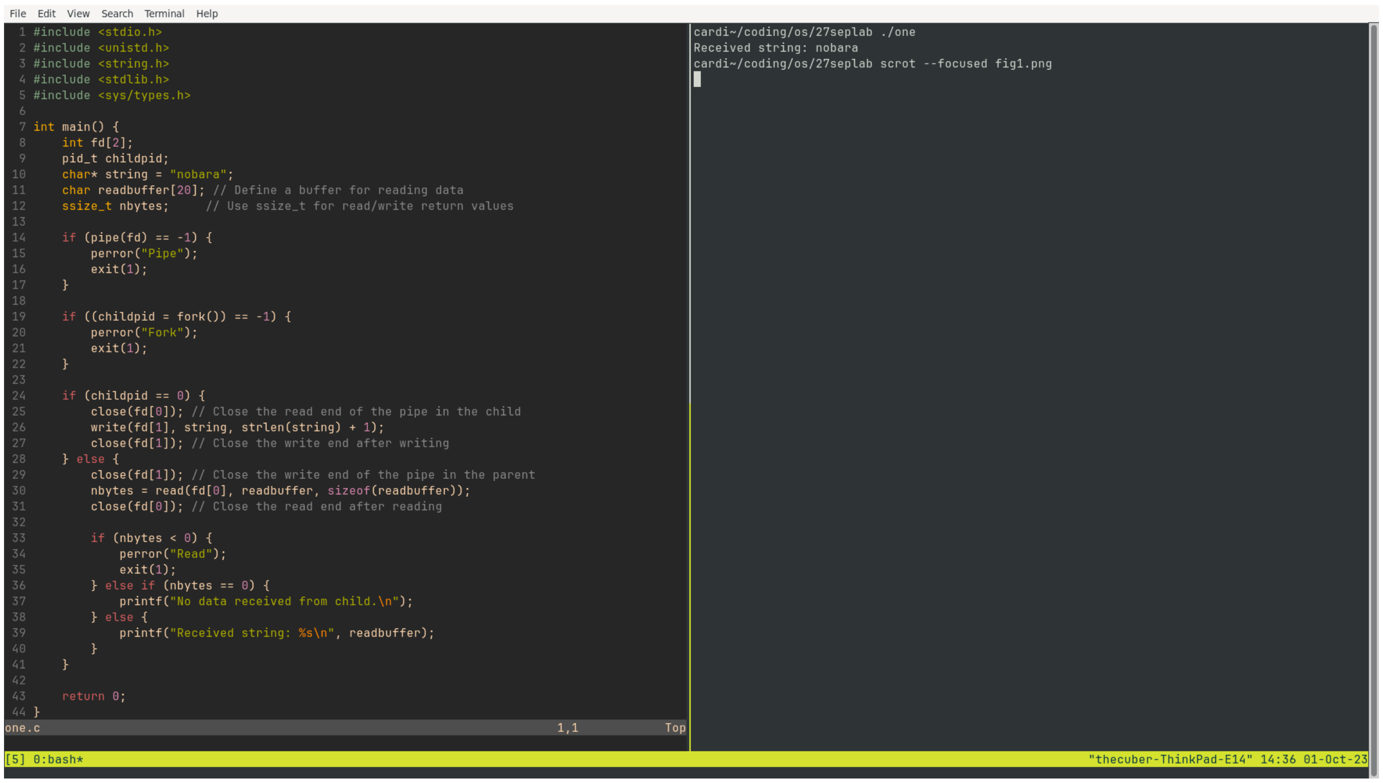
$22bps1059$

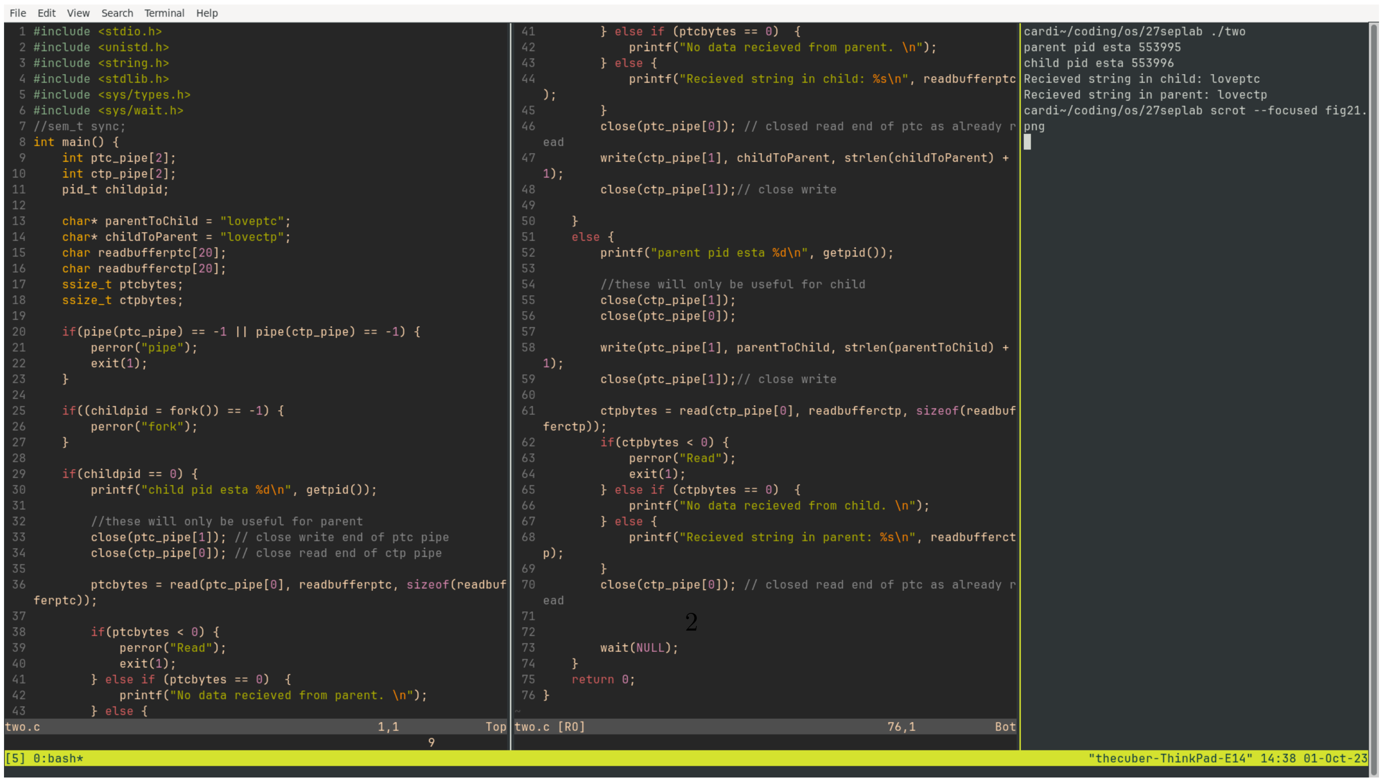
first



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

int main() {
    int fd[2];
    pid_t childpid;
    char* string = "nobara";
    char readbuffer[20]; // Define a buffer for reading data
    ssize_t nbytes;     // Use ssize_t for read/write return values

    if (pipe(fd) == -1) {
        perror("Pipe");
        exit(1);
    }

    if ((childpid = fork()) == -1) {
        perror("Fork");
        exit(1);
    }

    if (childpid == 0) {
        close(fd[0]); // Close the read end of the pipe in the child
        write(fd[1], string, strlen(string) + 1);
        close(fd[1]); // Close the write end after writing
    } else {
        close(fd[1]); // Close the write end of the pipe in the parent
        nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
        close(fd[0]); // Close the read end after reading

        if (nbytes < 0) {
            perror("Read");
            exit(1);
        } else if (nbytes == 0) {
            printf("No data received from child.\n");
        } else {
            printf("Received string: %s\n", readbuffer);
        }
    }

    return 0;
}
```

```
cardi~/coding/os/27seplab ./one
Received string: nobara
cardi~/coding/os/27seplab scrot --focused fig1.png
```

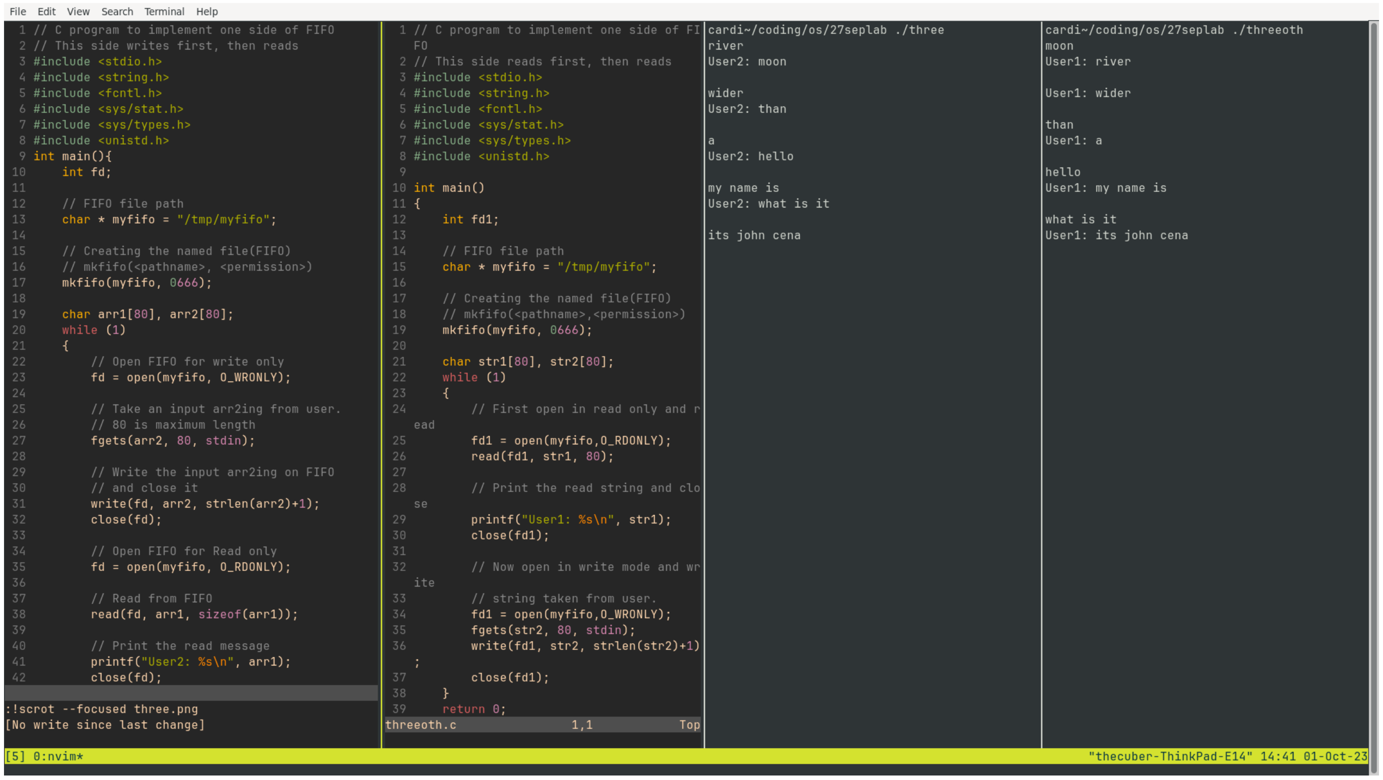
second



```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
//sem_t sync;
int main() {
    int ptc_pipe[2];
    int ctp_pipe[2];
    pid_t childpid;

    char* parentToChild = "loveptc";
    char* childToParent = "lovectp";
    char readbufferptc[20];
    char readbufferctp[20];
    ssize_t ptcbytes;
    ssize_t ctpbytes;

    if(pipe(ptc_pipe) == -1 || pipe(ctp_pipe) == -1) {
        perror("pipe");
        exit(1);
    }

    if((childpid = fork()) == -1) {
        perror("fork");
    }

    if(childpid == 0) {
        printf("child pid esta %d\n", getpid());

        //these will only be useful for parent
        close(ptc_pipe[1]); // close write end of ptc pipe
        close(ctp_pipe[0]); // close read end of ctp pipe

        ptcbytes = read(ptc_pipe[0], readbufferptc, sizeof(readbufferptc));

        if(ptcbytes < 0) {
            perror("Read");
            exit(1);
        } else if (ptcbytes == 0)  {
            printf("No data recieved from parent. \n");
        } else {
            printf("Recieved string in child: %s\n", readbufferptc);
        }
        close(ptc_pipe[0]); // closed read end of ptc as already read
        write(ctp_pipe[1], childToParent, strlen(childToParent) + 1);
        close(ctp_pipe[1]);// close write


    }
    else {
        printf("parent pid esta %d\n", getpid());

        //these will only be useful for child
        close(ctp_pipe[1]);
        close(ptc_pipe[0]);

        write(ptc_pipe[1], parentToChild, strlen(parentToChild) + 1);
        close(ptc_pipe[1]);// close write

        ctpbytes = read(ctp_pipe[0], readbufferctp, sizeof(readbufferctp));
        if(ctpbytes < 0) {
            perror("Read");
            exit(1);
        } else if (ctpbytes == 0)  {
            printf("No data recieved from child. \n");
        } else {
            printf("Recieved string in parent: %s\n", readbufferctp);
        }
        close(ctp_pipe[0]); // closed read end of ptc as already read


        wait(NULL);
    }
    return 0;
}
```

```
cardi~/coding/os/27seplab ./two
parent pid esta 553995
child pid esta 553996
Recieved string in child: loveptc
Recieved string in parent: lovectp
cardi~/coding/os/27seplab scrot --focused fig21.png
```

third



```c
// C program to implement one side of FIFO
// This side writes first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main(){
    int fd;

    // FIFO file path
    char * myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)
    // mkfifo(<pathname>, <permission>)
    mkfifo(myfifo, 0666);

    char arr1[80], arr2[80];
    while (1)
    {
        // Open FIFO for write only
        fd = open(myfifo, O_WRONLY);

        // Take an input arr2ing from user.
        // 80 is maximum length
        fgets(arr2, 80, stdin);

        // Write the input arr2ing on FIFO
        // and close it
        write(fd, arr2, strlen(arr2)+1);
        close(fd);

        // Open FIFO for Read only
        fd = open(myfifo, O_RDONLY);

        // Read from FIFO
        read(fd, arr1, sizeof(arr1));

        // Print the read message
        printf("User2: %s\n", arr1);
        close(fd);
```

```
:!scrot --focused three.png
[No write since last change]
```

```c
// C program to implement one side of FIFO
// This side reads first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int fd1;

    // FIFO file path
    char * myfifo = "/tmp/myfifo";

    // Creating the named file(FIFO)
    // mkfifo(<pathname>,<permission>)
    mkfifo(myfifo, 0666);

    char str1[80], str2[80];
    while (1)
    {
        // First open in read only and read
        fd1 = open(myfifo,O_RDONLY);
        read(fd1, str1, 80);

        // Print the read string and close
        printf("User1: %s\n", str1);
        close(fd1);

        // Now open in write mode and write
        // string taken from user.
        fd1 = open(myfifo,O_WRONLY);
        fgets(str2, 80, stdin);
        write(fd1, str2, strlen(str2)+1);
        close(fd1);
    }
    return 0;
```

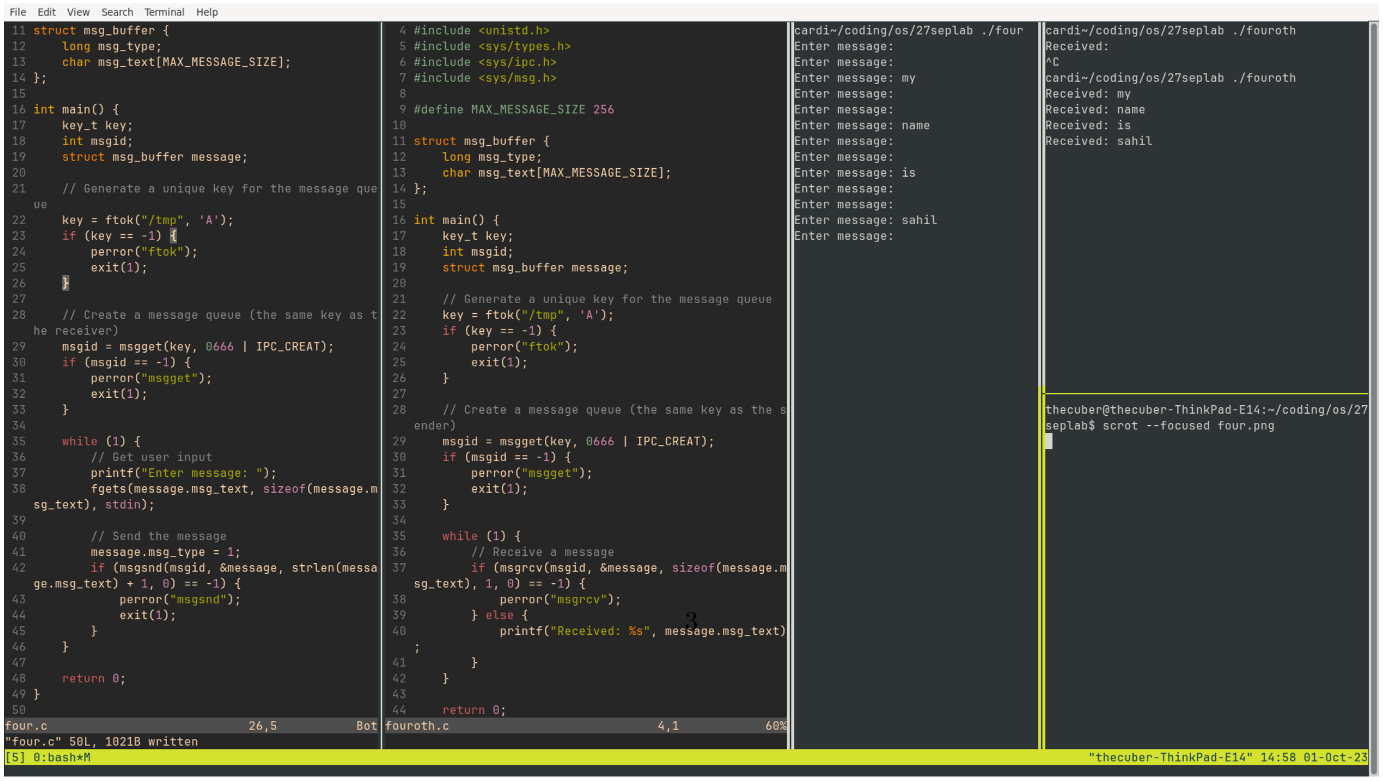
```
threeoth.c                    1,1            Top
```

```
cardi~/coding/os/27seplab ./three
river
User2: moon

wider
User2: than

a
User2: hello

my name is
User2: what is it

its john cena
```

```
cardi~/coding/os/27seplab ./threeoth
moon
User1: river

User1: wider

than
User1: a

hello
User1: my name is

what is it
User1: its john cena
```

```
[5] 0:nvim*                                   "thecuber-ThinkPad-E14" 14:41 01-Oct-23
```

fourth



```c
struct msg_buffer {
    long msg_type;
    char msg_text[MAX_MESSAGE_SIZE];
};

int main() {
    key_t key;
    int msgid;
    struct msg_buffer message;

    // Generate a unique key for the message queue
    key = ftok("/tmp", 'A');
    if (key == -1) {
        perror("ftok");
        exit(1);
    }

    // Create a message queue (the same key as the receiver)
    msgid = msgget(key, 0666 | IPC_CREAT);
    if (msgid == -1) {
        perror("msgget");
        exit(1);
    }

    while (1) {
        // Get user input
        printf("Enter message: ");
        fgets(message.msg_text, sizeof(message.msg_text), stdin);

        // Send the message
        message.msg_type = 1;
        if (msgsnd(msgid, &message, strlen(message.msg_text) + 1, 0) == -1) {
            perror("msgsnd");
            exit(1);
        }
    }

    return 0;
}
```

```
four.c                        26,5           Bot
"four.c" 50L, 1021B written
```

```c
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#define MAX_MESSAGE_SIZE 256

struct msg_buffer {
    long msg_type;
    char msg_text[MAX_MESSAGE_SIZE];
};

int main() {
    key_t key;
    int msgid;
    struct msg_buffer message;

    // Generate a unique key for the message queue
    key = ftok("/tmp", 'A');
    if (key == -1) {
        perror("ftok");
        exit(1);
    }

    // Create a message queue (the same key as the sender)
    msgid = msgget(key, 0666 | IPC_CREAT);
    if (msgid == -1) {
        perror("msgget");
        exit(1);
    }

    while (1) {
        // Receive a message
        if (msgrcv(msgid, &message, sizeof(message.msg_text), 1, 0) == -1) {
            perror("msgrcv");
        } else {
            printf("Received: %s", message.msg_text);
        }
    }

    return 0;
```

```
fouroth.c                          4,1         60%
```

```
cardi~/coding/os/27seplab ./four
Enter message:
Enter message:
Enter message: my
Enter message:
Enter message:
Enter message: name
Enter message:
Enter message:
Enter message: is
Enter message:
Enter message:
Enter message: sahil
Enter message:
```

```
cardi~/coding/os/27seplab ./fouroth
Received:
^C
cardi~/coding/os/27seplab ./fouroth
Received: my
Received: name
Received: is
Received: sahil
```

```
thecuber@thecuber-ThinkPad-E14:~/coding/os/27seplab$ scrot --focused four.png
```

```
[5] 0:bash*M                                  "thecuber-ThinkPad-E14" 14:58 01-Oct-23
```