5.**Introduction to pipes and related system calls for pipe management. Write a program to create a pipe and send “Hello” message.**

Introduction to Pipes: Pipes are one of the simplest forms of interprocess communication (IPC) in Unix-like systems. They allow data to flow from one process to another. Here are the key concepts:

1. Pipe Characteristics:
    - Unidirectional flow (one-way communication)
    - FIFO (First In, First Out) order
    - Works between related processes (parent-child)
2. Important System Calls:
    - pipe(): Creates a new pipe
    - read(): Reads data from the pipe
    - write(): Writes data to the pipe
    - close(): Closes pipe endpoints
3. File Descriptors:
    - pipe[0]: Read end of pipe
    - pipe[1]: Write end of pipe

<u>Let's break down how this program works:</u>

1. pipe(pipefd) creates a new pipe and stores file descriptors in the array:
    - pipefd[0] is for reading
    - pipefd[1] is for writing
2. write() sends data to the pipe:
    - First argument: write end of pipe (pipefd[1])
    - Second argument: data to write
    - Third argument: number of bytes to write
3. read() retrieves data from the pipe:
    - First argument: read end of pipe (pipefd[0])
    - Second argument: buffer to store data

    - Third argument: maximum number of bytes to read
4. close() is called on both ends when we're done

    To compile and run the program:

    I'll guide you through running this pipe program in Kali Linux step by step.

1. First, open Terminal in Kali Linux:
    - Click on the terminal icon, or
    - Press Ctrl + Alt + T
2. *Create the program file:*

    (editor) nano pipe_program.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
 #include <string.h>
int main()
{
 int pipefd[2];
 char buffer[20];
const char *message = "Hello";
if (pipe(pipefd) == -1)
{
perror("pipe");
exit(EXIT_FAILURE);
}
printf("Writing message to pipe...\n");
write(pipefd[1], message, strlen(message) + 1);
printf("Reading message from pipe...\n");
read(pipefd[0], buffer, sizeof(buffer));
printf("Received message: %s\n", buffer);
close(pipefd[0]);
close(pipefd[1]);
 return 0;
}
```

Save the file:

Quick save:ctrl+s

Compile the program:

```
gccpipe_program.c -o pipe_program
```

Run the program:

```
./pipe_program
```

Troubleshooting:

If gcc is not installed

sudo apt-get update sudo apt-get install gcc…

If you get "Permission denied"

sudochmod 755 pipe_program

The program should now run and show the output we discussed earlier:

```
Writing message to pipe...
 Reading message from pipe...
Received message: Hello
```

I'll show you how to save the program in nano editor in Kali Linux:

1. When you're in the nano editor, to save the file:
   - Press Ctrl + X (or ^X as shown at the bottom of nano)
   - You'll see: "Save modified buffer?"
   - Press Y for yes

  - Press Enter to confirm the filename

Alternative methods to save in nano:

- Ctrl + O then Enter (This saves without exiting)
- Ctrl + S (Quick save)

Quick nano editor commands reference:

Ctrl + X : Exit (will prompt to save)

Ctrl + O : Save without exiting

Ctrl + S : Quick save

Ctrl + C : Show current cursor position

Remember:

- The ^ symbol in nano's bottom menu means Ctrl
- After saving, you can verify the file exists by typing:

```
ls pipe_program.c
```

- To check the content of your saved file:

```
cat pipe_program.c
```