# Basic Linux File Handling

*Prepared By Eng. Manar AL-Jabr*

# Basic Linux File Handling

- Most Linux resources can be accessed as files

– ***File Descriptors***: *are small positive integers that act as indices into an array of open files that the kernel maintains for each process*

# Basic Linux File Handling

**Opening && Closing Files:**

- Two ways to open files:

  Open and create

–They are prototyped in:

<unistd.h>, You also must include <fcntl.h>

*– Both forms of open return file descriptor when they succeed!*

*–On failure returns –1 and set errno*

# Basic Linux File Handling

- **Reading && Writing Files:**
- **Example(1):**
  - Open text editor, write the following c code, and save file with .c extension. (example b.c)
  - Compile the code by opining terminal , and type the command:

    gcc b.c
  - to compile c programs you need gcc compiler,
  - if it is not installed on your machine, you can install it by command:

    sudo apt-get install gcc

# Basic Linux File Handling

## Example(1): b.c source code:

```c
// C program to illustrate
// open system call
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<stdlib.h>
#include<errno.h>


extern int errno;
int main()
{


        char  buf[2];
       // if file does not have in directory
        // then file foo.txt is created.
        int fd = open("foo.txt", O_RDONLY | O_CREAT);

        //printf("fd = %d/n", fd);

        if (fd ==-1)
        {
                // print which type of error have in a code
                printf("Error Number % d\n", errno);

                // print program detail "Success or failure"
                perror("Program");
        }


      int i =0;
          for (i=0; i < 3 ; i++)
         {
            printf("The read contents  of %s file are:   ", "foo.txt");

            read(fd,  buf, 2);
            //printf("%c", ch);

            // printf("%s",buf);
            printf(buf);

           printf("%s\n");

            }
      close(fd);
        return 0;
}
```

# Basic Linux File Handling

Example(1): **b.c run :**

- The contents of file foo.txt are:

```
abcdefghijklmnopqrstuvwxyz
```

- Compile b.c and execute with ./a.out

```
linux-bbme:/home/computer/fileSystem # gcc b.c
linux-bbme:/home/computer/fileSystem # ./a.out
The read contents  of foo.txt file are:   ab
The read contents  of foo.txt file are:   cd
The read contents  of foo.txt file are:   ef
linux-bbme:/home/computer/fileSystem #
```

# Basic Linux File Handling

## Positioning The file pointer:

1. Read and write from random location
2. **Lseek** call is the tool for this purpose

- SEEK_SET sets the pointer's offset bytes into the file
- SEEK_CUR sets the pointer's location to offset bytes into the file relative to the pointer's current location.

(offset can be negative)

# Basic Linux File Handling

## Positioning The file pointer:

- Example (b-2.c):

```
 char  buf[2];
// if file does not have in directory
 // then file foo.txt is created.

 int fd = open("foo.txt", O_RDONLY | O_CREAT);

 lseek(fd, 10, SEEK_SET);



  printf("The read contents  of %s file after 10 bytes

  read(fd, buf, 2);
     //printf("%c", ch);

  printf(buf);
  printf("\n");
  lseek(fd, 5, SEEK_CUR);

 printf("The read contents  of %s file after +5 bytes

 read(fd, buf, 2);
     //printf("%c", ch);

 printf(buf);
 printf("\n");

 return 0;
```

- يتم فتح الملف النصي للقراءة
- وضع المؤشر اي ازاحته عن موضع البداية بمقدار 10بايت (SEEK_SET)
- قراءة حرفين وتخزينهما في البفر ثم الطباعة.
- ازاحة المؤشر من موقعه الحالي (10) الى بعد 5 بايت وقراءة حرفين (SEEK_CUR)

# Basic Linux File Handling

## Positioning The file pointer:

- Example (b-2.c) << compile && run>>:

fileSystem : bash

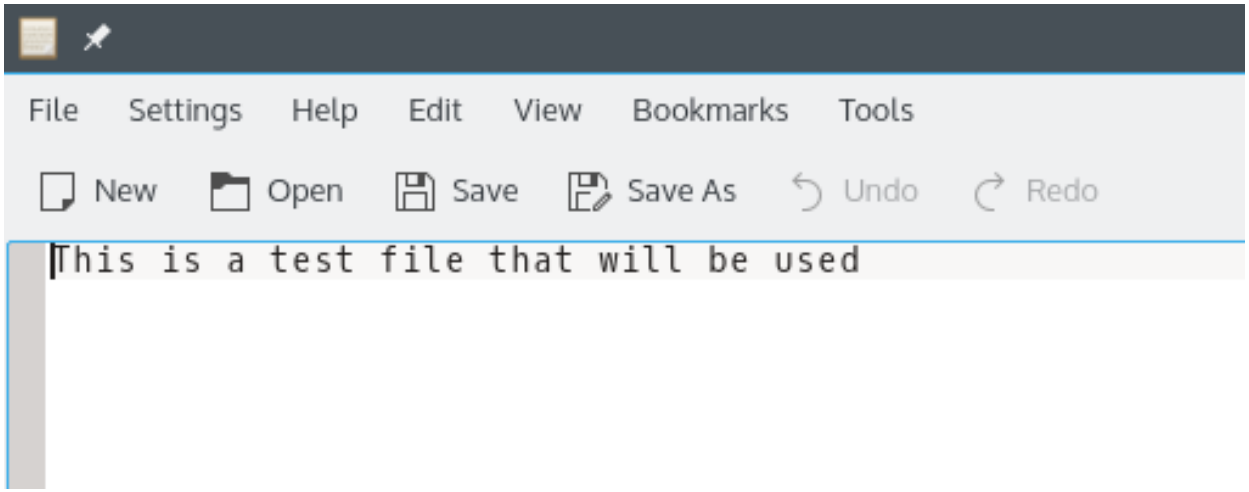
File Edit View Bookmarks Settings Help

```
linux-bbme:/home/computer/fileSystem # gcc b-2.c
linux-bbme:/home/computer/fileSystem # ./a.out
The read contents  of foo.txt file after 10 bytes are:   kl
??
The read contents  of foo.txt file after +5 bytes from current  are:   rs
??
linux-bbme:/home/computer/fileSystem #
```

# Basic Linux File Handling

## Positioning The file pointer:

- Example (mn.c):
- Create txt file and write the following statement:



# Basic Linux File Handling

## Positioning The file pointer:

- Example (mn.c):
- Write code:

```c
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
 // C program to illustrate
// open system call
#include<stdio.h>
#include<fcntl.h>
#include<errno.h>
#include <sys/types.h>
int main()
{
        int file=0;
        if((file=open("testfile.txt",O_RDONLY)) < -1)
                return 1;

        char buffer[19];
        if(read(file,buffer,19) != 19)  return 1;
        printf("%s\n",buffer);

        if(lseek(file,10,SEEK_SET) < 0) return 1;

        if(read(file,buffer,19) != 19)  return 1;
        printf("%s\n",buffer);

        return 0;
}
```

# Basic Linux File Handling

## Positioning The file pointer:

- Example (mn.c):
- Compile and Run Code:

```
fileSystem : bash
File   Edit   View   Bookmarks   Settings   Help
computer@linux-bbme:~/fileSystem> su
Password:
linux-bbme:/home/computer/fileSystem # gcc mn.c
linux-bbme:/home/computer/fileSystem # ./a.out
This is a test file[
test file that will[
linux-bbme:/home/computer/fileSystem #
```