

# CSGE602055 Operating Systems

## CSF2600505 Sistem Operasi

### Week 08: Scheduling

Rahmat M. Samik-Ibrahim

University of Indonesia

<http://rms46.vlsm.org/2/207.html>

Always check for the latest revision!

REV120 22-FEB-2018

# Operating Systems 2018-1 (Room 3114 Tue/Thu)

Class: A (10:00-12:00) | B (13:00-15:00) | C (16:00-18:00)

Week	Schedule	Topic	OSC9
Week 00	06 Feb - 12 Feb 2018	Intro & Review1	Ch. 1, 16
Week 01	13 Feb - 19 Feb 2018	Review2 & Scripting	Ch. 1, 2
Week 02	20 Feb - 26 Feb 2018	Protection, Security, Privacy, & C-language	Ch. 14, 15
Week 03	27 Feb - 05 Mar 2018	I/O, BIOS, Loader, & Systemd	Ch. 13
Week 04	06 Mar - 12 Mar 2018	Addressing, Shared Lib, & Pointer	Ch. 8
Week 05	13 Mar - 19 Mar 2018	Virtual Memory	Ch. 9
Reserved	20 Mar - 24 Mar 2018		
Mid-Term	26 Mar - 03 Apr 2018	(UTS)	
Week 06	05 Apr - 11 Apr 2018	Concurrency: Processes & Threads	Ch. 3, 4
Week 07	12 Apr - 18 Apr 2018	Synchronization	Ch. 5, 7
Week 08	19 Apr - 25 Apr 2018	Scheduling	Ch. 6
Week 09	26 Apr - 05 May 2018	File System & Persistent Storage	Ch. 10, 11, 12
Week 10	14 May - 19 May 2018	I/O Programming & Network Sockets Programming	
Reserved	22 May - 22 May 2018		
Final	23 May - 26 May 2018	(UAS)	
Deadline	07 Jun 2018 16:00	Extra assignment <b>deadline</b>	

## ● Operating Systems Check List

- ☐ Trace this document from <http://rms46.vlsm.org/2/207.html>
- ☐ Have a decent OS Book and map it to **OSC9**.
- ☐ Create **public** project "os181" on your github.com account.
  - ☐ Write in "README.md" file:  
Special for Week 00: "**ZCZC Sistem Operasi 2018 Awal (1)**".  
Add line on Week01: "**ZCZC W01 ...**".  
On Week02: "**ZCZC W02 ...**".  
On WeekXX: "**ZCZC WXX ...**".
- ☐ Encode your **QRC** with image size of approximately 250x250 pixels:  
"**OS181 CLASS ID GITHUB-ACCOUNT SSO-ACCOUNT SIAK-Full-Name**"  
Special for Week 00: Mail your **embedded** QRC to: os181@vlsm.org  
with Subject: [W00] CLASS ID SIAK-NAME.
- ☐ Write your Memo (with QRC) **every week**.
- ☐ Using your **SSO** account, login to badak.cs.ui.ac.id via kawung.cs.ui.ac.id.
  - ☐ Check folder badak:///extra/Week00/
  - ☐ Week00: Copy the folder to your home directory:  
cp -r /extra/Week00/W00-demos/ W00-demos/
  - ☐ For WeekXX: Copy the folder to your home directory:  
cp -r /extra/WeekXX/WXX-demos/ WXX-demos/ (XX=01, 02,... 10).
- ☐ **How to improve this document?**

# Agenda

- 1 Start
- 2 Agenda
- 3 Scheduling
- 4 Threads
- 5 Scheduling Model
- 6 Sockets
- 7 00-server
- 8 01-client
- 9 OUTPUT: 00-server – 01-client
- 10 02-clisvr
- 11 OUTPUT: 02-clisvr
- 12 The End

- Reference: (OSC9-ch06 demo-w08)
- Scheduling
  - Basic Concepts
    - **WARNING:** It's just a BURST
    - IO Burst
    - CPU Burst
    - CPU Burst vs. Freq (OLD)
  - Utilization, throughput, {turnaround, waiting, response} time.
  - (Burst) Algorithm
    - FCFS, SJF, RR, Priority, Multilevel Queue.
  - Preemptive / Non-preemptive Scheduling
  - I/O Bound / CPU Bound Processes
- Standard Linux Scheduling
  - Completely Fair Scheduler (CFS).
  - Real Time Scheduling.

# Thread Scheduling

- Thread Scheduling
  - User-level thread scheduling
  - Kernel-level thread scheduling
- Multi-threading Models:
  - Many to One Model
  - One to One Model
  - Many to Many Model
- Pthread Contention Scope
  - Process-Contention Scope (PCS): many to many (eg. Linux).
  - System-Contention Scope (SCS): one to one.
- MultiCore/ MultiProcessor/ MultiThread
  - affinity
  - load balancing
- Soft / Hard Real Time
- Big O Notation
  - $O(1)$
  - $O(\log N)$
  - $O(N)$

# Scheduling Model

- Two State Model: CPU State – I/O State – CPU State – ...
  - $n$ : processes in memory.
  - $p$ : I/O time fraction.
  - $p^n$ : probability  $n$  processes waiting for I/O.
  - $1 - p^n$ : CPU utilization of  $n$  processes.
  - $\left[ \frac{(1-p^n)}{n} \right]$ : CPU utilization of ONE processes.
- Example:  $p = 60\% \Rightarrow$  **CPU Utilization Per Process:**  $\left[ \frac{1-(60\%)^n}{n} \right]$

CPU Utilization	Multiprogramming (%)				
N	1	2	3	4	5
Per Process	40	32	26	21	18

- For 5 concurrent processes:  
If total time is 100 seconds, each CPU time will be 18 seconds.

- Sockets

- `atoi()`
- `accept()`
- `bind()`
- `connect()`
- `exit()`
- `fprintf()`
- `getenv()`
- `gethostbyname()`
- `htons()`
- `listen()`
- `memcpy()`
- `memset()`



- Sockets

- `perror()`
- `sizeof()`
- `socket()`
- `snprintf()`
- `strchr()`
- `strcmp()`
- `strncpy()`
- `strlen()`
- `read()`
- `write()`

```
/*
 * (c) 2007-2016 Rahmat M. Samik-Ibrahim -- This is free software
 * This program was copased from the net and hacked until it works.
 * Feel free to copy and/or modify and/or distribute it,
 * provided this notice, and the copyright notice, are preserved.
 * REV00 Tue Nov 8 11:45:35 WIB 2016
 * START Xxx Xxx XX XX:XX:XX UTC 2007
 */

char pesan[]="[FROM SERVER] ACK MESSAGE...\n";
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <arpa/inet.h>
typedef struct sockaddr      sockad;
typedef struct sockaddr_in   sockadin;
typedef struct hostent       shostent;

void error(char *msg){
    perror(msg);
    exit(0);
}
```

## 00-server (2)

```
int main(int argc, char *argv[]) {
    char    buffer[256];
    int      clilen, newsockfd, nn, portno, sockfd;
    sockaddr serv_addr, cli_addr;
    if (argc < 2) {
        fprintf(stderr, "ERROR, no port provided\n");
        exit(1);
    }
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    memset(&serv_addr, 0, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family      = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port        = htons(portno);
    if (bind(sockfd, (sockaddr*)&serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd, (sockaddr*)&cli_addr, (socklen_t*)&clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    memset(buffer, 0, 256);
    nn = read(newsockfd, buffer, 255);
    if (nn < 0)
        error("ERROR reading from socket");
    printf("[FROM CLIENT]:\n %s\n", buffer);
    nn = write(newsockfd, pesan, sizeof(pesan));
    if (nn < 0)
        error("ERROR writing to socket");
    return 0;
}
```

# 01-client

```
/*
 * (c) 2007-2016 Rahmat M. Samik-Ibrahim -- This is free software
 * This program was copased from the net and hacked until it works.
 * Feel free to copy and/or modify and/or distribute it,
 * provided this notice, and the copyright notice, are preserved.
 * REV00 Tue Nov 8 11:45:52 WIB 2016
 * START Xxx Xxx XX XX:XX:XX UTC 2007
 */
```

```
char pesan[]="[FROM SERVER] ACK MESSAGE...\n";
```

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <arpa/inet.h>
typedef struct sockaddr      sockad;
typedef struct sockaddr_in   sockadin;
typedef struct hostent       shostent;
```

```
void error(char *msg){
    perror(msg);
    exit(0);
}
```

```
int main(int argc, char *argv[]) {
    char      buffer[256];
    int       nn, portno, sockfd;
    sockadin  serv_addr;
    shostent* server;
```

# 01-client (2)

```
if (argc < 3) {
    fprintf(stderr, "usage %s hostname port\n", argv[0]);
    exit(0);
}
portno = atoi(argv[2]);
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
    error("ERROR opening socket");
server = gethostbyname(argv[1]);
if (server == NULL) {
    fprintf(stderr, "ERROR, no such host\n");
    exit(0);
}
memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
memmove(&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
serv_addr.sin_port = htons(portno);
if (connect(sockfd, (const struct sockaddr*) &serv_addr, sizeof(serv_addr)) < 0)
    error("ERROR connecting");
printf("Enter the message: ");
memset(buffer, 0, 256);
fgets(buffer, 255, stdin);
nn = write(sockfd, buffer, strlen(buffer));
if (nn < 0)
    error("ERROR writing to socket");
memset(buffer, 0, 256);
nn = read(sockfd, buffer, 255);
if (nn < 0)
    error("ERROR reading from socket");
printf("%s\n", buffer);
return 0;
}
```

## OUTPUT: 00-server – 01-client

```
>>>> $ PS1="SERVER >> "  
SERVER >> 00-server 4444  
[FROM CLIENT]:  
  This is from client via port 4444.
```

```
SERVER >>
```

```
>>>> $ PS1="CLIENT >> "  
CLIENT >> 01-client localhost 4444  
Enter the message: This is from client via port 4444.  
[FROM SERVER] ACK MESSAGE...
```

```
CLIENT >>
```

```
/*
 * (c) 2007 Tadeus Prastowo and Rahmat M. Samik-Ibrahim.
 * (c) 2017 Rahmat M. Samik-Ibrahim.
 * This is free software. It was copased from the net and hacked until
 * it works. Feel free to copy and/or modify and/or distribute it,
 * provided this notice, and the copyright notice, are preserved.
 * REV01 Wed Nov 8 20:00:02 WIB 2017
 * START 2007
 *
 * This program serves as both a client and a server. Three modes of
 * operation are available:
 * - initiating mode
 * - bridging mode
 * - terminating mode
 *
 * The following are how to run thisprogram for each mode:
 * - Initiating mode: client_server null ANOTHER_HOST ANOTHER_PORT
 * - Bridging mode: client_server CURRENT_PORT ANOTHER_HOST ANOTHER_PORT
 * - Terminating mode: client_server CURRENT_PORT null null
 *
 * The program having the initiating mode _MUST_ run last after all other
 * instances of this program with other operational modes has been started.
 *
 * In initiating mode, this program just simply sends a hello message to
 * another instance of this program that operates either as a bridge or
 * as a terminator that this program points to as specified in
 * ANOTHER_HOST and ANOTHER_PORT. After that this program will quit
 * without printing out any message.
 */
```

## 02-clisvr (2)

```
/*
 * In bridging mode, this program just simply waits for an incoming hello
 * message in CURRENT_PORT. Once it receives a hello message, it prints
 * out the message in a certain format. Next, this program forwards the
 * modified message to another instance of this program that acts either as
 * a bridge or as a terminator that this program points to as specified
 * in ANOTHER_HOST and ANOTHER_PORT. After that this program will quit.
 *
 * In terminating mode, this program just simply waits for an incoming hello
 * message in CURRENT_PORT. Once it receives a hello message, it prints out
 * the message in a certain format, and then quits.
 *
 * The following illustrates the idea above:
 * 192.168.10.18 (alvin)
 * $ ./client_server 8888 localhost 7777
 * 192.168.10.18 (user)$
 * $ ./client_server 7777 null null
 * 192.168.12.17 (eus)$
 * $ ./client_server null 192.168.10.18 8888
 * The print out will be:
 * 192.168.10.18 (alvin):
 *   From eus to alvin: Hello
 * 192.168.10.18 (user):
 *   From eus to alvin to user: Hello
 */
```



## 02-clisvr (3)

```
char pesan[]=" [FROM SERVER] ACK MESSAGE...\n";
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <arpa/inet.h>

typedef struct sockaddr    sockad;
typedef struct sockaddr_in sockadin;
typedef struct hostent     shostent;

void error(char *msg){
    perror(msg);
    exit(0);
}
```

## 02-clisvr (4)

```
#define BUFFER_SIZE 4096

int main (int argc, char *argv []) {
    int sockfd, newsockfd, portno, clilen, count, nn, sysup;
    char buffer [BUFFER_SIZE], temp_buffer [BUFFER_SIZE], *colon_pos;
    struct sockaddr_in serv_addr, cli_addr;
    struct hostent *server;
    struct timeval tval;

    if (argc < 4) {
        fprintf (stderr,
            "\nUsage: %s this_port next_sever next_server_port\n\n"
            "Start the chain with 'this_port' = 'null'\n\n"
            "Terminte the chain with 'next_server' = 'next_server_port'"
            " = 'null'\n\n", argv [0]);
        exit (1);
    }
}
```

## 02-clisvr (5)

```
if (strcmp (argv [1], "null") == 0) {
    portno = atoi    (argv [3]);
    sockfd = socket (AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        error ("ERROR opening socket");
    }
    server = gethostbyname(argv[2]);
    if (server == NULL) {
        fprintf (stderr, "ERROR, no such host\n");
        exit (1);
    }
    memset (&serv_addr, 0, sizeof (serv_addr));
    serv_addr.sin_family = AF_INET;
    memcpy(&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
    serv_addr.sin_port = htons(portno);
    if (connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr))< 0){
        error ("ERROR connecting");
    }
    /* Begin: action */
    memset (buffer, 0, BUFFER_SIZE);
    gettimeofday(&tval,NULL);
    sysup = 0x0000FFFF & (int) (tval.tv_sec * 1000 + tval.tv_usec / 1000);
    snprintf (buffer, BUFFER_SIZE, "From %s[%d]: Hello", getenv ("USER"), sysup);
    nn = write (sockfd, buffer, strlen (buffer));

    if (nn < 0) {
        error ("ERROR writing to socket");
    }
    /* End: action */
    exit (0);
}
```

## 02-clisvr (6)

```
sockfd = socket(AF_INET,SOCK_STREAM,0);
if (sockfd < 0) {
    error ("ERROR opening socket");
}
memset(&serv_addr,0,sizeof(serv_addr));
portno = atoi (argv [1]);
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons (portno);

if (bind (sockfd,(struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    error ("ERROR on binding");
}
listen (sockfd, 5);
clilen    = sizeof (cli_addr);
newsockfd = accept (sockfd, (struct sockaddr *) &cli_addr,
                    (socklen_t *) &clilen);
if (newsockfd < 0) {
    error ("ERROR on accept");
}
memset (buffer, 0, BUFFER_SIZE);
nn = read(newsockfd,buffer,BUFFER_SIZE-1);
if (nn < 0) {
    error ("ERROR reading from socket");
}
```

```
/* Modify buffer's message */
colon_pos = strchr (buffer, ':');
nn        = colon_pos - buffer;
memset (temp_buffer, 0, BUFFER_SIZE);
strncpy (temp_buffer, buffer, nn);
memset (buffer, 0, BUFFER_SIZE);
strncpy (buffer, temp_buffer, nn);
for (long ii=0; ii<5000000L; ii++)
    ; // delay
gettimeofday(&tval,NULL);
sysup = 0x0000FFFF &
    (int) (tval.tv_sec * 1000 + tval.tv_usec / 1000);
snprintf (buffer + nn, BUFFER_SIZE-nn,
    " to %s[%d]: Hello", getenv ("USER"), sysup);
/*End of modifying buffer's message*/
```

## 02-clisvr (8)

```
if (strcmp (argv [2], "null") != 0 && strcmp (argv [3], "null") != 0) {
    portno = atoi (argv [3]);
    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if (sockfd < 0) {
        error ("ERROR opening socket");
    }
    server = gethostbyname (argv [2]);
    if (server == NULL) {
        fprintf (stderr, "ERROR, no such host\n");
        exit (1);
    }
    serv_addr.sin_family = AF_INET;
    memcpy (&serv_addr.sin_addr.s_addr, server->h_addr, server->h_length);
    serv_addr.sin_port = htons (portno);
    if (connect (sockfd,(struct sockaddr *)&serv_addr,sizeof (serv_addr))<0){
        error ("ERROR connecting");
    }
    /* Begin: action */
    printf ("%s\n", buffer);
    nn=write(sockfd,buffer,strlen(buffer));
    if (nn < 0) {
        error ("ERROR writing to socket");
    }
    /* End: action */
} else {
    printf ("%s\n", buffer);
}
return 0;
}
```

# OUTPUT: 02-clisvr

```
TERMINAL >> PS1="TERMINAL >> "  
TERMINAL >> 02-clisvr 4000 localhost null  
From demo[23440] to demo[23450] to demo[23461]: Hello  
TERMINAL >>
```

```
MIDDLE >> PS1="MIDDLE >> "  
MIDDLE >> 02-clisvr 4001 localhost 4000  
From demo[23440] to demo[23450]: Hello  
MIDDLE >>
```

```
START >> PS1="START >> "  
START >> 02-clisvr null localhost 4001  
START >>
```

# The End

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