

# CSGE602055 Operating Systems

## CSF2600505 Sistem Operasi

### Week 10: I/O & Programming

Rahmat M. Samik-Ibrahim (ed.)

University of Indonesia

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

Always check for the latest revision!

REV258 24-Nov-2020

# Operating Systems 202<sup>3)</sup> — PJJ from HOME

## ZOOM: International [Tue 08-10] — A/Matrix [Tue 10-12]

Week	Schedule & Deadline <sup>1)</sup>	Topic	OSC10 <sup>2)</sup>
Week 00	15 Sep - 21 Sep 2020	Overview 1, Virtualization & Scripting	Ch. 1, 2, 18.
Week 01	22 Sep - 28 Sep 2020	Overview 2, Virtualization & Scripting	Ch. 1, 2, 18.
Week 02	29 Sep - 05 Oct 2020	Security, Protection, Privacy, & C-language.	Ch. 16, 17.
Week 03	06 Oct - 12 Oct 2020	File System & FUSE	Ch. 13, 14, 15.
Week 04	13 Oct - 19 Oct 2020	Addressing, Shared Lib, & Pointer	Ch. 9.
Week 05	20 Oct - 26 Oct 2020	Virtual Memory	Ch. 10.
Week 06	27 Oct - 16 Nov 2020 29 Oct 2020	Concurrency: Processes & Threads Maulid Nabi	Ch. 3, 4.
Week 07	17 Nov - 23 Nov 2020	Synchronization & Deadlock	Ch. 6, 7, 8.
Week 08	24 Nov - 30 Nov 2020	Scheduling + W06/W07	Ch. 5.
Week 09	01 Dec - 07 Dec 2020	Storage, Firmware, Bootloader, & Systemd	Ch. 11.
Week 10	08 Dec - 16 Dec 2020	I/O & Programming	Ch. 12.
	09 Dec 2020	Pil Kada	

<sup>1)</sup> The **DEADLINE** of Week 00 is 21 Sep 2020, whereas the **DEADLINE** of Week 01 is 28 Sep 2020, and so on...

<sup>2)</sup> Silberschatz et. al.: **Operating System Concepts**, 10<sup>th</sup> Edition, 2018.

<sup>3)</sup> This information will be on **EVERY** page two (2) of this course material.

# STARTING POINT — <https://os.vlsm.org/>

- **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/>.
- **Resources**
  - **SCELE** — <https://scele.cs.ui.ac.id/course/view.php?id=3020>. The enrollment key is **XXX**.
  - **Download Slides and Demos from GitHub.com**  
<https://github.com/UI-FASILKOM-OS/SistemOperasi/>:  
os00.pdf (W00), os01.pdf (W01), os02.pdf (W02), os03.pdf (W03),  
os04.pdf (W04), os05.pdf (W05), os06.pdf (W06), os07.pdf (W07),  
os08.pdf (W08), os09.pdf (W09), os10.pdf (W10).
  - **Problems** — <https://rms46.vlsm.org/2/>:  
195.pdf (W00), 196.pdf (W01), 197.pdf (W02), 198.pdf (W03),  
199.pdf (W04), 200.pdf (W05), 201.pdf (W06), 202.pdf (W07),  
203.pdf (W08), 204.pdf (W09), 205.pdf (W10).
- **Build your own Virtual Guest**  
<https://osp4diss.vlsm.org/>

# Agenda

- 1 Start
- 2 Schedule
- 3 Agenda
- 4 Week 10
- 5 Week 10: I/O & Programming
- 6 I/O
- 7 PCH: Platform Controller Hub
- 8 Sockets
- 9 10-server
- 10 11-client
- 11 12-clisvr
- 12 XXX

# Agenda (2)

- 13 54-write
- 14 55-write
- 15 57-dup
- 16 58-dup2
- 17 59a-IO
- 18 59b-IO
- 19 59c-IO
- 20 71-os171
- 21 72-os172
- 22 73-os181
- 23 74-os182
- 24 75-os191
- 25 76-os192
- 26 Week 10: Check List
- 27 The End

# Week 10 I/O & Programming: Topics<sup>1</sup>

- Characteristics of serial and parallel devices
- Abstracting device differences
- Buffering strategies
- Direct memory access
- Recovery from failures
- I/O Programming
- Network Programming

---

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

# Week 10 I/O & Programming: Learning Outcomes<sup>1</sup>

- Explain the key difference between serial and parallel devices and identify the conditions in which each is appropriate. [Familiarity]
- Identify the relationship between the physical hardware and the virtual devices maintained by the operating system. [Usage]
- Explain buffering and describe strategies for implementing it. [Familiarity]
- Differentiate the mechanisms used in interfacing a range of devices (including hand-held devices, networks, multimedia) to a computer and explain the implications of these for the design of an operating system. [Usage]
- Describe the advantages and disadvantages of direct memory access and discuss the circumstances in which its use is warranted. [Usage]
- Identify the requirements for failure recovery. [Familiarity]
- Implement a simple device driver for a range of possible devices. [Usage]
- I/O Programming [Usage]
- Network Programming [Usage]

# Week 10: I/O & Programming

- Reference: (OSC10-ch12)
- Overview
- I/O Hardware
- Application I/O Interface
- Kernel I/O Subsystem
- Transforming I/O Requests to Hardware Operations
- STREAMS
- Legacy Linux I/O Scheduling Algorithm.
  - Deadline Scheduler
  - Completely Fair Queueing (CFQ)



# I/O (1)

- Direct I/O vs. Memory Mapped I/O
- Interrupts: Non Maskable (NMI) vs Maskable (MI)
- DMA: Direct Memory Access
- I/O Structure:
  - Kernel (S/W).
  - I/O (S/W: Kernel Subsystem)
  - Driver (S/W)
  - Controller (H/W)
  - Device (H/W)
- I/O Streams
  - APP
  - HEAD
  - MODULES
  - DRIVER
  - H/W.

- I/O Interface Dimensions
  - Character-stream vs. Block;
  - Sequential vs. Random-access;
  - Sharable vs. Dedicated;
  - Parallel vs. Serial;
  - Speed;
  - Read Write – Read Only – Write Only.
  - Synchronous vs. Asynchronous;
  - Blocking vs. Non-Blocking.
- Where should a new algorithm be implemented?
  - APP?
  - Kenel?
  - Driver?
  - Controller?
  - HW?

# PCH: Platform Controller Hub



Figure: PCH: Platform Controller Hub

# Some Terms

- PCH: Platform Controller Hub
  - The successor of north/south-bridge architecture chipsets.
- PCIe: Peripheral Component Interconnect Express
  - 1 lane = dual simplex channel (1x); 2 lanes = 2x; etc.
  - 40 lanes = 8 GTs (GigaTransfers per second).
  - Configurations: 8x and 16x.
- DDR4 SDRAM (single/dual/quad channel(s))
  - Double Data Rate Fourth-generation Synchronous Dynamic Random-Access Memory: 2 x DDR2 (DDR2 = 2 x DDR (DDR = 2 x SDRAM)). Eg. DDR4-3200 (8x SDRAM); Memory Clock: 400 MHz; Data Rate: 3200 MT/s; Module Name PC4-25600; Peak Transfer Rate: 25600 MB/s,
- DMI 2.0 (Direct Media Interface): 4x.
- SMB: System Management Bus
- SPI: Serial Peripheral Interface, a de facto standard bus.
- SATA: Serial AT Attachment. Eg. SATA 3.2  $\approx$  2 GB/s.
- 1 KB (KiloByte) = 1000 bytes — 1 KiB (Kibibyte) = 1024 bytes<sup>1</sup>

<sup>1</sup>In IT tradition; 1 KB = 1024 bytes

- Sockets

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

- Sockets

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

# 10-server (01)

```
/* Copyright (C) 2007-2020 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software.
 * REV02 Sun May 3 07:53:26 WIB 2020
 * 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;
```

# 10-server (02)

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

int main(int argc, char *argv[]) {
    char    buffer[256];
    int     clilen, newsockfd, nn, portno, sockfd;
    sockad_t 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");
    int enable = 1;
    if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR,
        &enable, sizeof(int)) < 0)
        error("setsockopt(SO_REUSEADDR) failed");
    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, (sockad_t) &serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR on binding");
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
```



## 10-server (03)

```
newsockfd=accept(sockfd,(sockad*)&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;
}
```

# 11-client (01)

```
/* Copyright (C) 2007-2018 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software.
 * REV01 Wed Aug 29 20:53:11 WIB 2018
 * 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;
```

# 11-client (02)

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

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

# 11-client (03)

```
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;
}

# TERMINAL 1 #####

$ ./10-server 6666
[FROM CLIENT]:
Hello World!
$

# TERMINAL 2 #####

$ ./11-client localhost 6666
Enter the message: Hello World!
[FROM SERVER] ACK MESSAGE...

$
```

# 12-clisvr (01)

```
/*
 * Copyright (C) 2007 Tadeus Prastowo
 * Copyright (C) 2017 - 2020 Rahmat M. Samik-Ibrahim
 * http://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 Sun May 3 07:59:57 WIB 2020
 * REV03 Wed Feb 27 19:21:44 WIB 2019
 * REV02 Wed Aug 29 20:54:25 WIB 2018
 * 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 this program 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.
 *
```

## 12-clisvr (02)

```
* 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
*/
```

```
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;
```

## 12-clisvr (03)

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

#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);
    }
    if (strcmp (argv [1], "null") == 0) {
        portno = atoi (argv [3]);
        sockfd = socket (AF_INET, SOCK_STREAM, 0);
        if (sockfd < 0) {
            error ("ERROR opening socket");
        }
        int enable = 1;
        if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR,
            &enable, sizeof(int)) < 0)
            error("setsockopt(SO_REUSEADDR) failed");
    }
```

## 12-clisvr (04)

```
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\n%s[%d]:", getenv ("USER"), sysup);
nn = write (sockfd, buffer, strlen (buffer));
if (nn < 0) {
    error ("ERROR writing to socket");
}
/* End: action */
exit (0);
}

sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0) {
    error ("ERROR opening socket");
}
```



## 12-clisvr (05)

```
int enable = 1;
if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR,
    &enable, sizeof(int)) < 0)
    error("setsockopt(SO_REUSEADDR) failed");
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);
```

## 12-clisvr (06)

```
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\n%s[%d]:\nEndOfMessage!", getenv ("USER"), sysup);
/*End of modifying buffer's message*/
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");
    }
    printf ("%s\n", buffer); // ===== Begin: action
    nn=write(sockfd,buffer,strlen(buffer));
    if (nn < 0) error ("ERROR writing to socket"); // ===== End: action
    else printf ("%s\n", buffer);
    return 0;
}
```

# 12-clisvr (07)

```
root@pamulang1:~# ./12-clisvr 9999 null null
$ host ckilat1.vlsm.org
ckilat1.vlsm.org has address 103.43.44.16
$ ./12-clisvr 9999 null null
From
rms46[16229] to
poor[16245] to
poor[16260]:
EndOfMessage!
$

$ host ckilat2.vlsm.org
ckilat2.vlsm.org has address 103.23.20.185
$ ./12-clisvr 9998 ckilat1.vlsm.org 9999
From
rms46[16229] to
poor[16245]:
EndOfMessage!
$

$ hostname
pamulang1
$ ./12-clisvr null ckilat2.vlsm.org 9998
$ date
Sun May  3 12:17:18 WIB 2020
$
```

Figure: Client Server

# 54-write (01)

```
/*
 * Copyright (C) 2015-2019 Rahmat M. Samik-Ibrahim
 * http://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.
 *
 * TAKE NOTE()
 * O_RDWR open for reading and writing
 * O_CREAT indicates that the call to open() has a mode argument,
 * if the file being opened already exist O_CREAT has no effect
 * if the file being opened does not exist it is created
 * if O_CREAT is specified and the file did not previously exist a successful open
 * () sets the access time, change time, and modification time for the file
 *
 * if successful, dup() returns a new file descriptor
 * if unsuccessful, dup() returns -1 and sets errno to EBADF or EMFILE
 *
 * REV09 Tue Nov 26 11:38:34 WIB 2019
 * REV08 Wed Aug 29 20:55:23 WIB 2018
 * REV07 Thu Oct 5 17:56:09 WIB 2017
 * REV02 Sun Oct 16 20:50:52 WIB 2016
 * START Xxx Apr 25 XX:XX:XX WIB 2015
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
```

## 54-write (02)

```
#define FILE5    "demo-file5.txt"
static char* str1 = "AAAXBBB\n";
static char* str2 = "CCC\n";

void main(void) {
    int fd1, fd2;
    fd1 = open (FILE5, O_RDWR | O_CREAT, 0644);
    fd2 = open (FILE5, O_RDWR | O_CREAT, 0644);
    printf("File Descriptors --- fd1 = %d, fd2 = %d\n", fd1, fd2);
    write(fd1, str1, strlen(str1));
    write(fd2, str2, strlen(str2));
    close(fd1);
    close(fd2);
    printf("See output file %s\n", FILE5);
}
```

```
# #####
```

```
$ ./54-write
```

```
File Descriptors --- fd1 = 3, fd2 = 4
```

```
See output file demo-file5.txt
```

```
$ cat demo-file5.txt
```

```
CCC
```

```
BBB
```

```
$
```

# 55-write (01)

```
/*
 * Copyright (C) 2015-2019 Rahmat M. Samik-Ibrahim
 * http://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.
 *
 * TAKE NOTE (MA)
 * Program ini akan membuat file baru dengan isi
 * buf1 pada 8 char pertama, dan buf2 pada 8 char terakhir
 *
 * Line 31 akan membuat program menulis 8 char
 * dari variabel char buf1 ke file yang didefine pada Line 19
 *
 * Line 35 akan membuat offset menjadi 32,
 * yang maksudnya adalah pointernya lompat ke huruf ke 32
 * Sehingga ketika menulis lg, akan dimulai pada huruf ke 33
 *
 * REV06 Tue Nov 26 11:39:10 WIB 2019
 * REV05 Wed Aug 29 20:55:23 WIB 2018
 * REV04 Wed Oct 18 17:54:25 WIB 2017
 * REV02 Thu Mar 9 21:21:28 WIB 2017
 * START Xxx Apr 25 XX:XX:XX WIB 2015
 * USE "hexdump FILE1"
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

## 54-write (02)

```
#define FILE6    "demo-file6.txt"
char buf1[] = "abcdefgh";
char buf2[] = "ABCDEFGH";
void main(void) {
    int fd;
    fd = creat(FILE6, 0644);
    if (fd < 0) {
        perror("creat error");
        exit(1);
    }
    if (write(fd, buf1, 8) != 8) {
        perror("buf1 write error");
        exit(1);
    } /* offset now = 8 */
    if (lseek(fd, 32, SEEK_SET) == -1) {
        perror("lseek error");
        exit(1);
    } /* offset now = 32 */
    if (write(fd, buf2, 8) != 8) {
        perror("buf2 write error");
        exit(1);
    } /* offset now = 40 */
    close(fd);
    printf("Run: hexdump -c %s\n", FILE6);
}
```

# ###

\$ ./55-write

Run: hexdump -c demo-file6.txt

\$ hexdump -c demo-file6.txt

```
00000000  a  b  c  d  e  f  g  h  \0  \0  \0  \0  \0  \0  \0  \0
00000100  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
00000200  A  B  C  D  E  F  G  H
00000280
```

# 57-dup (01)

```
/*
 * Copyright (C) 2016-2019 Rahmat M. Samik-Ibrahim
 * http://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.
 *
 * TAKE NOTE(TA)
 * O_RDWR open for reading and writing
 * O_CREAT indicates that the call to open() has a mode argument,
 * if the file being opened already exist O_CREAT has no effect
 * if the file being opened does not exist it is created
 * if O_CREAT is specified and the file did not previously exist a successful open
 * () sets the access time, change time, and modification time for the file
 *
 * if successful, dup() returns a new file descriptor
 * if unsuccessful, dup() returns -1 and sets errno to EBADF or EMFILE
 *
 * REV07 Tue Nov 26 11:39:10 WIB 2019
 * START Xxx Apr 25 XX:XX:XX WIB 2015
 * dup(fd) duplicates fd
 * fd2=dup(fd1)  <---> dup2(fd1, fd2)
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
```



## 57-dup (02)

```
#define FILE1 "demo-file7.txt"

static char* str1 = "AAAXBBB\n";
static char* str2 = "CCC\n";

void main(void) {
    int fd1, fd2;
    fd1 = open (FILE1, O_RDWR | O_CREAT, 0644);
    fd2 = dup(fd1);
    printf("File Descriptors --- fd1 = %d, fd2 = %d\n", fd1, fd2);
    write(fd1, str1, strlen(str1));
    write(fd2, str2, strlen(str2));
    close(fd1);
    close(fd2);
    printf("**** Please check file %s ****\n", FILE1);
    printf("**** Compare with 54-write\n");
}
```

```
# #####
$ ./54-write
File Descriptors --- fd1 = 3, fd2 = 4
See output file demo-file5.txt
$ ./57-dup
File Descriptors --- fd1 = 3, fd2 = 4
**** Please check file demo-file7.txt ****
**** Compare with 54-write
$ cat demo-file5.txt
CCC
BBB
$ cat demo-file7.txt
AAAXBBB
CCC
$
```

# 58-dup2 (01)

```
/*
 * Copyright (C) 2015-2019 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software.
 * REV07 Tue May 7 18:46:12 WIB 2019
 * REV04 Thu Mar 9 21:22:36 WIB 2017
 * REV02 Sun Oct 16 20:52:15 WIB 2016
 * START Xxx Apr 25 XX:XX:XX WIB 2015
 *
 * fd2=dup2(fd1, NEWFD)
 *
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
```

## 58-dup2 (02)

```
#define FILE1 "demo-file8.txt"
#define NEWFD 10
static char* str1 = "AAAXBBB\n";
static char* str2 = "CCC\n";

void main(void) {
    int fd1, fd2;
    fd1 = open (FILE1, O_RDWR | O_CREAT, 0644);
    fd2=dup2(fd1, NEWFD);
    printf("File Descriptors --- fd1 = %d, fd2 = %d\n", fd1, fd2);
    write(fd1, str1, strlen(str1));
    write(fd2, str2, strlen(str2));
    close(fd1);
    close(fd2);
    printf("**** Please check file %s *****\n", FILE1);
    printf("**** Compare with 54-write\n");
}
```

```
# #####
```

```
$ ./58-dup2
```

```
File Descriptors --- fd1 = 3, fd2 = 10
**** Please check file demo-file8.txt ****
```

```
**** Compare with 54-write
```

```
$ cat demo-file8.txt
```

```
AAAXBBB
```

```
CCC
```

```
$ cat demo-file5.txt
```

```
CCC
```

```
BBB
```

```
$
```

# 59a-IO

```
$ cat 59a-io.c
// Copyright (C) 2015-2019 Rahmat M. Samik-Ibrahim
// #include ETC ETC

#define FILE1 "59a-io-demo.txt"

void main(void) {
    int fd1, fd2;
    char strvar[100];
    printf ("***** Please check file %s *****\n", FILE1);
    fd1 = open (FILE1, O_RDWR | O_CREAT | O_TRUNC, 0644);
    fd2 = dup(fd1);
    printf(      "AAAAA print to standard output!!\n");
    fprintf(stdout, "BBBBB print to standard output!!\n");
    fprintf(stderr, "CCCCC print to standard error!!!\n");
    sprintf(strvar, "DDDDD print to fd1=%d!!!\n", fd1);
    dprintf(fd1,      "%s", strvar);
    dprintf(fd2,      "EEEEE print to fd2=%d!!!\n", fd2);
    close(fd1);
    close(fd2);
}

# #####
$ ./59a-io
***** Please check file 59a-io-demo.txt *****
AAAAA print to standard output!!
BBBBB print to standard output!!
CCCCC print to standard error!!!
$ cat 59a-io-demo.txt
DDDDD print to fd1=3!!!
EEEEE print to fd2=4!!!
$
```

```
// Copyright (C) 2015-2019 Rahmat M. Samik-Ibrahim
// #include ETC ETC
#define FILE1 "59b-io-demo.txt"

void main(void) {
    int fd1, fd2;
    char strvar[100];
    printf ("***** Please check file %s *****\n", FILE1);

    close(STDERR_FILENO);

    fd1 = open (FILE1, O_RDWR | O_CREAT | O_TRUNC, 0644);
    fd2 = dup(fd1);
    printf("AAAAA print to standard output!!\n");
    fprintf(stdout, "BBBBB print to standard output!!\n");
    fprintf(stderr, "CCCCC print to standard error!!!\n");
    sprintf(strvar, "DDDDD print to fd1=%d!!!\n", fd1);
    dprintf(fd1, "%s", strvar);
    dprintf(fd2, "EEEEEE print to fd2=%d!!!\n", fd2);
    close(fd1);
    close(fd2);
}

# #####
$
$ ./59b-io
***** Please check file 59b-io-demo.txt *****
AAAAA print to standard output!!
BBBBB print to standard output!!
$ cat 59b-io-demo.txt
CCCCC print to standard error!!!
DDDDD print to fd1=2!!!
EEEEEE print to fd2=3!!!
$
```

# 59c-IO

```
// Copyright (C) 2015-2019 Rahmat M. Samik-Ibrahim
// #include ETC ETC
#define FILE1 "59c-io-demo.txt"

void main(void) {
    int fd1, fd2;
    char strvar[100];
    printf ("***** Please check file %s ***** \n", FILE1);

    close(STDERR_FILENO);
    close(STDOUT_FILENO);

    fd1 = open (FILE1, O_RDWR | O_CREAT | O_TRUNC, 0644);
    fd2 = dup(fd1);
    printf("AAAAA print to standard output!!\n");
    fprintf(stdout, "BBBBB print to standard output!!\n");
    fprintf(stderr, "CCCCC print to standard error!!!\n");
    sprintf(strvar, "DDDDD print to fd1=%d!!!\n", fd1);
    dprintf(fd1, "%s", strvar);
    dprintf(fd2, "EEEEE print to fd2=%d!!!\n", fd2);
    close(fd1);
    close(fd2);
}

# #####
$ ./59c-io
***** Please check file 59c-io-demo.txt *****
$ cat 59c-io-demo.txt
AAAAA print to standard output!!
BBBBB print to standard output!!
CCCCC print to standard error!!!
DDDDD print to fd1=1!!!
EEEEE print to fd2=2!!!
```

# 70-os161

```
// Copyright (C) 2015-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC

#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

#define FILE "70-os161-demo.txt"

char *string = "ABCD\n";
void main(void) {
    int fileDescriptor;
    printf("See also file %s\n", FILE);
    close(STDOUT_FILENO);
    fileDescriptor = open (FILE, O_RDWR|O_CREAT|O_TRUNC, 0644);
    printf ( "%s", string);
    write(fileDescriptor, string, strlen(string));
}

# #####

$ ./70-os161
See also file 70-os161-demo.txt

$ cat 70-os161-demo.txt
ABCD
ABCD
```

# 71-os171

```
// Copyright (C) 2017-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC

static char* str1 = "AABB\n";
static char* str2 = "CCDD\n";
static char* str3 = "EEFF\n";

void main(void) {
    int fd1, fd2, fd3;
    printf("See also file %s\n", FILE);
    /* STDIN=0, STDOUT=1, STDERR=2, therefore
       fd1, fd2, fd3 will be 3, 4, and 5 */
    fd1 = open (FILE, O_TRUNC | O_RDWR | O_CREAT, 0644);
    fd2 = open (FILE, O_TRUNC | O_RDWR | O_CREAT, 0644);
    fd3 = dup(fd2);
    printf("fd1 = %d, fd2 = %d, fd3 = %d\n", fd1, fd2, fd3);
    write(fd1, str1, strlen(str1));
    write(fd2, str2, strlen(str2));
    write(fd3, str3, strlen(str3));
    close(fd1);
    close(fd2);
    close(fd3);
}

# #####

$ ./71-os171
See also file 71-os171-demo.txt
fd1 = 3, fd2 = 4, fd3 = 5

$ cat 71-os171-demo.txt
CCDD
EEFF
```



```
// Copyright (C) 2017-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC
#define FILE "72-os172-demo.txt"
void main(void) {
    int fd1, fd2;
    printf("See also file %s\n", FILE);
    fd1 = open (FILE, O_RDWR | O_CREAT | O_TRUNC, 0644);
    fd2 = dup(fd1);
    write (fd1, "0123456789\n", 5);
    write (fd2, "abcdefghij\n", 5);
    close(fd1);
    close(fd2);
}
# #####
$X$ ./72-os172
See also file 72-os172-demo.txt
$X$ cat 72-os172-demo.txt
01234abcde$X$
```

# 73-os181

```
// Copyright (C) 2017-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC
```

```
#define FLAGS O_RDWR|O_TRUNC|O_CREAT
#define FILE "73-os181-demo.txt"
```

```
static char* str1 = "AAAAAAAAAAAA";
static char* str2 = "BBBBB";
```

```
void main(void) {
    int fd1, fd2, fd3;
    printf("See also file %s\n", FILE);
    /* STDIN=0, STDOUT=1, STDERR=2,
       fd1,fd2,fd3 will be 3,4,and 5 */
    fd1=open(FILE, FLAGS, 0644);
    fd2=open(FILE, FLAGS, 0644);
    fd3=dup(fd1);
    dprintf(fd1,"%s", str1);
    dprintf(fd2,"X%dX%dX%dX",fd1,fd2,fd3);
    dprintf(fd3,"%s", str2);
    close(fd1);
    close(fd2);
    close(fd3);
}
# #####
```

```
$X$ ./73-os181
See also file 73-os181-demo.txt
```

```
$X$ cat 73-os181-demo.txt
X3X4X5XAAABBBBB$X$
```

# 74-os182

```
// Copyright (C) 2018-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC
#define FLAGS O_RDWR|O_CREAT|O_TRUNC
#define MODES 0644
#define FILE3 "74-os182-demo3.txt"
#define FILE4 "74-os182-demo4.txt"
void main(void) {
    printf("See %s and %s\n", FILE3, FILE4);
    int fd3 = open (FILE3,FLAGS,MODES);
    int fd4 = open (FILE4,FLAGS,MODES);
    dprintf(fd3, "fd%d\n", fd3);
    dprintf(fd4, "fd%d\n", fd4);
    close(STDOUT_FILENO); // STDOUT = 1
    int fd1 = dup(fd3);
    close(STDERR_FILENO); // STDERR = 2
    int fd2 = dup(fd4);
    dprintf(fd1, "fd%d\n", fd1);
    dprintf(fd2, "fd%d\n", fd2);
    close (fd1);
    close (fd2);
    close (fd3);
    close (fd4);
}
# #####
$ ./74-os182
See 74-os182-demo3.txt and 74-os182-demo4.txt
$ cat 74-os182-demo3.txt
fd3
fd1
$ cat 74-os182-demo4.txt
fd4
fd2
$
```

# 75-os191

```
// Copyright (C) 2019-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC
```

```
#define FILE      "75-os191-demo.txt"
#define STRING1   "AAABBBCCC"
#define STRING2   "DDDEEEFFF"
#define STRING3   "GGGHHHHIII"
```

```
void main(void) {
    printf("See %s\n", FILE);
    int fd1=open(FILE,
        O_CREAT|O_TRUNC|O_RDWR, 0644);
    int fd2=open(FILE,
        O_CREAT|O_TRUNC|O_RDWR, 0644);
    int fd3=open(FILE,
        O_CREAT|O_TRUNC|O_RDWR, 0644);
    write (fd1,STRING1, 9);
    write (fd2,STRING2, 6);
    write (fd3,STRING3, 3);
    close(fd1);
    close(fd2);
    close(fd3);
}
```

```
### #####
```

```
$X$ ./75-os191
```

```
See 75-os191-demo.txt
```

```
$X$ cat 75-os191-demo.txt
```

```
GGGEEEECCC$X$
```

# 76-os192

```
// Copyright (C) 2019-2020 Rahmat M. Samik-Ibrahim
// #include ETC ETC
```

```
#define FILE      "76-os192-demo.txt"

void main(void) {
    printf("See %s\n", FILE);
    printf ("OUT=%d\n", STDOUT_FILENO);
    close(STDOUT_FILENO);
    int fd1 = open (FILE, O_RDWR |
                    O_CREAT | O_TRUNC, 0644);
    int fd2 = dup2(fd1, 9);
    printf(        "A\n");
    fprintf(stdout, "B\n");
    dprintf(fd2,    "fd1=%d\nfd2=%d\n",
                    fd1, fd2);
}
```

```
# #####
```

```
$ ./76-os192
See 76-os192-demo.txt
OUT=1
```

```
$ cat 76-os192-demo.txt
A
B
fd1=1
fd2=9
$
```

## • 18 Knowledge Areas

AL - Algorithms and Complexity	AR - Architecture and Organization
CN - Computational Science	DS - Discrete Structures
GV - Graphics and Visualization	HCI - Human-Computer Interaction
IAS - Information Assurance and Security	IM - Information Management
IS - Intelligent Systems	NC - Networking and Communications
OS - Operating Systems	PBD - Platform-based Development
PD - Parallel and Distributed Computing	PL - Programming Languages
SDF - Software Development Fundamentals	SE - Software Engineering
SF - Systems Fundamentals	SP - Social Issues and Professional Practice

## • OS - Operating Systems (IEEE/ACM 2013)

- OS/Overview of Operating Systems (T1:2)
- OS/Operating System Principles (T1:2)
- OS/Concurrency (T2:3)
- OS/Scheduling and Dispatch (T2:3)
- OS/Memory Management (T2:3)
- OS/Security and Protection (T2:2)
- OS(Electives): Virtual Machines, Device Management, File Systems, Real Time and Embedded Systems, Fault Tolerance, System Performance Evaluation.

## Week 10: Check List (Deadline: Monday, 07-Dec-2020).

- ☐ **Starting Point:** <https://os.vlsm.org/>
- ☐ Week 10: Assignment (more details in **os10.pdf**).
  - ① Read: (OSC10 chapter 12)
  - ② TBA.
- ☐ The "Assignment Day" is every Thursday morning.
- ☐ This page is <https://os.vlsm.org/Slides/check10.pdf>.

# The End

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