

# CSGE602055 Operating Systems

## CSF2600505 Sistem Operasi

### Week 05: Virtual Memory

Rahmat M. Samik-Ibrahim (ed.)

University of Indonesia

<https://os.vlsm.org/>

Always check for the latest revision!

REV192 13-Feb-2019

# Operating Systems 2019-1

A (Rm 3114) [Tu/Th 10-12] — B (Rm 3114) [Tu/Th 13-15] — C (Rm 3114)

[Tu/Th 16-18] — D (Rm 2401) [Tu/Th 10-12] — E (Rm 2306) [Tu/Th 13-15]

Week	Schedule	Topic	OSC10
Week 00	07 Feb - 13 Feb 2019	Overview 1, Virtualization & Scripting	Ch. 1, 2, 18.
Week 01	14 Feb - 20 Feb 2019	Overview 2, Virtualization & Scripting	Ch. 1, 2, 18.
Week 02	21 Feb - 27 Feb 2019	Security, Protection, Privacy, & C-language	Ch. 16, 17
Week 03	28 Feb - 06 Mar 2019	File System & FUSE	Ch. 13, 14, 15
Week 04	12 Mar - 18 Mar 2019	Addressing, Shared Lib, & Pointer	Ch. 9
Week 05	19 Mar - 25 Mar 2019	Virtual Memory	Ch. 10
Mid-Term	23-30 Mar 2019 (tba)	MidTerm (UTS)	
Week 06	02 Apr - 08 Apr 2019	Concurrency: Processes & Threads	Ch. 3, 4
Week 07	09 Apr - 15 Apr 2019	Synchronization & Deadlock	Ch. 6, 7, 8
Week 08	16 Apr - 22 Apr 2019	Scheduling	Ch. 5
Week 09	23 Apr - 29 Apr 2019	Storage, BIOS, Loader, & Systemd	Ch. 11
Week 10	30 Apr - 06 May 2019	I/O & Programming	Ch. 12
Reserved	07 May - 17 May 2019		
Final	18-25 May 2019 (tba)	Final (UAS)	This schedule is subject to change.
Extra	27 Jun 2019	Extra assignment confirmation	

# The Weekly Check List

- ☐ **Resources:** <https://os.vlsm.org/>
  - ☐ **Download Slides and Demos from GitHub.com**  
<https://github.com/UI-FASILKOM-OS/SistemOperasi/>
  - ☐ **Problems** — <https://rms46.vlsm.org/2/>:  
195.pdf (Week 00), 196.pdf (Week 01), 197.pdf (Week 02),  
198.pdf (Week 03), 199.pdf (Week 04), 200.pdf (Week 05),  
201.pdf (Week 06), 202.pdf (Week 07), 203.pdf (Week 08),  
204.pdf (Week 09), 205.pdf (Week 10).
  - ☐ **Badak All in One** — [BADAK.cs.ui.ac.id:///extra/](http://badak.cs.ui.ac.id///extra/)
- ☐ **Text Book:** any recent/decent OS book. Eg. (**OSC10**) Silberschatz et. al.: **Operating System Concepts**, 10<sup>th</sup> Edition, 2018. See also <http://codex.cs.yale.edu/avi/os-book/OS10/>.
- ☐ Encode your **QRC** with size upto 7cm x 7cm (ca. 400x400 pixels):  
**"OS191 CLASS ID SSO-ACCOUNT Your-Full-Name"**
- ☐ Write your Memo (with QRC) **every week**.
- ☐ Login to [badak.cs.ui.ac.id](http://badak.cs.ui.ac.id) via [kawung.cs.ui.ac.id](http://kawung.cs.ui.ac.id) for at least **10 minutes** every week. Copy the weekly demo folders into your own badak home directory.  
Eg.: `cp -r /extra/Demos/* ~/mydemos/`

# Week 05: Memory

- 1 Start
- 2 Schedule
- 3 Week 05
- 4 Week 05
- 5 Virtual Memory
- 6 Memory Allocation Algorithm
- 7 TOP
- 8 06-memory
- 9 The End

# Week 05 Virtual Memory: Topics<sup>1</sup>

- Review of physical memory and memory management hardware
- Virtual Memory
- Caching
- Memory Allocation
- Memory Performance
- Working sets and thrashing

---

<sup>1</sup>Source: ACM IEEE CS Curricula 2013

# Week 05 Virtual Memory: Learning Outcomes<sup>1</sup>

- Explain memory hierarchy and cost-performance trade-offs. [Familiarity]
- Summarize the principles of virtual memory as applied to caching and paging. [Familiarity]
- Describe the reason for and use of cache memory (performance and proximity, different dimension of how caches complicate isolation and VM abstraction). [Familiarity]
- Defend the different ways of allocating memory to tasks, citing the relative merits of each. [Assessment]
- Evaluate the trade-offs in terms of memory size (main memory, cache memory, auxiliary memory) and processor speed. [Assessment]
- Discuss the concept of thrashing, both in terms of the reasons it occurs and the techniques used to recognize and manage the problem. [Familiarity]

---

<sup>1</sup>Source: ACM IEEE CS Curricula 2013

# Virtual Memory

- Reference: (OSC10-ch10 demo-w05)
- Virtual Memory: Separation Logical from Physical.
- Virtual Address Space: logical view.
- Demand Paging
- Page Flags: Valid / Invalid
- Page Fault
- Demand Paging Performance
- Copy On Write (COW)
- Page Replacement Algorithm
  - Reference String
  - First-In-First-Out (FIFO)
  - Belady Anomaly
  - Optimal Algorithm
  - Least Recently Used (LRU)
  - LRU Implementation
  - Least Frequently Used (LFU)
  - Most Frequently Used (MFU)

# Allocation Algorithm

- Page-Buffering Algorithms
- Allocation of Frames
- Fixed Allocation
- Priority Allocation
- Global vs. Local Allocation
- Non-Uniform Memory Access (NUMA)
- Thrashing
- Working-Set Model
- Shared Memory via Memory-Mapped I/O
- Kernel
  - Buddy System Allocator
  - Slab Allocator



# TOP



A terminal window titled "@rmsbase: ~" with multiple tabs. The terminal shows the following commands and output:

```
>>>>> $ rm -f .toprc
>>>>> $ top
```

Figure: top

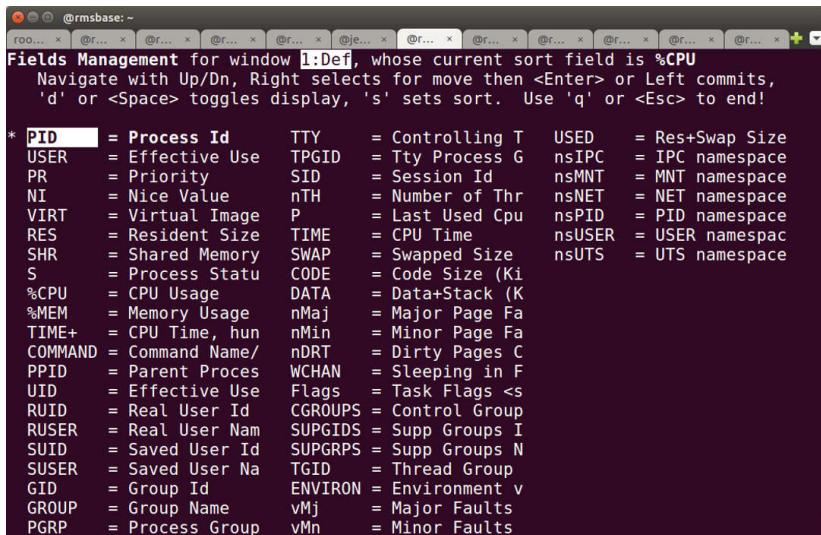
# TOP (2)

```
@rmsbase: ~
top - 18:37:28 up 14:07, 1 user, load average: 2.77, 2.71, 2.74
Tasks: 128 total, 1 running, 127 sleeping, 0 stopped, 0 zombie
%Cpu(s): 14.6 us, 17.2 sy, 0.0 ni, 68.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 8197060 total, 935152 used, 7261908 free, 191512 buffers
KiB Swap: 683004 total, 0 used, 683004 free. 639140 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
518	root	20	0	162032	112	0	S	225.2	0.0	1882:33	rngd
3448	root	20	0	0	0	0	S	14.0	0.0	0:09.14	kworker/0:2
3198	root	20	0	0	0	0	S	9.6	0.0	5:29.03	kworker/4:0
3062	root	20	0	0	0	0	S	5.0	0.0	11:55.39	kworker/1:2
3289	root	20	0	0	0	0	S	2.3	0.0	3:41.00	kworker/6:1
7	root	20	0	0	0	0	S	2.0	0.0	1:08.44	rcu_sched
3376	root	20	0	0	0	0	S	1.3	0.0	0:18.73	kworker/5:0
1914	root	20	0	0	0	0	S	0.3	0.0	13:10.69	kworker/2:1
1	root	20	0	28684	4736	3012	S	0.0	0.1	0:02.91	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:15.26	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:+
8	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
9	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
10	root	rt	0	0	0	0	S	0.0	0.0	0:00.25	watchdog/0
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.28	watchdog/1
12	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/1
13	root	20	0	0	0	0	S	0.0	0.0	0:06.80	ksoftirqd/1

Figure: "h" = help

# TOP (3)

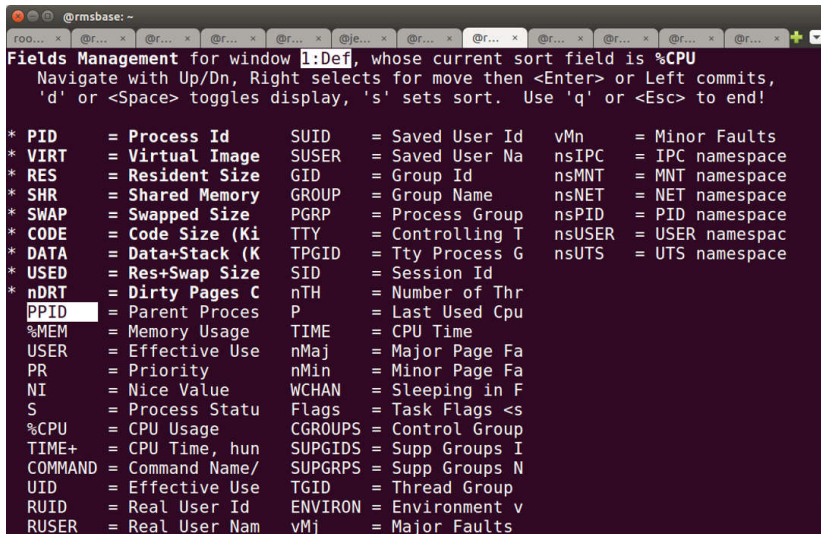


```
@rmsbase: ~
Fields Management for window 1:Def, whose current sort field is %CPU
Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
'd' or <Space> toggles display, 's' sets sort. Use 'q' or <Esc> to end!

* PID = Process Id      TTY = Controlling T      USED = Res+Swap Size
USER = Effective Use    TPGID = Tty Process G    nsIPC = IPC namespace
PR = Priority           SID = Session Id        nsMNT = MNT namespace
NI = Nice Value        nTH = Number of Thr     nsNET = NET namespace
VIRT = Virtual Image    P = Last Used Cpu       nsPID = PID namespace
RES = Resident Size     TIME = CPU Time         nsUSER = USER namespace
SHR = Shared Memory     SWAP = Swapped Size     nsUTS = UTS namespace
S = Process Statu      CODE = Code Size (Ki
%CPU = CPU Usage        DATA = Data+Stack (K
%MEM = Memory Usage     nMaj = Major Page Fa
TIME+ = CPU Time, hun   nMin = Minor Page Fa
COMMAND = Command Name/ nDRT = Dirty Pages C
PPID = Parent Proces    WCHAN = Sleeping in F
UID = Effective Use     Flags = Task Flags <s
RUID = Real User Id     CGROUPS = Control Group
RUSER = Real User Nam   SUPGIDS = Supp Groups I
SUID = Saved User Id    SUPGRPS = Supp Groups N
SUSER = Saved User Na   TGID = Thread Group
GID = Group Id          ENVIRON = Environment v
GROUP = Group Name      vMj = Major Faults
GRPR = Process Group    vMn = Minor Faults
```

Figure: Moving Fields: "f"

# TOP (4)



```
@rmsbase: ~
Fields Management for window 1:Def, whose current sort field is %CPU
Navigate with Up/Dn, Right selects for move then <Enter> or Left commits,
'd' or <Space> toggles display, 's' sets sort. Use 'q' or <Esc> to end!

* PID      = Process Id      SUID       = Saved User Id    vMn        = Minor Faults
* VIRT     = Virtual Image  SUSER      = Saved User Na    nsIPC      = IPC namespace
* RES      = Resident Size  GID        = Group Id        nsMNT      = MNT namespace
* SHR      = Shared Memory  GROUP      = Group Name      nsNET      = NET namespace
* SWAP     = Swapped Size   PGRP       = Process Group  nsPID      = PID namespace
* CODE     = Code Size (Ki  TTY        = Controlling T nsUSER     = USER namespac
* DATA    = Data+Stack (K  TPGID      = Tty Process G nsUTS      = UTS namespace
* USED     = Res+Swap Size  SID        = Session Id
* nDRT     = Dirty Pages C  nTH        = Number of Thr
PPID      = Parent Proces  P          = Last Used Cpu
%MEM      = Memory Usage   TIME       = CPU Time
USER      = Effective Use  nMaj       = Major Page Fa
PR        = Priority       nMin       = Minor Page Fa
NI        = Nice Value    WCHAN      = Sleeping in F
S         = Process Statu  Flags      = Task Flags <s
%CPU      = CPU Usage     CGROUPS    = Control Group
TIME+     = CPU Time, hun  SUPGIDS    = Supp Groups I
COMMAND   = Command Name/ SUPGRPS     = Supp Groups N
UID       = Effective Use  TGID       = Thread Group
RUID      = Real User Id   ENVIRON    = Environment v
RUSER     = Real User Nam  vMj        = Major Faults
```

Figure: Moving Fields

# TOP (5)

```
@rmsbase: ~/Downloads
top - 19:57:14 up 11:38, 1 user, load average: 0.43, 0.54, 0.58
Tasks: 285 total, 2 running, 283 sleeping, 0 stopped, 0 zombie
%Cpu(s): 3.8 us, 1.3 sy, 0.0 ni, 94.6 id, 0.3 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16385976 total, 269672 free, 3179788 used, 12936516 buff/cache
KiB Swap: 1000444 total, 994752 free, 5692 used. 12649780 avail Mem
```

PID	VIRT	RES	SHR	SWAP	CODE	DATA	USED	nDRT
3547	2377296	394828	165776	0	196	1642748	394828	0
1234	278216	87880	59116	0	2288	25164	87880	0
3321	2683572	433176	149376	0	196	1856708	433176	0
2708	1687448	214112	80608	0	12	1179008	214112	0
2841	679488	50860	30484	0	292	389096	50860	0
3748	1896812	321288	76656	0	133688	1474084	321288	0
3971	2047252	440112	97384	0	133688	1587052	440112	0
32501	630768	33500	27960	0	76	373220	33500	0
4067	8554396	320516	109756	0	196	7954584	320516	0
4130	2391592	341632	117636	0	196	1717824	341632	0
22635	2198448	274812	108000	0	196	1532152	274812	0
1292	0	0	0	0	0	0	0	0
2514	930224	34304	26028	0	36	448864	34304	0
3233	4515228	360812	126784	0	133688	3757984	360812	0
32495	33488	3380	2836	0	96	1264	3380	0
2388	44036	4424	2724	0	212	1716	4424	0
2412	423204	11380	5264	0	152	374232	11380	0
2512	685824	74188	36868	0	552	399836	74188	0

Figure: Write Configuration .toprc: "W"

# 06-memory

```
/* Copyright (C) 2016-2018 Rahmat M. Samik-Ibrahim
 * https://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * REV04 Mon Mar 12 17:33:30 WIB 2018
 * START Mon Oct 3 09:26:51 WIB 2016
 */
#define MSIZE0 0x10000
#define MSIZE1 0x10008
#define MSIZE2 0x10009
#define MSIZE3 0x1000A
#define MSIZE4 0x20978
#define MSIZE5 0x20979
#define MSIZE6 0x2097A
#define MSIZE7 0xF0000
#define MSIZE8 0x10000
#define MSIZE9 0x1000
#define LINE 75
#define MAXSTR 80
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>

void printLine(int line) {
    while(line-- > 0) putchar('x');
    putchar('\n');
    fflush(NULL);
}
```

## 06-memory (2)

```
void main (void) {
    int  msize[] = {MSIZE0, MSIZE1, MSIZE2, MSIZE3, MSIZE4,
                   MSIZE5, MSIZE6, MSIZE7, MSIZE8, MSIZE9};

    int  ii, jj;
    int  myPID   = (int) getpid();
    char strSYS1[MAXSTR], strOUT[MAXSTR];
    char* chrStr  = strSYS1;
    char* chrPTR;

    printLine(LINE);
    sprintf(strSYS1, "top -b -n 1 -p%d | tail -5", myPID);
    system (strSYS1);
    sprintf(strSYS1, "top -b -n 1 -p%d | tail -1", myPID);
    for (ii=0; ii< (sizeof(msize)/sizeof(int)); ii++){
        chrStr = malloc(msize[ii]);
        fgets(strOUT, sizeof(strOUT)-1, popen(strSYS1, "r"));
        strOUT[(int) strlen(strOUT)-1]='\0';
        printf("%s [%X]\n", strOUT, msize[ii]);
        free(chrStr);
    }
    for (ii=0; ii< (sizeof(msize)/sizeof(int)); ii++){
        chrPTR = chrStr = malloc(msize[ii]);
        for (jj=0;jj<msize[ii];jj++)
            *chrPTR++='x';
        fgets(strOUT, sizeof(strOUT)-1, popen(strSYS1, "r"));
        strOUT[(int) strlen(strOUT)-1]='\0';
        printf("%s [%X]\n", strOUT, msize[ii]);
        free(chrStr);
    }
}
```

## 06-memory (2)

```
>>>>> $ ./06-memory
```

[illegible]

```
KiB Mem:  8197060 total,  957928 used,  7239132 free,  192520 buffers
```

```
KiB Swap: 683004 total, 0 used, 683004 free. 660108 cached
```

Mem

PID	VIRT	RES	SHR	SWAP	CODE	DATA	USED	nDRT
4362	4172	640	564	0	4	320	640	0
4362	4172	640	564	0	4	320	640	0 [10000]
4362	4172	640	564	0	4	320	640	0 [10008]
4362	4308	640	564	0	4	456	640	0 [10009]
4362	4244	1176	1068	0	4	392	1176	0 [1000A]
4362	4244	1176	1068	0	4	392	1176	0 [20978]
4362	4376	1176	1068	0	4	524	1176	0 [20979]
4362	4376	1192	1068	0	4	524	1192	0 [2097A]
4362	5340	1192	1068	0	4	1488	1192	0 [F0000]
4362	4376	1200	1068	0	4	524	1200	0 [10000]
4362	4376	1200	1068	0	4	524	1200	0 [1000]



## 06-memory (3)

4362	4376	1200	1068	0	4	524	1200	0 [1000]
4362	4376	1200	1068	0	4	524	1200	0 [10000]
4362	4376	1276	1068	0	4	524	1276	0 [10008]
4362	4376	1276	1068	0	4	524	1276	0 [10009]
4362	4376	1284	1068	0	4	524	1284	0 [1000A]
4362	4376	1284	1068	0	4	524	1284	0 [20978]
4362	4376	1352	1068	0	4	524	1352	0 [20979]
4362	4376	1352	1068	0	4	524	1352	0 [2097A]
4362	5340	2144	1068	0	4	1488	2144	0 [F0000]
4362	5340	2324	1068	0	4	1488	2324	0 [10000]
4362	5340	2324	1068	0	4	1488	2324	0 [1000]

>>>>> \$

# The End

- ☐ This is the end of the presentation.
- ☒ This is the end of the presentation.
  - This is the end of the presentation.