# CSGE602055 Operating Systems
# CSF2600505 Sistem Operasi
# Week 10: I/O & Programming

Rahmat M. Samik-Ibrahim

University of Indonesia

REV180 16-Jan-2018

# 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, | Ch. 16, 17 |
|         |                      | & C-language | |
| 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 | Concurency: 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 |
| Extra | 27 Jun 2019 | Extra assignment confirmation | subject to change. |

# The Weekly Check List

- ☐ **Resources**: `https://os.vlsm.org/`
  - ☐ **(THIS) Slides** — `https://github.com/UI-FASILKOM-OS/` `SistemOperasi/tree/master/pdf/`
  - ☐ **Demos** — `https://github.com/UI-FASILKOM-OS/` `SistemOperasi/tree/master/demos/`
  - ☐ **Extra** — `BADAK.cs.ui.ac.id:///extra/`
  - ☐ **Problems** — `rms46.vlsm.org/2/195.pdf`, `196.pdf`, ..., `205.pdf`
- ☐ **Text Book**: any recent/decent OS book. Eg. (**OSC10**) Silberschatz et. al.: **Operating System Concepts**, $10^{th}$ Edition, 2018.
- ☐ Encode your **QRC** with size upto 7cm x 7cm (ca. 400x400 pixels): **"OS182 CLASS ID SSO-ACCOUNT Your-Full-Name"**
- ☐ For **Week 00**, send your **embedded** QRC **before the** $2^{nd}$ **lecture** `mailto:operatingsystems@vlsm.org` With Subject: OS182 CLASS ID SSO-ACCOUNT Your-Full-Name
- ☐ Write your Memo (with QRC) **every week**.
- ☐ Login to `badak.cs.ui.ac.id` via `kawung.cs.ui.ac.id` for at least **10 minutes** every week. Copy the weekly demo files to your own home directory.
  Eg. (Week00): `cp -r /extra/Week00/W00-demos/ W00-demos/`

# Agenda

# Agenda (2)

# Week 10 I/O & Programming: Topics[1]

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

---

[1]Source: ACM IEEE CS Curricula 2013

# Week 10 I/O & Programming: Learning Outcomes[1]

- 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 (2)

- 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?
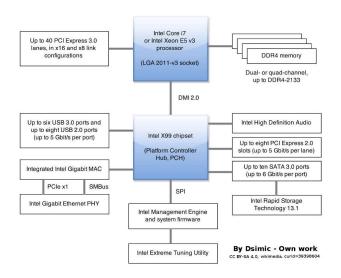  - Controller?
  - HW?

# PCH: Platform Controller Hub



Figure: PCH: Platform Controller Hub

# Some Terms

- PCH: Platform Controller Hub
- PCIe: Peripheral Component Interconnect Express — 32 bits for (16 * 1x or 8 * 2x or 4 * 4x or 2 * 8x or 1 * 16x) * (2 direction) lanes.
- DMI: Direct Media Interface. Eg. DMI 2.0 (2 GB/s; 4x)
- GT/s: GigaTransfers per second
- 1 KB (KiloByte) = 1000 bytes — 1 KiB (Kibibyte) = 1024 bytes[1]
- 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.
- DDR4 SDRAM: 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,

---

[1]In IT tradition; 1 KB = 1024 bytes

# Sockets

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

# Sockets

- 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);
}
```

```
int main(int argc, char *argv[]) {
    char      buffer[256];
    int       clilen, newsockfd, nn, portno, sockfd;
    sockadin  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, (sockad*) &serv_addr, sizeof(serv_addr))< 0)
        error("ERROR on binding");
    listen(sockfd, 5);
    clilen = sizeof(cli_addr);
    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;
}
```

```c
/*
 * (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);
}
```

```
    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;
}
```

```
>>>>> $ 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.
 */
```

```
/*
 * 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
 */
```

```
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);
}
```

# 12-clisvr (4)

```c
#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];
   char* 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");
    }
    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);
}
```

```
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*/
```

```
    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;
}
```

```
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 >>
```

```
#include <stdio.h>

void main (void) {
    int cc = getchar();
    putchar(cc);
    putchar('\n');
}

>>>>> $ 50-get-put
x
x
>>>>> $ 50-get-put
abcde
a

=======================================
#include <stdio.h>

void main (void) {
    int cc;
    while((cc = getchar()) != EOF) {
        putchar(cc);
    }
}

>>>>> $ 51-get-put-loop
xxxx
xxxx
```

```
* === umask() ===
* int open(const char* pathname, int flags, mode_t mode);
* === FLAGS: ===
* O_RDONLY     Open the file so that it is read only.
* O_WRONLY     Open the file so that it is write only.
* O_RDWR       Open the file so that it can be read from and written to.
* O_APPEND     Append new information to the end of the file.
* O_TRUNC      Initially clear all data from the file.
* O_CREAT      If the file does not exist, create it.
               You must include the third parameter.
* O_EXCL       With O_CREAT: exists, the call will fail.
* === MODE ===
* S_IRWXU  00700 user (file  owner) has read, write and execute permission
* S_IRUSR  00400 user has read permission
* S_IWUSR  00200 user has write permission
* S_IXUSR  00100 user has execute permission
*
* S_IRWXG  00070 group has read, write and execute permission
* S_IRGRP  00040 group has read permission
* S_IWGRP  00020 group has write permission
* S_IXGRP  00010 group has execute permission
*
* S_IRWXO  00007 others have read, write and execute permission
* S_IROTH  00004 others have read permission
* S_IWOTH  00002 others have write permission
* S_IXOTH  00001 others have execute permission
```

```c
#define FILE1 "demo-file1.txt"
#define FILE2 "demo-file2.txt"
#define FILE3 "demo-file3.txt"

#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
void main(void) {
    char*  file1=FILE1;
    char*  file2=FILE2;
    char*  file3=FILE3;

    int fd; /* to hold a file descriptor */
    /* umask(0);    ****************** */
    fd = open (file1, O_CREAT | O_RDWR, S_IRWXU);
    close(fd);
    fd = open (file2, O_CREAT | O_RDWR, S_IRWXU|S_IRGRP|S_IWGRP|S_IROTH);
    close(fd);
    fd = open (file3, O_CREAT | O_RDWR, 0711);
    close(fd);
    fd = open (file3, O_CREAT | O_RDWR, 0700);
    close(fd);
}

>>>>> $ ls -al demo-file[234].txt
-rwxr--r-- 1 demo demo  0 Oct  5 17:49 demo-file2.txt
-rwx--x--x 1 demo demo  0 Oct  5 17:49 demo-file3.txt
-rw-r--r-- 1 demo demo 75 Oct  5 17:49 demo-file4.txt
>>>>> $
```

```
#define FILE4 "demo-file4.txt"
#include <stdio.h>
#include <stdlib.h>

void main(void) {
    FILE* fp;
    int   cc;

    printf ("*** Open and listing file %s ***\n\n", FILE4);
    if ((fp=fopen(FILE4, "r")) == NULL) {
        printf("fopen error...\n");
        exit(1);
    }
    while((cc=fgetc(fp)) != EOF) {
        printf("%c", cc);
    }
    printf("\n");
    fclose(fp);
}

*** Open and listing file demo-file4.txt ***

Line 1: Blah Blah Blah 1
Line 2: Blah Blah Blah 2
Line 3: Blah Blah Blah 3
```

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

#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);
}


#######################################
File Descriptors --- fd1 = 3, fd2 = 4
See output file demo-file5.txt

#######################################
demo-file5.txt:
CCC
BBB
```

# 55-write

```c
#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);
}
```

```
>>>>> $ hexdump -c demo-file6.txt
0000000    a   b   c   d   e   f   g   h  \0  \0  \0  \0  \0  \0  \0  \0
0000010   \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020    A   B   C   D   E   F   G   H
```

```c
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#define  BUF_SIZE 16

void main(int argc, char* argv[])
{
   int          fdread, fdwrite;
   unsigned int total_bytes = 0;
   ssize_t      nbytes_read, nbytes_write;
   char buf[BUF_SIZE];
   if (argc != 3) {
      printf("Usage: %s source destination\n",
      argv[0]);
      exit(1);
   }
   fdread = open(argv[1], O_RDONLY);
   if (fdread < 0) {
      perror("Failed to open source file");
      exit(1);
   }
   fdwrite = creat(argv[2], S_IRWXU);
   if (fdwrite < 0) {
      perror("Failed to open destination file");
      exit(1);
   }
```

```
   do {
      nbytes_read = read(fdread, buf, BUF_SIZE);
      if (nbytes_read < 0) {
         perror("Failed to read from file");
         exit(1);
      }
      nbytes_write = write(fdwrite, buf, nbytes_read);
      if (nbytes_write < 0) {
         perror("Failed to write to file");
         exit(1);
      }
   } while (nbytes_read > 0);
   close(fdread);
   close(fdwrite);
   exit(0);
}

#########################################################

>>>>> $ ./56-copy demo-file4.txt demo-copy.txt
>>>>> $ ls -al demo-file4.txt demo-copy.txt
-rwx------ 1 demo demo 75 Oct  5 18:12 demo-copy.txt
-rw-r--r-- 1 demo demo 75 Oct  5 17:49 demo-file4.txt
>>>>> $
```

# 57-dup

```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define FILE1 "demo-file7.txt"

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

Coming Soon
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");
}
#############################################
>>>>> $ 57-dup
File Descriptors --- fd1 = 3, fd2 = 4
**** Please check file demo-file7.txt *****
**** Compare with 54-write
>>>>> $ cat demo-file7.txt
AAAXBBB
CCC
>>>>> $
```

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#define FILE1 "demo-file8.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);
   dup2(fd1, fd2);
   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 = 0
**** Please check file demo-file8.txt ****
**** Compare with 54-write
>>>>> $ cat demo-file8.txt
AAAXBBB
CCC
>>>>> $
```

```c
#include <stdio.h>
#include .....
#define FILE1 "demo-file9.txt"

void main(void) {
    int fd1, fd2;
    char strvar[100];
    printf ("***** Please check file %s ***** *****\n", FILE1);
/* BLOCK ***************
    close(STDERR_FILENO);
    close(STDOUT_FILENO);
    BLOCK *************** */
    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);
}
>>>>> $ 59-io ; echo "^^^^^";cat demo-file9.txt
***** Please check file demo-file9.txt ***** *****
AAAAA print to standard output!!
BBBBB print to standard output!!
CCCCC print to standard error!!!
^^^^^
DDDDD print to fd1=3!!!
EEEEE print to fd2=4!!!
```

```c
#include <stdio.h>
#include .....
#define FILE1 "demo-file9.txt"

void main(void) {
   int fd1, fd2;
   char strvar[100];
   printf ("***** Please check file %s ***** *****\n", FILE1);
   close(STDERR_FILENO);
/* BLOCK ***************
   close(STDOUT_FILENO);
   BLOCK *************** */
   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);
}
>>>>> $ 59-io ; echo "^^^^^";cat demo-file9.txt
***** Please check file demo-file9.txt ***** *****
AAAAA print to standard output!!
BBBBB print to standard output!!
^^^^^
CCCCC print to standard error!!!
DDDDD print to fd1=2!!!
EEEEE print to fd2=3!!!
```

```
#include <stdio.h>
#include .....
#define FILE1 "demo-file9.txt"

void main(void) {
    int fd1, fd2;
    char strvar[100];
    printf ("***** Please check file %s ***** *****\n", FILE1);
    close(STDERR_FILENO);
    close(STDOUT_FILENO);
/* BLOCK ***************
   BLOCK *************** */
    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);
}
>>>>> $ 59-io ; echo "^^^^^";cat demo-file9.txt
***** Please check file demo-file9.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!!!
```

```
#define FILE1 "demo-fileA.txt"
#define  OLOOP 10
#define  ILOOP 3650
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
#include <fcntl.h>
#include <dirent.h>
void rwfile (char* fname);
void dirfile(char* dname);
void error  (char* msg);
/* MAIN ========================== */
void main(void) {
    printf("Listing current directory...\n");
    dirfile(".");
    printf("Testing read-write speed...\n");
    rwfile(FILE1);
}
/* DIRFILE ====================== */
void dirfile(char* dname) {
    DIR*           ddir;
    struct dirent* dp;
    printf("      ");
    ddir = opendir(dname);
    if (ddir != NULL) {
        while ((dp=readdir(ddir))!= NULL)
            printf("%s  ", dp->d_name);
        closedir(ddir); }
    printf("\n\n"); }
```

```c
/* ERROR ========================= */
void error(char* msg){
    perror(msg);
    exit(0); }
/* RWFILE ========================= */
void rwfile(char* fname) {
    time_t tt;
    int    fd, ii, jj;
    char   buf[] = "Achtung... Achtung... AAAA BBBB CCCC DDDD\n";
    time(&tt);
    for (ii=0;ii<OLOOP;ii++) {
        if ((fd=creat(fname,00644)) < 0 )
            error("RWFILE: can not create file\n");
        for (jj=0;jj<ILOOP;jj++) {
            write(fd,buf,sizeof(buf)-1);
            fsync(fd); }
        close(fd);
        putchar('.');
        fflush(NULL); }
    tt=time(NULL)-tt;
    putchar('\n');
    printf("Total time: %d seconds\n", (int) tt);
}
#####################
>>>>> $ time 60-readwrite
Listing current directory...
      .shsh  52-open-close.c  demo-file4.txt  02-pointers.c ...
Testing read-write speed...
..........
Total time: 10 seconds
real    0m9.998s ----- user   0m0.024s ----- sys    0m0.576s
```

# IEEE/ACM 2013

- 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.

# The End

☐ This is the end of the presentation.

☒ This is the end of the presentation.

- This is the end of the presentation.