg Gagne, Peter Galvin, Avi Silberschatz
002 // Slightly Modified by: Rahmat M. Samik-Ibrahim
003 // Copyright (c) 2000 by Greg Gagne, Peter Galvin, Avi Silberschatz
004 // Applied Operating Systems Concepts - John Wiley and Sons, Inc.
005 //
006 // Class "Date":
007 //     Allocates a Date object and initializes it so that it represents
008 //     the time at which it was allocated,
009 //     (E.g.): "Wed Apr 09 11:12:34 JAVT 2003"
010 // Class "Object"/ method "notify":
011 //     Wakes up a single thread that is waiting on this object's monitor.
012 // Class "Thread"/ method "start":
013 //     Begins the thread execution and calls the run method of the thread.
014 // Class "Thread"/ method "run":
015 //     The Runnable object's run method is called.
016
017 import java.util.*;
018 // main *****
019 public class BoundedBufferServer
020 {
021     public static void main(String args[])
022     {
023         BoundedBuffer server          = new BoundedBuffer();
024         Producer      producerThread = new Producer(server);
025         Consumer      consumerThread = new Consumer(server);
026         producerThread.start();
027         consumerThread.start();
028     }
029 }
030
031 // Producer *****
032 class Producer extends Thread
033 {
034     public Producer(BoundedBuffer b)
035     {
036         buffer = b;
037     }
038 }
```

```

039     public void run()
040     {
041         Date message;
042         while (true)
043         {
044             BoundedBuffer.napping();
045
046             message = new Date();
047             System.out.println("P: PRODUCE " + message);
048             buffer.enter(message);
049         }
050     }
051     private BoundedBuffer buffer;
052 }
053
054 // Consumer *****
055 class Consumer extends Thread
056 {
057     public Consumer(BoundedBuffer b)
058     {
059         buffer = b;
060     }
061     public void run()
062     {
063         Date message;
064         while (true)
065         {
066             BoundedBuffer.napping();
067             System.out.println("C: CONSUME START");
068             message = (Date)buffer.remove();
069         }
070     }
071     private BoundedBuffer buffer;
072 }
073
074 // BoundedBuffer.java *****
075 class BoundedBuffer
076 {
077     public BoundedBuffer()
078     {
079         count = 0;
080         in = 0;
081         out = 0;
082         buffer = new Object[BUFFER_SIZE];
083         mutex = new Semaphore(1);
084         empty = new Semaphore(BUFFER_SIZE);
085         full = new Semaphore(0);
086     }
087     public static void napping()
088     {
089         int sleepTime = (int) (NAP_TIME * Math.random());
090         try { Thread.sleep(sleepTime*1000); }
091         catch(InterruptedException e) { }
092     }
093     public void enter(Object item)
094     {
095         empty.P();
096         mutex.P();
097         ++count;
098         buffer[in] = item;
099         in = (in + 1) % BUFFER_SIZE;
100         System.out.println("P: ENTER " + item);
101         mutex.V();
102         full.V();
103     }

```

```

104 public Object remove ()
105 {
106     Object item;
107     full.P();
108     mutex.P();
109     --count;
110     item = buffer[out];
111     out = (out + 1) % BUFFER_SIZE;
112     System.out.println("C: CONSUMED " + item);
113     mutex.V();
114     empty.V();
115     return item;
116 }
117 public static final int NAP_TIME = 5;
118 private static final int BUFFER_SIZE = 3;
119 private Semaphore mutex;
120 private Semaphore empty;
121 private Semaphore full;
122 private int count, in, out;
123 private Object[] buffer;
124 }
125
126 // Semaphore.java *****
127
128 final class Semaphore
129 {
130     public Semaphore ()
131     {
132         value = 0;
133     }
134     public Semaphore(int v)
135     {
136         value = v;
137     }
138     public synchronized void P ()
139     {
140         while (value <= 0)
141         {
142             try { wait(); }
143             catch (InterruptedException e) { }
144         }
145         value --;
146     }
147     public synchronized void V ()
148     {
149         ++value;
150         notify();
151     }
152     private int value;
153 }

```

Sinkronisasi (2005)

- Terangkan peranan/fungsi dari semafor-semafor pada program Java berikut ini!
- Tuliskan keluaran dari program tersebut!
- Modifikasi program (baris mana?), agar object “proses” dengan index tinggi mendapat prioritas diutamakan dibandingkan “proses” dengan index rendah.
- Terangkan kelemahan dari program ini! Kondisi bagaimana yang mengakibatkan semafor tidak berperan seperti yang diinginkan!

```

0 /*****
1 * SuperProses (c) 2005 Rahmat M. Samik-Ibrahim, GPL-like */
2
3 // ***** SuperProses *
4 public class SuperProses {
5     public static void main(String args[]) {
6         Semafor[] semafor1 = new Semafor[JUMLAH_PROSES];
7         Semafor[] semafor2 = new Semafor[JUMLAH_PROSES];
8         for (int ii = 0; ii < JUMLAH_PROSES; ii++) {
9             semafor1[ii] = new Semafor();
10            semafor2[ii] = new Semafor();
11        }
12
13        Thread superp=new Thread(new SuperP(semafor1,semafor2,JUMLAH_PROSES));
14        superp.start();
15
16        Thread[] proses= new Thread[JUMLAH_PROSES];
17        for (int ii = 0; ii < JUMLAH_PROSES; ii++) {
18            proses[ii]=new Thread(new Proses(semafor1,semafor2,ii));
19            proses[ii].start();
20        }
21    }
22
23    private static final int JUMLAH_PROSES = 5;
24 }
25
26 // ** SuperP *****
27 class SuperP implements Runnable {
28     SuperP(Semafor[] sem1, Semafor[] sem2, int jmlh) {
29         semafor1 = sem1;
30         semafor2 = sem2;
31         jumlah_proses = jmlh;
32     }
33
34     public void run() {
35         for (int ii = 0; ii < jumlah_proses; ii++) {
36             semafor1[ii].kunci();
37         }
38         System.out.println("SUPER PROSES siap...");
39         for (int ii = 0; ii < jumlah_proses; ii++) {
40             semafor2[ii].buka();
41             semafor1[ii].kunci();
42         }
43     }
44
45     private Semafor[] semafor1, semafor2;
46     private int jumlah_proses;
47 }// ** Proses *****
48
49 class Proses implements Runnable {
50     Proses(Semafor[] sem1, Semafor[] sem2, int num) {
51         num_proses = num;
52         semafor1 = sem1;
53         semafor2 = sem2;
54     }
55
56     public void run() {
57         semafor1[num_proses].buka();
58         semafor2[num_proses].kunci();
59         System.out.println("Proses " + num_proses + " siap...");
60         semafor1[num_proses].buka();
61     }
62
63     private Semafor[] semafor1, semafor2;
64     private int num_proses;
65 }
66 }

```

```

68 // ** Semafor *
69 class Semafor {
70     public Semafor()          { value = 0;   }
71     public Semafor(int val) { value = val; }
72
73     public synchronized void kunci() {
74         while (value == 0) {
75             try { wait(); }
76             catch (InterruptedException e) { }
77         }
78         value--;
79     }
80
81     public synchronized void buka() {
82         value++;
83         notify();
84     }
85
86     private int value;
87 }

```

IPC (2003)

Perhatikan berkas program java berikut ini:

```

001 /* Gabungan Berkas:
002 * FirstSemaphore.java, Runner.java, Semaphore.java, Worker.java.
003 * Copyright (c) 2000 oleh Greg Gagne, Peter Galvin, Avi Silberschatz.
004 * Applied Operating Systems Concepts - John Wiley and Sons, Inc.
005 * Slightly modified by Rahmat M. Samik-Ibrahim.
006 *
007 * Informasi Singkat (RMS46):
008 * Threat.start() --> memulai thread yang akan memanggil Threat.run().
009 * Threat.sleep(xxx) --> thread akan tidur selama xxx milidetik.
010 * try {...} catch (InterruptedException e) {} --> sarana terminasi program.
011 */
012
013 public class FirstSemaphore
014 {
015     public static void main(String args[]) {
016         Semaphore sem = new Semaphore(1);
017         Worker[] bees = new Worker[NN];
018         for (int ii = 0; ii < NN; ii++)
019             bees[ii] = new Worker(sem, ii);
020         for (int ii = 0; ii < NN; ii++)
021             bees[ii].start();
022     }
023     private final static int NN=4;
024 }
025
026 // Worker =====
027 class Worker extends Thread
028 {
029     public Worker(Semaphore sss, int nnn) {
030         sem = sss;
031         wnumber = nnn;
032         wstring = WORKER + (new Integer(nnn)).toString();
033     }
034 }

```

```

035     public void run() {
036         while (true) {
037             System.out.println(wstring + PESAN1);
038             sem.P();
039             System.out.println(wstring + PESAN2);
040             Runner.criticalSection();
041             System.out.println(wstring + PESAN3);
042             sem.V();
043             Runner.nonCriticalSection();
044         }
045     }
046     private Semaphore sem;
047     private String    wstring;
048     private int      wnumber;
049     private final static String PESAN1=" akan masuk ke Critical Section.";
050     private final static String PESAN2=" berada di dalam Critical Section.";
051     private final static String PESAN3=" telah keluar dari Critical Section.";
052     private final static String WORKER="PEKERJA ";
053 }
054
055 // Runner =====
056 class Runner
057 {
058     public static void criticalSection() {
059         try {
060             Thread.sleep( (int) (Math.random() * CS_TIME * 1000) );
061         }
062         catch (InterruptedException e) { }
063     }
064
065     public static void nonCriticalSection() {
066         try {
067             Thread.sleep( (int) (Math.random() * NON_CS_TIME * 1000) );
068         }
069         catch (InterruptedException e) { }
070     }
071     private final static int    CS_TIME = 2;
072     private final static int NON_CS_TIME = 2;
073 }
074
075 // Semaphore =====
076 final class Semaphore
077 {
078     public Semaphore() {
079         value = 0;
080     }
081
082     public Semaphore(int v) {
083         value = v;
084     }
085
086     public synchronized void P() {
087         while (value <= 0) {
088             try {
089                 wait();
090             }
091             catch (InterruptedException e) { }
092         }
093         value --;
094     }

```

```

095
096     public synchronized void V() {
097         ++value;
098         notify();
099     }
100
101     private int value;
102 }
103
104 // END =====

```

- a) Berapakah jumlah *object* dari "*Worker Class*" yang akan terbentuk?
- b) Sebutkan nama-nama *object* dari "*Worker Class*" tersebut!
- c) Tuliskan/perkirakan keluaran (*output*) 10 baris pertama, jika menjalankan program ini!
- d) Apakah keluaran pada butir "c" di atas akan berubah, jika parameter CS_TIME diubah menjadi dua kali NON_CS_TIME? Terangkan!
- e) Apakah keluaran pada butir "c" di atas akan berubah, jika selain parameter CS_TIME diubah menjadi dua kali NON_CS_TIME, dilakukan modifikasi NN menjadi 10? Terangkan!

Status Memori (2004)

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.

```

top - 10:59:25 up 3:11, 1 user, load average: 9.18, 9.01, 7.02
Tasks: 122 total, 3 running, 119 sleeping, 0 stopped, 0 zombie
Cpu(s): 14.5% user, 35.0% system, 1.4% nice, 49.1% idle
Mem: 256712k total, 253148k used, 3564k free, 20148k buffers
Swap: 257032k total, 47172k used, 209860k free, 95508k cached

```

PID	USER	VIRT	RES	SHR	%MEM	PPID	SWAP	CODE	DATA	nDRT	COMMAND
1	root	472	432	412	0.2	0	40	24	408	5	init
4	root	0	0	0	0.0	1	0	0	0	0	kswapd
85	root	0	0	0	0.0	1	0	0	0	0	kjournald
334	root	596	556	480	0.2	1	40	32	524	19	syslogd
348	root	524	444	424	0.2	1	80	20	424	5	gpm
765	rms46	1928	944	928	0.4	1	984	32	912	23	kdeinit
797	rms46	6932	5480	3576	2.1	765	1452	16	5464	580	kdeinit
817	rms46	1216	1144	1052	0.4	797	72	408	736	31	bash
5441	rms46	932	932	696	0.4	817	0	44	888	59	top
819	rms46	1212	1136	1072	0.4	797	76	404	732	32	bash
27506	rms46	908	908	760	0.4	819	0	308	600	37	shsh
27507	rms46	920	920	808	0.4	27506	0	316	604	38	sh
5433	rms46	1764	1764	660	0.7	27507	0	132	1632	282	rsync
5434	rms46	1632	1628	1512	0.6	5433	4	124	1504	250	rsync
5435	rms46	1832	1832	1524	0.7	5434	0	140	1692	298	rsync
27286	rms46	24244	23m	14m	9.4	765	0	52	23m	2591	firefox-bin
27400	rms46	24244	23m	14m	9.4	27286	0	52	23m	2591	firefox-bin
27401	rms46	24244	23m	14m	9.4	27400	0	52	23m	2591	firefox-bin
27354	rms46	17748	17m	7948	6.9	1	0	496	16m	2546	evolution-mail
27520	rms46	17748	17m	7948	6.9	27354	0	496	16m	2546	evolution-mail
27521	rms46	17748	17m	7948	6.9	27520	0	496	16m	2546	evolution-mail

(Status Memori)

- Berapakah ukuran total, memori fisik dari sistem tersebut di atas?
- Terangkan, apa yang dimaksud dengan: "VIRT", "RES", "SHR", "PPID", "SWAP", "CODE", "DATA", "nDRT".
- Bagaimanakah, hubungan (rumus) antara "RES" dengan parameter lainnya?
- Bagaimanakah, hubungan (rumus) antara "VIRT", dengan parameter lainnya?

Managemen Memori dan Utilisasi CPU (2004)

- Terangkan bagaimana pengaruh derajat "*multiprogramming*" (MP) terhadap utilisasi CPU. Apakah peningkatan MP akan selalu meningkatkan utilisasi CPU? Mengapa?
- Terangkan bagaimana pengaruh dari "*page-fault*" memori terhadap utilisasi CPU!
- Terangkan bagaimana pengaruh ukuran memori (RAM size) terhadap utilisasi CPU!
- Terangkan bagaimana pengaruh memori virtual (VM) terhadap utilisasi CPU!
- Terangkan bagaimana pengaruh teknologi "*copy on write*" terhadap utilisasi CPU!
- Sebutkan Sistem Operasi berikut mana saja yang telah mengimplementasi teknologi "*copy on write*": Linux 2.4, Solaris 2, Windows 2000.

Memori I (2002)

Diketahui spesifikasi sistem memori virtual **sebuah proses** sebagai berikut:

- *page replacement* menggunakan algoritma LRU (*Least Recently Used*).
- alokasi memori fisik dibatasi hingga 1000 bytes (per proses).
- ukuran halaman (*page size*) harus tetap (*fixed*, minimum 100 bytes).
- usahakan, agar terjadi "*page fault*" sesedikit mungkin.
- proses akan mengakses alamat berturut-turut sebagai berikut:

1001, 1002, 1003, 2001, 1003, 2002, 1004, 1005, 2101, 1101,
2099, 1001, 1115, 3002, 1006, 1007, 1008, 1009, 1101, 1102

- Tentukan ukuran halaman yang akan digunakan.
- Berapakah jumlah *frame* yang dialokasikan?
- Tentukan *reference string* berdasarkan ukuran halaman tersebut di atas!
- Buatlah bagan untuk algoritma LRU!
- Tentukan jumlah *page-fault* yang terjadi!

Memori II (2003)

Sebuah proses secara berturut-turut mengakses alamat memori berikut:

1001, 1002, 1003, 2001, 2002, 2003, 2601, 2602, 1004, 1005,
1507, 1510, 2003, 2008, 3501, 3603, 4001, 4002, 1020, 1021.

Ukuran setiap halaman (*page*) ialah 500 bytes.

- Tentukan "*reference string*" dari urutan pengaksesan memori tersebut.
- Gunakan algoritma "*Optimal Page Replacement*".
Tentukan jumlah "*frame*" minimum yang diperlukan agar terjadi "*page fault*" minimum!
Berapakah jumlah "*page fault*" yang terjadi? Gambarkan dengan sebuah bagan!
- Gunakan algoritma "*Least Recently Used (LRU)*".
Tentukan jumlah "*frame*" minimum yang diperlukan agar terjadi "*page fault*" minimum!
Berapakah jumlah "*page fault*" yang terjadi? Gambarkan dengan sebuah bagan!
- Gunakan jumlah "*frame*" hasil perhitungan butir "b" di atas serta algrgoritma LRU.
Berapakah jumlah "*page fault*" yang terjadi? Gambarkan dengan sebuah bagan!

Memori III (2002)

- Terangkan, apa yang dimaksud dengan algoritma penggantian halaman *Least Recently Used* (LRU)!
- Diketahui sebuah *reference string* berikut: " 1 2 1 7 6 7 3 4 3 5 6 7 ". Jika proses mendapat alokasi tiga *frame*; gambarkan pemanfaatan *frame* tersebut menggunakan *reference string* tersebut di atas menggunakan algoritma LRU.
- Berapa *page fault* yang terjadi?
- Salah satu implementasi LRU ialah dengan menggunakan stack; yaitu setiap kali sebuah halaman memori dirujuk, halaman tersebut diambil dari stack serta diletakkan ke atas (TOP of) stack. Gambarkan urutan penggunaan stack menggunakan *reference string* tersebut.

Multilevel Paging Memory I (2003)

Diketahui sekeping memori berukuran 32 byte dengan alamat fisik "00" - "1F" (Heksadesimal) - yang digunakan secara "*multilevel paging*" - serta dialokasikan untuk keperluan berikut:

- "*Outer Page Table*" ditempatkan secara permanen (*non-swappable*) pada alamat "00" - "07" (Heks).
- Terdapat alokasi untuk dua (2) "*Page Table*", yaitu berturut-turut pada alamat "08" - "0B" dan "0C" - "0F" (Heks). Alokasi tersebut dimanfaatkan oleh semua "*Page Table*" secara bergantian (*swappable*) dengan algoritma "*LRU*".
- Sisa memori "10" - "1F" (Heks) dimanfaatkan untuk menempatkan sejumlah "*memory frame*".

Keterangan tambahan perihal memori sebagai berikut:

- Ukuran "*Logical Address Space*" ialah tujuh (7) bit.
- Ukuran data ialah satu byte (8 bit) per alamat.
- "*Page Replacement*" menggunakan algoritma "*LRU*".
- "*Invalid Page*" ditandai dengan bit pertama (MSB) pada "*Outer Page Table*"/"*Page Table*" diset menjadi "1".
- sebaliknya, "*Valid Page*" ditandai dengan bit pertama (MSB) pada "*Outer Page Table*"/"*Page Table*" diset menjadi "0", serta berisi alamat awal (*pointer*) dari "*Page Table*" terkait.

Pada suatu saat, isi keping memori tersebut sebagai berikut:

<i>address</i>	<i>isi</i>	<i>address</i>	<i>isi</i>	<i>address</i>	<i>isi</i>	<i>address</i>	<i>isi</i>
00H	08H	08H	10H	10H	10H	18H	18H
01H	0CH	09H	80H	11H	11H	19H	19H
02H	80H	0AH	80H	12H	12H	1AH	1AH
03H	80H	0BH	18H	13H	13H	1BH	1BH
04H	80H	0CH	14H	14H	14H	1CH	1CH
05H	80H	0DH	1CH	15H	15H	1DH	1DH
06H	80H	0EH	80H	16H	16H	1EH	1EH
07H	80H	0FH	80H	17H	17H	1FH	1FH

- a) Berapa byte, kapasitas maksimum dari "Virtual Memory" dengan "Logical Address Space" tersebut?
- b) Gambarkan pembagian "Logical Address Space" tersebut: berapa bit untuk P1/"Outer Page Table", berapa bit untuk P2/"Page Table", serta berapa bit untuk alokasi offset?



- c) Berapa byte, ukuran dari sebuah "memory frame" ?
- d) Berapa jumlah total dari "memory frame" pada keping tersebut?
- e) Petunjuk: Jika terjadi "page fault", terangkan juga apakah terjadi pada "Outer Page Table" atau pada "Page Table". Jika tidak terjadi "page fault", sebutkan isi dari Virtual Memory Address berikut ini:
- Virtual Memory Address: 00H
 - Virtual Memory Address: 3FH
 - Virtual Memory Address: 1AH

Multilevel Paging Memory II (2004)

Diketahui sekeping memori berukuran 32 byte dengan alamat fisik "00" - "1F" (Heksadesimal) - yang digunakan secara "multilevel paging" - serta dialokasikan dengan ketentuan berikut:

- "Outer Page Table" ditempatkan secara permanen (*non-swappable*) pada alamat "00" - "03" (Heks).
 - Terdapat alokasi untuk tiga (3) "Page Table", yaitu berturut-turut pada alamat "04" - "07", "08-0B", dan "0C" - "0F" (Heks).
 - Sisa memori "10" - "1F" (Heks) dimanfaatkan untuk menempatkan sejumlah "memory frame".
- Ukuran "Logical Address Space" ialah tujuh (7) bit.
 - Ukuran data ialah satu byte (8 bit) per alamat.
 - "Page Replacement" menggunakan alrorithma "**LRU**".
 - "Invalid Page" ditandai dengan bit pertama (MSB) pada "Outer Page Table"/"Page Table" diset menjadi "1".
 - sebaliknya, "Valid Page" ditandai dengan bit pertama (MSB) pada "Outer Page Table"/"Page Table" diset menjadi "0", serta berisi alamat awal (*pointer*) dari "Page Table" terkait.

Pada suatu saat, isi keping memori tersebut sebagai berikut:

<i>address</i>	<i>isi</i>	<i>address</i>	<i>isi</i>	<i>address</i>	<i>isi</i>	<i>address</i>	<i>isi</i>
00H	80H	08H	80H	10H	10H	18H	18H
01H	04H	09H	80H	11H	11H	19H	19H
02H	08H	0AH	80H	12H	12H	1AH	1AH
03H	0CH	0BH	80H	13H	13H	1BH	1BH
04H	80H	0CH	80H	14H	14H	1CH	1CH
05H	10H	0DH	80H	15H	15H	1DH	1DH
06H	80H	0EH	80H	16H	16H	1EH	1EH
07H	80H	0FH	18H	17H	17H	1FH	1FH

- a) Berapa byte, kapasitas maksimum dari "Virtual Memory" dengan "Logical Address Space" tersebut?
- b) Gambarkan pembagian "Logical Address Space" tersebut: berapa bit untuk P1/"Outer Page Table", berapa bit untuk P2/"Page Table", serta berapa bit untuk alokasi offset?



- c) Berapa byte, ukuran dari sebuah "memory frame" ?
- d) Berapa jumlah total dari "memory frame" pada keping tersebut?
- e) Petunjuk: Jika terjadi "page fault", terangkan juga apakah terjadi pada "Outer Page Table" atau pada "Page Table". Jika tidak terjadi "page fault", sebutkan isi dari Virtual Memory Address berikut ini:
 - a) Virtual Memory Address: 00H
 - b) Virtual Memory Address: 28H
 - c) Virtual Memory Address: 55H
 - d) Virtual Memory Address: 7BH

Multilevel Paging Memory III (2005)

[1 k = 2^{10} ; 1 M = 2^{20} ; 1 G = 2^{30}]

- a) Sebuah sistem komputer menggunakan ruang alamat logika (*logical address space*) 32 bit dengan ukuran halaman (*page size*) 4 kbyte. Jika sistem menggunakan skema tabel halaman satu tingkat (*single level page table*); perkirakan ukuran memori yang diperlukan untuk tabel halaman tersebut! Jangan lupa: setiap masukan tabel halaman memerlukan satu bit ekstra sebagai *flag*!
- b) Jika sistem menggunakan skema tabel halaman dua tingkat (*two level page table*) dengan ukuran *outer-page* 10 bit; tentukan bagaimana konfigurasi minimum tabel yang diperlukan (minimum berapa *outer-page table* dan minimum berapa *page table*)? Perkirakan ukuran memori yang diperlukan untuk konfigurasi minimum tersebut?
- c) Terangkan keuntungan dan kerugian skema tabel halaman satu tingkat tersebut!
- d) Terangkan keuntungan dan kerugian skema tabel halaman dua tingkat tersebut!
- e) Terangkan mengapa skema table halaman bertingkat kurang cocok untuk ruang alamat yang lebih besar dari 32 bit? Bagaimana cara mengatasi hal tersebut?

FHS (File Hierarchy Standards) (2002)

- a. Sebutkan tujuan dari FHS.
- b. Terangkan perbedaan antara "*shareable*" dan "*unshareable*"
- c. Terangkan perbedaan antara "*static*" dan "*variable*"
- d. Terangkan/berikan ilustrasi sebuah direktori yang "*shareable*" dan "*static*".
- e. Terangkan/berikan ilustrasi sebuah direktori yang "*shareable*" dan "*variable*".
- f. Terangkan/berikan ilustrasi sebuah direktori yang "*unshareable*" dan "*static*".
- g. Terangkan/berikan ilustrasi sebuah direktori yang "*unshareable*" dan "*variable*".

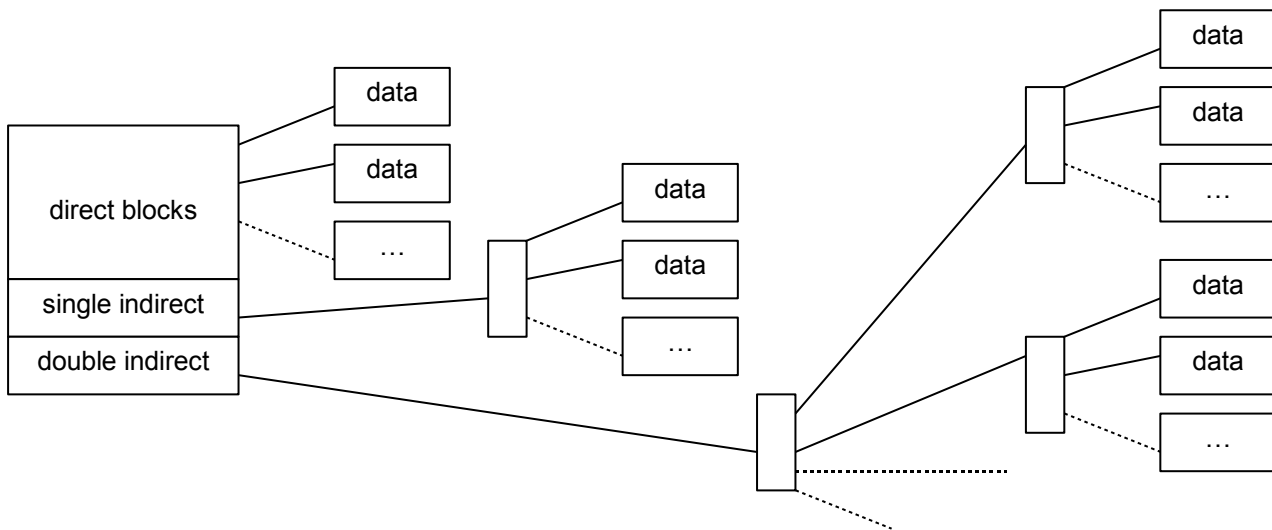
Sistem Berkas I (2003)

Pada saat merancang sebuah situs web, terdapat pilihan untuk membuat link berkas yang **absolut** atau pun **relatif**.

- Berikan sebuah contoh, link berkas yang absolut.
- Berikan sebuah contoh, link berkas yang relatif.
- Terangkan keunggulan dan/atau kekurangan jika menggunakan link absolut.
- Terangkan keunggulan dan/atau kekurangan jika menggunakan link relatif.

Sistem Berkas II (2002)

Sebuah sistem berkas menggunakan metoda alokasi serupa *i-node* (unix). Ukuran pointer berkas (*file pointer*) ditentukan 10 bytes. Inode dapat mengakomodir 10 *direct blocks*, serta masing-masing sebuah *single indirect block* dan sebuah *double indirect block*.



- Jika ukuran blok = 100 bytes, berapakah ukuran maksimum sebuah berkas?
- Jika ukuran blok = 1000 bytes, berapakah ukuran maksimum sebuah berkas?
- Jika ukuran blok = N bytes, berapakah ukuran maksimum sebuah berkas?

Sistem Berkas III (2004)

- Terangkan persamaan dan perbedaan antara operasi dari sebuah **sistem direktori** dengan operasi dari sebuah **sistem sistem berkas (filesystem)**.
- Silberschatz et. al. mengilustrasikan sebuah model sistem berkas berlapis enam (6 layers), yaitu "*application programs*", "*logical file system*", "*file-organization module*", "*basic file system*", "*I/O control*", "*devices*". Terangkan lebih rinci serta berikan contoh dari ke-enam lapisan tersebut!
- Terangkan mengapa pengalokasian blok pada sistem berkas berbasis FAT (MS DOS) dikatakan efisien! Terangkan pula kelemahan dari sistem berkas berbasis FAT tersebut!
- Sebutkan dua fungsi utama dari sebuah Virtual File Sistem (secara umum atau khusus Linux).

Sistem Berkas IV (2005)

- a) Terangkan kedua fungsi dari sebuah VFS (*Virtual File System*).
- b) Bandingkan implementasi sistem direktori antara "*Linier List*" dan "*Hash Table*". Terangkan kelebihan/kekurangan masing-masing!

RAID (*Redudant Array of I* Disks*) (2004)

- a) Terangkan dan ilustrasikan: apa yang dimaksud dengan RAID level 0
- b) Terangkan dan ilustrasikan: apa yang dimaksud dengan RAID level 1
- c) Terangkan dan ilustrasikan: apa yang dimaksud dengan RAID level 0 + 1
- d) Terangkan dan ilustrasikan: apa yang dimaksud dengan RAID level 1 + 0

Mass Storage System I (2002)

Bandingkan jarak tempuh (dalam satuan silinder) antara penjadualan FCFS (*First Come First Served*), SSTF (*Shortest-Seek-Time-First*), dan LOOK. Isi antrian permintaan akses berturut-turut untuk silinder:

100, 200, 300, 101, 201, 301.

Posisi awal *disk head* pada silinder 0.

Mass Storage System II (2003)

Posisi awal sebuah "*disk head*" pada silinder 0. Antrian permintaan akses berturut-turut untuk silinder:

100, 200, 101, 201.

- a) Hitunglah jarak tempuh (dalam satuan silinder) untuk algoritma penjadualan "*First Come First Served*" (FCFS).
- b) Hitunglah jarak tempuh (dalam satuan silinder) untuk algoritma penjadualan "*Shortest Seek Time First*" (STTF).

Mass Storage System III (2003)

Pada sebuah PC terpasang sebuah disk IDE/ATA yang berisi dua sistem operasi: MS Windows 98 SE dan Debian GNU/Linux Woody 3.0 r1.

Informasi "*fdisk*" dari perangkat disk tersebut sebagai berikut:

```
# fdisk /dev/hda
```

=====

Device	Boot	Start	End	Blocks	Id	System
		(cylinders)		(kbytes)		
/dev/hda1	*	1	500	4000000	0B	Win95 FAT32
/dev/hda2		501	532	256000	82	Linux swap
/dev/hda3		533	2157	13000000	83	Linux
/dev/hda4		2158	2500	2744000	83	Linux

Sedangkan informasi berkas "**fstab**" sebagai berikut:

```
# cat /etc/fstab
```

```
# <file system> <mount point> <type> <options> <dump> <pass>
# -----
/dev/hda1      /win98        vfat          defaults      0             2
/dev/hda2      none          swap          sw            0             0
/dev/hda3      /             ext2          defaults      0             0
/dev/hda4      /home        ext2          defaults      0             2
```

Gunakan pembulatan 1 Gbyte = 1000 Mbytes = 1000000 kbytes dalam perhitungan berikut ini:

- Berapa Gbytes kapasitas disk tersebut di atas?
- Berapa jumlah silinder disk tersebut di atas?
- Berapa Mbytes terdapat dalam satu silinder?
- Berapa Mbytes ukuran partisi dari direktori "/home"?

Tambahkan disk ke dua (/dev/hdc) dengan spesifikasi teknis serupa dengan disk tersebut di atas (/dev/hda). Bagilah disk kedua menjadi tiga partisi:

- 4 Gbytes untuk partisi Windows FAT32 (Id: 0B)
- 256 Mbytes untuk partisi Linux Swap (Id: 82)
- Sisa disk untuk partisi "/home" yang baru (Id: 83).

Partisi "/home" yang lama (disk pertama) dialihkan menjadi "/var".

- Bagaimana bentuk informasi "fdisk" untuk "/dev/hdc" ini?
- Bagaimana seharusnya isi berkas "/etc/fstab" setelah penambahan disk tersebut?

Sistem Berkas "ReiserFS" (2003)

- Terangkan secara singkat, titik fokus dari pengembangan sistem berkas "*reiserfs*": apakah berkas berukuran besar atau kecil, serta terangkan alasannya!
- Sebutkan secara singkat, dua hal yang menyebabkan ruangan (*space*) sistem berkas "*reiserfs*" lebih efisien!
- Sebutkan secara singkat, manfaat dari "*balanced tree*" dalam sistem berkas "*reiserfs*"!
- Sebutkan secara singkat, manfaat dari "*journaling*" pada sebuah sistem berkas!
- Sistem berkas "*ext2fs*" dilaporkan 20% lebih cepat jika menggunakan blok berukuran 4 *kbyte* dibandingkan 1 *kbyte*. Terangkan mengapa penggunaan ukuran blok yang besar dapat meningkatkan kinerja sistem berkas!
- Para pengembang sistem berkas "*ext2fs*" merekomendasikan blok berukuran 1 *kbyte* dari pada yang berukuran 4 *kbyte*. Terangkan, mengapa perlu menghindari penggunaan blok berukuran besar tersebut!

I/O Interface (2003)

Bandingkan perangkat disk yang berbasis IDE/ATA dengan yang berbasis SCSI:

- Sebutkan kepanjangan dari IDE/ATA.
- Sebutkan kepanjangan dari SCSI.
- Berapakah kisaran harga kapasitas disk IDE/ATA per satuan Gbytes?
- Berapakah kisaran harga kapasitas disk SCSI per satuan Gbytes?
- Bandingkan beberapa parameter lainnya seperti unjuk kerja, jumlah perangkat, penggunaan CPU, dst.

I/O dan USB (2004)

- Sebutkan sedikitnya sepuluh (10) kategori perangkat yang telah berbasis USB!
- Standar IEEE 1394b (FireWire800) memiliki kinerja tinggi, seperti kecepatan alih data 800 MBit per detik, bentangan/jarak antar perangkat hingga 100 meter, serta dapat menyalurkan catu daya hingga 45 Watt. Bandingkan spesifikasi tersebut dengan USB 1.1 dan USB 2.0.
- Sebutkan beberapa keunggulan perangkat USB dibandingkan yang berbasis standar IEEE 1394b tersebut di atas!
- Sebutkan dua trend perkembangan teknologi perangkat I/O yang saling bertentangan (konflik).
- Sebutkan dua aspek dari sub-sistem I/O kernel yang menjadi perhatian utama para perancang Sistem Operasi!
- Bagaimana USB dapat mengatasi trend dan aspek tersebut di atas?

Struktur Keluaran/Masukan Kernel (I/O) (2004)

- Buatlah sebuah bagan yang menggambarkan hubungan/relasi antara lapisan-lapisan (layers) kernel, subsistem M/K (I/O), device driver, device controller, dan devices.
- Dalam bagan tersebut, tunjukkan dengan jelas, bagian mana yang termasuk perangkat keras, serta bagian mana yang termasuk perangkat lunak.
- Dalam bagan tersebut, berikan contoh sekurangnya dua devices!
- Terangkan apa yang dimaksud dengan *devices*!
- Terangkan apa yang dimaksud dengan *device controller*!
- Terangkan apa yang dimaksud dengan *device driver*!
- Terangkan apa yang dimaksud dengan subsistem M/K (I/O)!
- Terangkan apa yang dimaksud dengan kernel!

Masukan/Keluaran (2005)

- Terangkan secara singkat, sekurangnya enam prinsip/cara untuk meningkatkan efisiensi M/K (Masukan/Keluaran)!
- Diketahui sebuah model M/K yang terdiri dari lapisan-lapisan berikut: Aplikasi, Kernel, *Device-Driver*, *Device-Controller*, *Device*. Terangkan pengaruh pemilihan lapisan tersebut untuk pengembangan sebuah aplikasi baru. Diskusikan aspek-aspek berikut ini: Jumlah waktu pengembangan, Efisiensi, Biaya Pengembangan, Abstraksi, dan Fleksibilitas.

HardDisk I (2001)

Diketahui sebuah perangkat DISK dengan spesifikasi:

- Kapasitas 100 Gbytes (asumsi 1Gbytes = 1000 Mbytes).
 - Jumlah lempengan (plate) ada dua (2) dengan masing-masing dua (2) sisi permukaan (surface).
 - Jumlah silinder = 2500 (Revolusi: 6000 RPM)
 - Pada suatu saat, hanya satu HEAD (pada satu sisi) yang dapat aktif.
- a) Berapakah waktu latensi maksimum dari perangkat DISK tersebut?
b) Berapakah rata-rata latensi dari perangkat DISK tersebut?
c) Berapakah waktu minimum (tanpa latensi dan seek) yang diperlukan untuk mentransfer satu juta (1 000 000) byte data?

HardDisk II (2003)

Diketahui sebuah disk dengan spesifikasi berikut ini:

- Dua (2) permukaan (*surface #0, #1*).
 - Jumlah silinder: 5000 (*cyl. #0 - #4999*).
 - Kecepatan Rotasi: 6000 rpm.
 - Kapasitas Penyimpanan: 100 Gbyte.
 - Jumlah sektor dalam satu trak: 1000 (*sec. #0 - #999*).
 - Waktu tempuh *seek* dari *cyl. #0* hingga *#4999* ialah 10 mS.
 - Pada $T=0$, *head* berada pada posisi *cyl #0, sec. #0*.
 - Satuan I/O terkecil untuk baca/tulis ialah satu (1) sektor.
 - Akan menulis data sebanyak 5010 byte pada *cyl. #500, surface #0, sec. #500*.
 - Untuk memudahkan, 1 kbyte = 1000 byte; 1 Mbyte = 1000 kbyte; 1 Gbyte = 1000 Mbyte.
- a) Berapakah kecepatan *seek* dalam satuan *cyl/ms* ?
b) Berapakah *rotational latency* (max.) dalam satuan ms ?
c) Berapakah jumlah (*byte*) dalam satu sektor ?
d) Berapa lama (ms) diperlukan *head* untuk mencapai *cyl. #500* dari *cyl. #0, sec. #0* ?
e) Berapa lama (ms) diperlukan *head* untuk mencapai *cyl. #500, sec. #500* dari *cyl. #0, sec. #0* ?
f) Berapa lama (ms) diperlukan untuk menulis kedalam **satu** sektor ?
g) Berdasarkan butir (e) dan (f) di atas, berapa kecepatan transfer efektif untuk menulis data sebanyak 5010 byte ke dalam disk tersebut dalam satuan Mbytes/detik?

HardDisk III (2004)

Diketahui sebuah disk dengan spesifikasi berikut ini:

- Dua (2) permukaan (muka *#0 dan #1*).
- Jumlah silinder: 5000 (silinder *#0 - #4999*).
- Kecepatan Rotasi: 6000 rpm.
- Kapasitas Penyimpanan: 100 Gbytes.
- Jumlah sektor dalam satu trak: 1000 (sektor *#0 - #999*).
- Waktu tempuh hingga stabil antar trak yang berurutan: 1 mS (umpama dari trak *#1* ke trak *#2*).
- Pada setiap saat, hanya satu muka yang *head*-nya aktif (baca/tulis). Waktu alih antar muka (dari muka *#0* ke muka *#1*) dianggap 0 mS.
- Algoritma pergerakan *head*: *First Come First Served*.
- Satuan I/O terkecil untuk baca/tulis ialah satu (1) sektor.
- Pada $T=0$, *head* berada pada posisi silinder *#0*, sektor *#0*.

• Untuk memudahkan, 1 kbyte = 1000 byte; 1 Mbyte = 1000 kbyte; 1 Gbyte = 1000 Mbyte.

- Berapa kapasitas (*Mbyte*) dalam satu trak?
- Berapa kapasitas (*kbyte*) dalam satu sektor?
- Berapakah *rotational latency* maksimum (mS) ?
- Berapakah waktu tempuh (mS) dari muka #0, trak #0, sektor #0 ke muka #0, trak #0, sektor #999 (**[0,0,0]** → **[0,0,999]**)?
- Berapakah waktu tempuh (mS) dari **[0,0,0]** → **[0,0,999]** → **[0,1,500]** → **[0,1,999]** → **[0,1,0]** → **[0,1,499]**?
- Berapakah waktu tempuh (mS) dari **[0,0,0]** → **[0,0,999]** → **[0,1,0]** → **[0,1,999]**?

Waktu Nyata/Multimedia (2005)

- Sebutkan sekurang-nya empat ciri/karakteristik dari sebuah sistem waktu nyata. Terangkan secara singkat, maksud masing-masing ciri tersebut!
- Sebutkan sekurang-nya tiga ciri/karakteristik dari sebuah sistem multimedia. Terangkan secara singkat, maksud masing-masing ciri tersebut!
- Terangkan perbedaan prinsip kompresi antara berkas dengan format “JPEG” dan berkas dengan format “GIF”.
- Terangkan lebih rinci, makna keempat faktor QoS berikut ini: *throughput*, *delay*, *jitter*, *reliability*!

Tugas Kelompok/Buku Sistem Operasi (2004)

Bandingkan buku Sistem Operasi versi 1.3 (terbitan awal 2003) dengan versi 1.9 (terbitan akhir 2003):

- Sebutkan beberapa perbaikan/kemajuan umum buku versi 1.9 ini, dibandingkan dengan versi sebelumnya.
- Sebutkan hal-hal yang masih perlu mendapatkan perhatian/perbaikan.
- Penulisan Pokok Bahasan mana yang terbaik untuk versi 1.9 ini? Sebutkan alasannya!
- Sebutkan sebuah Sub-Pokok Bahasan (**SPB**) yang anda ingat/kuasai (tidak harus yang anda kerjakan). SPB tersebut merupakan bagian dari Pokok Bahasan yang mana?
- Bandingkan SPB tersebut di butir "d" dengan SPB setara yang ada di buku-buku Silberschatz et. al.; Tanenbaum, dan Stalling. Dimana perbedaan/persamaannya?