



# CS415: Systems Programming

## File Manipulation using System Calls

# How can...

- How can the OS run the same needed programs every time it runs?
- How can the initially executed programs know what to do next?
- How to use the needed arguments every time the OS runs?

There are many side text files used to store needed arguments, calls, boot information. Such files are open, read, closed at time of starting up

# Unix file I/O

Performed mostly using 7 system calls:

- open
- close
- read
- write
- lseek
- dup, dup2

# The open system call

- A file can be opened using the “open” system call as follows.

`#include <sys/file.h> // can be replaced by <fcntl.h>`

`int open(char* filename, int flags, int mode);`

- The above code opens the filename for **reading** or **writing** as specified by the 2<sup>nd</sup> argument.
- It returns an **integer** which refers or describes that file (known as **file descriptor**, will be discussed later).
- If the file does not exist, then the “open” system call creates the file with the given name.

# The open system call

```
#include <sys/file.h> // can be replaced by <fcntl.h>
```

```
int open(char* filename, int flags, int mode);
```

- **filename:** A string that represents filename
- **flags:** An integer code describing the access (Next slide)
- **mode:** The file protection mode (for opening files keep 0)

# The open system call

`#include <sys/file.h> // can be replaced by <fcntl.h>`

`int open(char* filename, int flags, int mode);`

- `O_RDONLY` -- opens a file for reading only
- `O_WRONLY` -- opens a file for writing only
- `O_RDWR` -- opens a file for reading and writing
- `O_APPEND` -- opens a file for appending
- `O_CREAT` -- creates a file if it does not exist
- `O_TRUNC` -- truncates size to 0

# Create Modes – Octal Values

| Permission | Values |
|------------|--------|
| Read       | 4      |
| Write      | 2      |
| execute    | 1      |

Special    User    Group    Others



```
open("hello.txt", O_RDWR | O_CREAT, 0755);
```

# Create Modes (examples)

| Special feature | User   | Group  | Other  |
|-----------------|--------|--------|--------|
| 3 bits          | 3 bits | 3 bits | 3 bits |

R: 4      W: 2      X: 1

7 = RWX

6 = RW

5 = RX

3 = WX



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(){
9     char* arg_list[]={ "ls", "-l", ".", NULL};
10    int pid = fork();
11
12    if(pid == 0) execvp(arg_list[0],arg_list);
13    else if(pid > 0){
14        wait(NULL);
15
16        int fd;
17        fd = open("newFile.txt", O_RDWR | O_CREAT, 0750);
18        if(fd < 0){
19            printf("Error\n");
20            exit(1);
21        }
22        printf("\n*****\n");
23        printf("File Created/Opened Successfully!\n");
24        printf("*****\n\n");
25
26        pid = fork();
27
28        if(pid == 0) execvp(arg_list[0],arg_list);
29        else if(pid > 0){
30            wait(NULL);
31        }
32    }
33
34    return 0;
35 }

```

```

total 16
-rwxr-xr-x 1 runner runner 8552 Mar  7 23:44 main
-rw-r--r-- 1 runner runner  783 Mar  7 23:44 main.c

*****
File Created/Opened Successfully!
*****

total 16
-rwxr-xr-x 1 runner runner 8552 Mar  7 23:44 main
-rw-r--r-- 1 runner runner  783 Mar  7 23:44 main.c
-rwxr-x--- 1 runner runner    0 Mar  7 23:44 newFile.txt

```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(){
9     char* arg_list[]={ "ls", "-l", ".", NULL};
10    int pid = fork();
11
12    if(pid == 0) execvp(arg_list[0],arg_list);
13    else if(pid > 0){
14        wait(NULL);
15
16        int fd;
17        fd = open("newFile2.txt", O_RDWR | O_CREAT, 0000);
18        if(fd < 0){
19            printf("Error\n");
20            exit(1);
21        }
22        printf("\n*****\n");
23        printf("File Created/Opened Successfully!\n");
24        printf("*****\n\n");
25
26        pid = fork();
27
28        if(pid == 0) execvp(arg_list[0],arg_list);
29        else if(pid > 0){
30            wait(NULL);
31        }
32    }
33
34    return 0;
35 }

```

```

total 16
-rwxr-xr-x 1 runner runner 8552 Mar  7 23:54 main
-rw-r--r-- 1 runner runner  701 Mar  7 23:54 main.c
-rwxr-x--- 1 runner runner    0 Mar  7 23:44 newFile.txt

```

```

*****
File Created/Opened Successfully!
*****

```

```

total 16
-rwxr-xr-x 1 runner runner 8552 Mar  7 23:54 main
-rw-r--r-- 1 runner runner  701 Mar  7 23:54 main.c
----- 1 runner runner    0 Mar  7 23:54 newFile2.txt
-rwxr-x--- 1 runner runner    0 Mar  7 23:44 newFile.txt

```

# Unix Permission Calculator (<http://permissions-calculator.org/>)

← → ↻ ⓘ Not secure | permissions-calculator.org

## Unix Permissions Calculator

Octal

Decode Octal

Symbolic

Info

Code Examples

### Permission bits

Select the permissions you require below. The tool will provide you with an octal code that corresponds to these permissions which can then be applied to relevant directories and files with `chmod`.

#### Special

- ☐ setuid ⓘ
- ☐ setgid ⓘ
- ☐ Sticky bit ⓘ

#### User

- ☒ Read
- ☒ Write
- ☒ Execute

#### Group

- ☒ Read
- ☐ Write
- ☐ Execute

#### Other

- ☒ Read
- ☐ Write
- ☐ Execute

### Absolute Notation (octal)

0744

*e.g: `chmod 0744 <path-to-file>`*

# Special Features of File Creating Modes

  
`open("hello.txt", O_RDWR | O_CREAT, 0755);`

## Set SUID: 4

- 0 - No options set
- 1 - Sticky Bit set
- 2 - SGID set
- 3 - Sticky Bit and SGID Set
- 4 - SUID set
- 5 - Sticky Bit and SUID Set
- 6 - SGID and SUID set
- 7 - Sticky Bit GID and UID Set

## Set SGID: 2    Set Sticky Bit: 1

a Sticky Bit lets only the owner of the file/directory delete or rename the file/director. However, it should be noted that root user is still able to delete or rename the file/directory as well.

SGID (Set-Group Identification) enables other users to inherit the effective GID (group identifier) of a group owner.

SUID (Set User Identification) enables other users to run the file with the effective permission of the file owner.

# The close system call

`#include <sys/file.h> // can be replaced by <fcntl.h>`

`int close (int fd);`

- Description: closes file
- **Fd is the file descriptor**
- Return -1 if error, 0 otherwise

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5
6  int main(){
7      int fd;
8      fd = open("newFile3.txt", O_RDWR | O_CREAT, 0700);
9
10     if(fd < 0){
11         printf("Error when opening/creating file\n");
12         exit(1);
13     }
14
15     printf("\n*****\n");
16     printf("File Created/Opened Successfully!\n");
17     printf("*****\n\n");
18
19     int x;
20     x = close(fd);
21
22     if(x < 0){
23         printf("Error when closing file\n");
24         exit(1);
25     }
26
27     printf("\n*****\n");
28     printf("File closed Successfully!\n");
29     printf("*****\n\n");
30
31     return 0;
32 }

```

```

*****
File Created/Opened Successfully!
*****

*****
File closed Successfully!
*****

```

# The read system call

```
#include <sys/types.h> // or #include <unistd.h>
```

```
int read(int fd, void *buffer, size_t bytes);
```

- Fd is the file descriptor.
  - Buffer is an address of a memory area into which the data is read.
  - bytes is the maximum amount of data to read from the stream.
- 
- The return value is the actual amount of data read from the file

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7
8  int main(){
9      int fd;
10     fd = open("random_text.txt", O_RDONLY, 0777);
11     if(fd < 0){
12         printf("Error when opening/creating file\n");
13         exit(1);
14     }
15     char buffer[10];
16     int readCount;
17     readCount = read(fd, buffer, 10);
18     if(readCount < 0){
19         printf("Error when reading file\n");
20         exit(1);
21     }
22     else{
23         printf("%d\n", readCount);
24         printf("%s\n", buffer);
25     }
26     if(close(fd) < 0){
27         printf("Error when closing file\n");
28         exit(1);
29     }
30     return 0;
31 }

```

random\_text.txt

The Department of Computer Science at GJU is a part of Electrical Engineering and Information Technology School (SEEIT).

```

./main
10
The Depart

```



```

3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7
8  int main(){
9      int fd;
10     fd = open("random_text.txt", O_RDONLY, 0777);
11     if(fd < 0){
12         printf("Error when opening/creating file\n");
13         exit(1);
14     }
15     char buffer[10];
16     int readCount;
17     readCount = read(fd, buffer, 10);
18     if(readCount < 0){
19         printf("Error when reading file\n");
20         exit(1);
21     }
22     else{
23         while(readCount>0){
24             printf("%s\n", buffer);
25             readCount = read(fd, buffer, 10);
26         }
27     }
28     if(close(fd) < 0){
29         printf("Error when closing file\n");
30         exit(1);
31     }
32     return 0;
33 }
34

```

random\_text.txt

The Department of Computer Science at GJU is a part of Electrical Engineering and Information Technology School (SEEIT).

```

> ./main
The Depart
ment of Co
mputer Sci
ence at GJ
U is a par
t of Elect
rical Engi
neering an
d Informat
ion Techno
logy Schoo
l (SEEIT).
>

```

# The write system call

```
#include <sys/types.h>
```

```
write(int fd, void *buffer, size_t bytes);
```

- **Fd is the file descriptor.**
- **buffer is an address of an area of memory where data is to be written out.**
- **bytes is the amount of data to copy.**
- The return value is the actual amount of data written, if this differs from bytes then something may be wrong.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <string.h>
8
9  int main(){
10     int fd;
11     fd = open("myFile.txt", O_RDWR | O_CREAT, 0755);
12     if(fd < 0){
13         printf("Error when opening/creating file\n"); exit(1);
14     }
15
16     char *writeThis = "I am writing to the file";
17     int writeSz;
18     writeSz = write(fd, writeThis, strlen(writeThis));
19
20     if(writeSz < 0){
21         printf("Error when writing file\n"); exit(1);
22     }
23
24     system ("cat myFile.txt");
25     printf("\n");
26
27     if(close(fd) < 0){
28         printf("Error when closing file\n");
29         exit(1);
30     }
31     return 0;
32 }
```

```
./main
I am writing to the file

```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 #include <string.h>
8 int main(){
9     int fd, fd2, numberOfReads; char buffer[10];
10    fd = open("random_text.txt", O_RDWR | O_CREAT, 0755);
11    fd2 = open("modified_text.txt", O_RDWR | O_CREAT, 0755);
12    if(fd < 0 || fd2 < 0){
13        printf("Error when opening/creating file\n"); exit(1);
14    }
15
16    int readCount = read(fd, buffer, 10);
17    while(readCount > 0){
18        if(numberOfReads % 2 == 0){
19            char *writeThis = "Even Read\n";
20            write(fd2, writeThis, strlen(writeThis));
21        }
22        else{
23            write(fd2, buffer, 10);
24            write(fd2, "\n", 1);
25        }
26        readCount = read(fd, buffer, 10);
27        numberOfReads++;
28    }
29
30    if(close(fd) < 0 || close(fd2) < 0){
31        printf("Error when closing file\n"); exit(1);
32    }
33    system("cat modified_text.txt");
34    return 0;
35 }
```

```
➤ ./main
Even Read
ment of Co
Even Read
ence at GJ
Even Read
t of Elect
Even Read
neering an
Even Read
ion Techno
Even Read
l (SEEIT).
➤
```

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <string.h>
8
9  typedef struct{
10     int left;
11     int right;
12 } pair_t;
13
14 int main(int argc, char * argv[]){
15
16     pair_t p;
17     p.left = 10;
18     p.right = 20;
19
20     int fd;
21     fd = open("pair.txt", O_RDWR | O_CREAT, 0755);
22     if(fd < 0){
23         printf("Error when opening/creating file\n"); exit(1);
24     }
25
26     write(fd, &p, sizeof(pair_t));
27
28     system("cat pair.txt");
29     system("ls -l .");
30     return 0;
31 }

```

```
> ./main
```

```
total 20
```

```

-rwxr-xr-x 1 runner runner 8504 Mar  8 01:17 main
-rw-r--r-- 1 runner runner  514 Mar  8 01:17 main.c
-rwxr-xr-x 1 runner runner    8 Mar  8 01:17 pair.txt

```

```
> 
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <sys/types.h>
6  #include <sys/wait.h>
7  #include <string.h>
8
9  typedef struct{
10     int left;
11     int right;
12 } pair_t;
13
14 int main(int argc, char * argv[]){
15
16     int fd;
17     fd = open("pair.txt", O_RDWR | O_CREAT, 0755);
18
19     if(fd < 0){
20         printf("Error when opening/creating file\n"); exit(1);
21     }
22
23     pair_t p;
24     read(fd, &p, sizeof(pair_t));
25
26     printf("left: %d right: %d\n", p.left, p.right);
27
28     return 0;
29 }
```

```
> ./main
left: 10 right: 20
> 
```

# lseek System Call

- Currently, we are able to read/write a file from the beginning of the file. In order to read/write a file from anywhere in a file (e.g., from the beginning, the middle, or at the end), you can use the “lseek” function
- lseek(fd, offset, whence)
  - Description: reposition the read/write offset
  - whence
    - SEEK\_SET: The file cursor is set to offset bytes from the beginning of the file.
    - SEEK\_CUR: The file cursor is set to its current location plus offset bytes.
    - SEEK\_END: The file cursor is set to the size of the file plus offset bytes.
- The return value is the new position of the file cursor (after moving it) from the beginning of the file. If the return value is -1, then there was an error when moving the file cursor.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <fcntl.h>
5  #include <string.h>
6
7  int main(){
8
9      printf("*****Contents of random_text.txt*****\n");
10     system("cat random_text.txt");
11     printf("\n*****Contents of random_text.txt*****\n");
12
13     int fd;
14     fd = open("random_text.txt", O_RDWR | O_CREAT, 0755);
15
16     if(fd < 0){
17         printf("Error when opening/creating file\n");
18         exit(1);
19     }
20     char *writeThis = "Writing from Begining";
21     write(fd, writeThis, strlen(writeThis));
22
23     close(fd);
24
25     printf("*****Contents of random_text.txt*****\n");
26     system("cat random_text.txt");
27     printf("\n*****Contents of random_text.txt*****\n");
28
29     return 0;
30 }
31
32

```

random\_text.txt

The Department of Computer Science at GJU is a part of Electrical Engineering and Information Technology School (SEEIT).

```

> ./main
*****Contents of random_text.txt*****
The Department of Computer Science at GJU is a part of Electrical Engineerin
g and Information Technology School (SEEIT)
*****Contents of random_text.txt*****
*****Contents of random_text.txt*****
Writing from Beginingputer Science at GJU is a part of Electrical Engineerin
g and Information Technology School (SEEIT)
*****Contents of random_text.txt*****
>

```



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <string.h>
6
7 int main(void) {
8     printf("*****Contents of random_text.txt*****\n");
9     system("cat random_text.txt");
10    printf("\n*****Contents of random_text.txt*****\n");
11
12    int fd;
13    fd = open("random_text.txt", O_RDWR | O_CREAT, 0755);
14    if(fd < 0){
15        printf("Error when opening/creating file"); exit(1);
16    }
17    char *writeThis = "Writing from Middle";
18
19    lseek(fd, 5, SEEK_SET);
20    write(fd, writeThis, strlen(writeThis));
21
22    lseek(fd, 10, SEEK_CUR);
23    write(fd, writeThis, strlen(writeThis));
24
25    writeThis = "Writing from End";
26    lseek(fd, 0, SEEK_END);
27    write(fd, writeThis, strlen(writeThis));
28
29    close(fd);
30    printf("*****Contents of random_text.txt*****\n");
31    system("cat random_text.txt");
32    printf("\n*****Contents of random_text.txt*****\n");
33    return 0;
34 }

```

random\_text.txt

The Department of Computer Science at GJU is a part of the School of Electrical Engineering and Information Technology (SEEIT).

```

*****Contents of random_text.txt*****
The Department of Computer Science at GJU is a part of the School of Electrical
Engineering and Information Technology (SEEIT).
*****Contents of random_text.txt*****
*****Contents of random_text.txt*****
The DWriting from Middleer ScienceWriting from Middlef the School of Electrical
Engineering and Information Technogy (SEEIT).Writing from End
*****Contents of random_text.txt*****

```

# What does this program do?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <fcntl.h>
4  #include <errno.h>
5  #include <sys/types.h>
6  #include <unistd.h>
7  #define BUF_SIZE 8192
8
9  int main()
10 {
11     int input_fd, output_fd; /* Input and output file descriptors */
12     ssize_t ret_in, ret_out; /* Number of bytes returned by read() and write() */
13     char buffer[BUF_SIZE]; /* Character buffer */
14
15     /* Create input file descriptor */
16     input_fd = open ("IOtext.txt", O_RDONLY);
17     if (input_fd == -1) { perror ("open"); return 2; }
18     /* Create output file descriptor */
19     output_fd = open("Output.txt", O_WRONLY | O_CREAT, 0644);
20     if(output_fd == -1) { perror("open"); return 3; }
21     /* Copy process */
22     while((ret_in = read (input_fd, &buffer, BUF_SIZE)) > 0)
23     {
24         ret_out = write (output_fd, &buffer, (ssize_t) ret_in);
25         if(ret_out != ret_in) { perror("write"); return 4; }
26     }
27     /* Close file descriptors */
28     close (input_fd); close (output_fd);
29     system("ls -l .");
30     printf("\n*****The Content of Output.txt -- begining*****\n");
31     system("cat Output.txt");
32     printf("\n*****The Content of Output.txt -- end*****\n");
33     return (EXIT_SUCCESS);
34 }
```

← = 8\*1024 (i.e., 8k bytes)

IOtext.txt  
Hello CS 415 Students.

Output.txt  
Hello CS 415 Students.

```
> ./main
total 24
-rw-r--r-- 1 runner runner  22 Mar 22 21:46 IOtext.txt
-rwxr-xr-x 1 runner runner 8608 Mar 22 21:52 main
-rw-r--r-- 1 runner runner 1167 Mar 22 21:50 main.c
-rw-r--r-- 1 runner runner  22 Mar 22 21:52 Output.txt

*****The Content of Output.txt -- begining*****
Hello CS 415 Students.
*****The Content of Output.txt -- end*****
> 
```

# What does this program do?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <fcntl.h>
4  #include <errno.h>
5  #include <sys/types.h>
6  #include <unistd.h>
7  #define BUF_SIZE 8192
8  int main()
9  {
10     int input_fd, output_fd; /* Input and output file descriptors */
11     ssize_t ret_in, ret_out; /* Number of bytes returned by read() and write() */
12     char buffer[BUF_SIZE]; /* Character buffer */
13     /* Create input file descriptor */
14     input_fd = open ("Iotext.txt", O_RDONLY);
15     if (input_fd == -1) { perror ("open"); return 2; }
16     /* Create output file descriptor */
17     output_fd = open("Output.txt", O_WRONLY | O_CREAT, 0644);
18     if(output_fd == -1) { perror("open"); return 3; }
19     lseek (input_fd, 6, SEEK_SET);
20     /* Copy process */
21     while((ret_in = read (input_fd, &buffer, BUF_SIZE)) > 0)
22     {
23         ret_out = write (output_fd, &buffer, (ssize_t) ret_in);
24         if(ret_out != ret_in) { perror("write"); return 4; }
25     }
26     /* Close file descriptors */
27     close (input_fd); close (output_fd);
28     system("ls -l .");
29     printf("\n*****The Content of Output.txt -- begining*****\n");
30     system("cat Output.txt");
31     printf("\n*****The Content of Output.txt -- end*****\n");
32     return (EXIT_SUCCESS);
33 }
34
```

Iotext.txt  
Hello CS 415 Students.

Output.txt  
CS 415 Students.

```
➤ ./main
total 24
-rw-r--r-- 1 runner runner  22 Mar 22 21:46 Iotext.txt
-rwxr-xr-x 1 runner runner 8656 Mar 22 21:55 main
-rw-r--r-- 1 runner runner 1195 Mar 22 21:55 main.c
-rw-r--r-- 1 runner runner  16 Mar 22 21:55 Output.txt

*****The Content of Output.txt -- begining*****
CS 415 Students.
*****The Content of Output.txt -- end*****
➤
```