

22bps1059

first

```
File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <sys/types.h>
6
7 int main() {
8     int fd[2];
9     pid_t childpid;
10    char* string = "nobara";
11    char readbuffer[20]; // Define a buffer for reading data
12    ssize_t nbytes;      // Use ssize_t for read/write return values
13
14    if (pipe(fd) == -1) {
15        perror("Pipe");
16        exit(1);
17    }
18
19    if ((childpid = fork()) == -1) {
20        perror("Fork");
21        exit(1);
22    }
23
24    if (childpid == 0) {
25        close(fd[0]); // Close the read end of the pipe in the child
26        write(fd[1], string, strlen(string) + 1);
27        close(fd[1]); // Close the write end after writing
28    } else {
29        close(fd[1]); // Close the write end of the pipe in the parent
30        nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
31        close(fd[0]); // Close the read end after reading
32
33        if (nbytes < 0) {
34            perror("Read");
35            exit(1);
36        } else if (nbytes == 0) {
37            printf("No data received from child.\n");
38        } else {
39            printf("Received string: %s\n", readbuffer);
40        }
41    }
42
43    return 0;
44 }
one.c 1,1 Top
cardi~/coding/os/27seplab ./one
Received string: nobara
cardi~/coding/os/27seplab scrot --focused fig1.png
[5] 0:~bash* "thecuber-ThinkPad-E14" 14:36 01-Oct-23
```

second

```
File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 //sem_t sync;
8 int main() {
9     int ptc_pipe[2];
10    int ctp_pipe[2];
11    pid_t childpid;
12
13    char* parentToChild = "loveptc";
14    char* childToParent = "lovecctp";
15    char readbufferptc[20];
16    char readbufferctp[20];
17    ssize_t ptcbytes;
18    ssize_t ctpbytes;
19
20    if (pipe(ptc_pipe) == -1 || pipe(ctp_pipe) == -1) {
21        perror("pipe");
22        exit(1);
23    }
24
25    if ((childpid = fork()) == -1) {
26        perror("fork");
27    }
28
29    if (childpid == 0) {
30        printf("child pid esta %d\n", getpid());
31
32        //these will only be useful for parent
33        close(ptc_pipe[1]); // close write end of ptc pipe
34        close(ctp_pipe[0]); // close read end of ctp pipe
35
36        ptcbytes = read(ptc_pipe[0], readbufferptc, sizeof(readbufferptc));
37
38        if (ptcbytes < 0) {
39            perror("Read");
40            exit(1);
41        } else if (ptcbytes == 0) {
42            printf("No data recieved from parent. \n");
43        } else {
44            printf("Recieved string in parent: %s\n", readbufferptc);
45        }
46
47        //these will only be useful for child
48        close(ctp_pipe[1]); // close write end of ctp pipe
49        close(ptc_pipe[0]); // close read end of ptc pipe
50
51        ctpbytes = read(ctp_pipe[0], readbufferctp, sizeof(readbufferctp));
52
53        if (ctpbytes < 0) {
54            perror("Read");
55            exit(1);
56        } else if (ctpbytes == 0) {
57            printf("No data recieved from child. \n");
58        } else {
59            printf("Recieved string in child: %s\n", readbufferctp);
60        }
61
62        //these will only be useful for parent
63        close(ptc_pipe[1]); // close write end of ptc pipe
64        close(ctp_pipe[0]); // close read end of ctp pipe
65
66        ptcbytes = read(ptc_pipe[0], readbufferptc, sizeof(readbufferptc));
67
68        if (ptcbytes < 0) {
69            perror("Read");
70            exit(1);
71        } else if (ptcbytes == 0) {
72            printf("No data recieved from parent. \n");
73        } else {
74            printf("Recieved string in parent: %s\n", readbufferptc);
75        }
76    }
77
78    if (childpid > 0) {
79        printf("parent pid esta %d\n", getpid());
80
81        //these will only be useful for child
82        close(ctp_pipe[1]); // close write end of ctp pipe
83        close(ptc_pipe[0]); // close read end of ptc pipe
84
85        ctpbytes = read(ctp_pipe[0], readbufferctp, sizeof(readbufferctp));
86
87        if (ctpbytes < 0) {
88            perror("Read");
89            exit(1);
90        } else if (ctpbytes == 0) {
91            printf("No data recieved from child. \n");
92        } else {
93            printf("Recieved string in child: %s\n", readbufferctp);
94        }
95
96        //these will only be useful for parent
97        close(ptc_pipe[1]); // close write end of ptc pipe
98        close(ctp_pipe[0]); // close read end of ctp pipe
99
100       ptcbytes = read(ptc_pipe[0], readbufferptc, sizeof(readbufferptc));
101
102       if (ptcbytes < 0) {
103           perror("Read");
104           exit(1);
105       } else if (ptcbytes == 0) {
106           printf("No data recieved from parent. \n");
107       } else {
108           printf("Recieved string in parent: %s\n", readbufferptc);
109       }
110   }
111
112   wait(NULL);
113   return 0;
114 }
two.c 1,1 Top
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
[5] 0:~bash* "thecuber-ThinkPad-E14" 14:38 01-Oct-23
```

third

<pre> 1 // C program to implement one side of FIFO 2 // This side writes first, then reads 3 #include <stdio.h> 4 #include <string.h> 5 #include <fcntl.h> 6 #include <sys/stat.h> 7 #include <sys/types.h> 8 #include <unistd.h> 9 int main(){ 10 int fd; 11 12 // FIFO file path 13 char * myfifo = "/tmp/myfifo"; 14 15 // Creating the named file(FIFO) 16 // mkfifo(<pathname>, <permission>) 17 mkfifo(myfifo, 0666); 18 19 char arr1[80], arr2[80]; 20 while (1) 21 { 22 // Open FIFO for write only 23 fd = open(myfifo, O_WRONLY); 24 25 // Take an input arr2ing from user. 26 // 80 is maximum length 27 fgets(arr2, 80, stdin); 28 29 // Write the input arr2ing on FIFO 30 // and close it 31 write(fd, arr2, strlen(arr2)+1); 32 close(fd); 33 34 // Open FIFO for Read only 35 fd = open(myfifo, O_RDONLY); 36 37 // Read from FIFO 38 read(fd, arr1, sizeof(arr1)); 39 40 // Print the read message 41 printf("User2: %s\n", arr1); 42 close(fd); </pre>	<pre> 1 // C program to implement one side of FI 2 // This side reads first, then reads 3 #include <stdio.h> 4 #include <string.h> 5 #include <fcntl.h> 6 #include <sys/stat.h> 7 #include <sys/types.h> 8 #include <unistd.h> 9 10 int main() 11 { 12 int fd1; 13 14 // FIFO file path 15 char * myfifo = "/tmp/myfifo"; 16 17 // Creating the named file(FIFO) 18 // mkfifo(<pathname>, <permission>) 19 mkfifo(myfifo, 0666); 20 21 char str1[80], str2[80]; 22 while (1) 23 { 24 // First open in read only and r 25 fd1 = open(myfifo, O_RDONLY); 26 read(fd1, str1, 80); 27 28 // Print the read string and clo 29 printf("User1: %s\n", str1); 30 close(fd1); 31 32 // Now open in write mode and wr 33 fd1 = open(myfifo, O_WRONLY); 34 fgetc(str2, 80, stdin); 35 write(fd1, str2, strlen(str2)+1) 36 37 close(fd1); 38 } 39 return 0; </pre>	<pre> cardi~/coding/os/27seplab ./three river User2: moon wider User2: than a User2: hello my name is User2: what is it its john cena </pre>	<pre> cardi~/coding/os/27seplab ./threeoeth moon User1: river wider User1: wider than User1: a hello User1: my name is what is it User1: its john cena </pre>
<pre> !lsctrl --focused three.png [No write since last change] </pre>	<pre> threeoeth.c 1,1 Top </pre>	<pre> "thecuber-ThinkPad-E14" 14:41 01-Oct-23 </pre>	

fourth

<pre> 11 struct msg_buffer { 12 long msg_type; 13 char msg_text[MAX_MESSAGE_SIZE]; 14 }; 15 16 int main() { 17 key_t key; 18 int msgid; 19 struct msg_buffer message; 20 21 // Generate a unique key for the message que 22 ue 23 key = ftok("/tmp", 'A'); 24 if (key == -1) { 25 perror("ftok"); 26 exit(1); 27 } 28 29 // Create a message queue (the same key as t 30 he receiver) 31 msgid = msgget(key, 0666 IPC_CREAT); 32 if (msgid == -1) { 33 perror("msgget"); 34 exit(1); 35 } 36 37 while (1) { 38 // Get user input 39 printf("Enter message: "); 40 fgets(message.msg_text, sizeof(message.m 41 sg_text), stdin); 42 43 // Send the message 44 message.msg_type = 1; 45 if (msgsnd(msgid, &message, strlen(messa 46 ge.msg_text) + 1, 0) == -1) { 47 perror("msgsnd"); 48 exit(1); 49 } 50 } 51 return 0; </pre>	<pre> 4 #include <unistd.h> 5 #include <sys/types.h> 6 #include <sys/ipc.h> 7 #include <sys/msg.h> 8 9 #define MAX_MESSAGE_SIZE 256 10 11 struct msg_buffer { 12 long msg_type; 13 char msg_text[MAX_MESSAGE_SIZE]; 14 }; 15 16 int main() { 17 key_t key; 18 int msgid; 19 struct msg_buffer message; 20 21 // Generate a unique key for the message queue 22 key = ftok("/tmp", 'A'); 23 if (key == -1) { 24 perror("ftok"); 25 exit(1); 26 } 27 28 // Create a message queue (the same key as the s 29 ender) 30 msgid = msgget(key, 0666 IPC_CREAT); 31 if (msgid == -1) { 32 perror("msgget"); 33 exit(1); 34 } 35 36 while (1) { 37 // Receive a message 38 if (msgrcv(msgid, &message, sizeof(message.m 39 sg_text), 1, 0) == -1) { 40 perror("msgrcv"); 41 exit(1); 42 } else { 43 printf("Received: %s", message.msg_text) 44 } 45 } 46 return 0; </pre>	<pre> cardi~/coding/os/27seplab ./four Enter message: Received: ^C cardi~/coding/os/27seplab ./fouroeth Received: my Received: name Received: is Received: sahil </pre>	<pre> cardi~/coding/os/27seplab ./fouroeth Received: my Received: name Received: is Received: sahil </pre>
<pre> four.c 50L, 1821B written </pre>	<pre> fouroeth.c 4,1 68% </pre>	<pre> thecuber@thecuber-ThinkPad-E14:~/coding/os/27 seplab\$ scrot --focused four.png </pre>	
<pre> "thecuber-ThinkPad-E14" 14:58 01-Oct-23 </pre>			