

操作系统实验

实验 4 Linux进程间的通信

一、实验目的及要求

学习如何利用管道机制、消息缓冲队列、共享存储区机制进行进程间的通信，并加深对上述通信机制的理解。

二、实验内容

1. 了解系统调用pipe()的功能和实现过程。
2. 编写一C语言程序，使其用管道来实现父子进程间通信。子进程向父进程发送字符串“is sending a message to parent!”；父进程则从管道中读出子进程发来的消息，并将其显示到屏幕上，然后终止。
3. 运行该程序，观察、记录并简单分析其运行结果。

三、实验源码

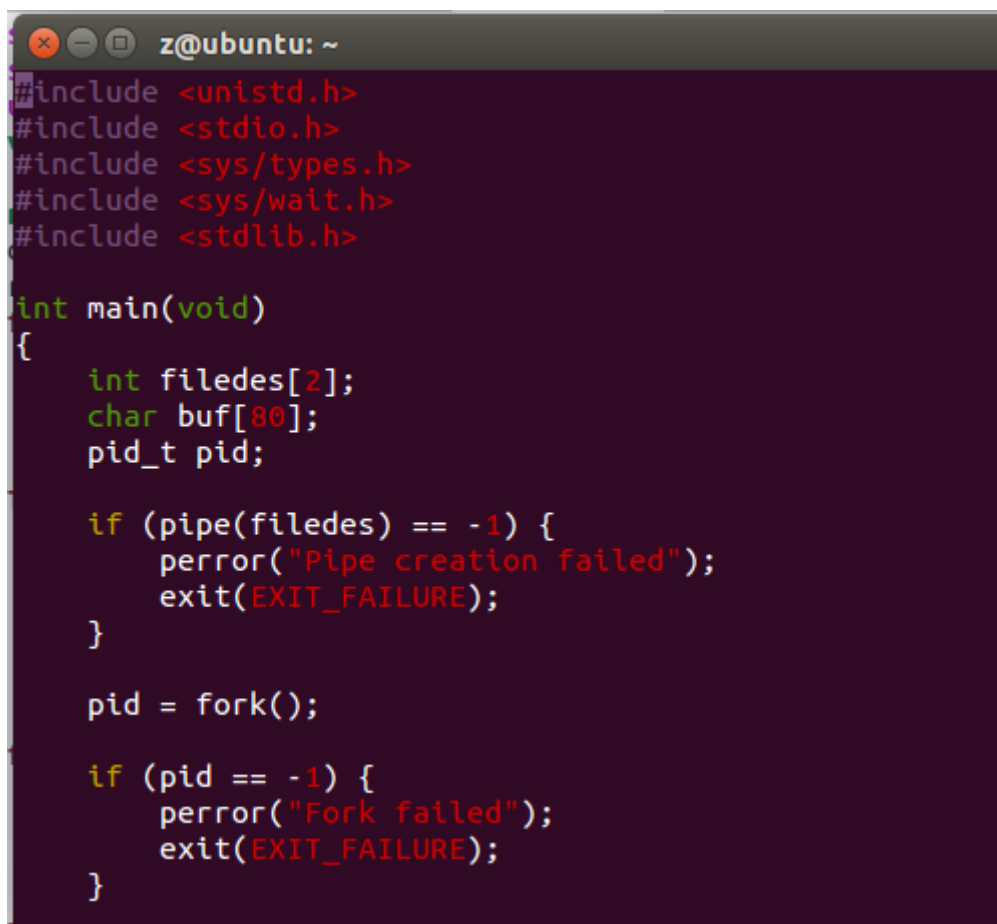
```
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <sys/types.h>
4  #include <sys/wait.h>
5  #include <stdlib.h>
6
7  int main(void)
8  {
9      int filed[2];
10     char buf[80];
11     pid_t pid;
12
13     if (pipe(filed) == -1) {
14         perror("Pipe creation failed");
15         exit(EXIT_FAILURE);
16     }
17
18     pid = fork();
19
20     if (pid == -1) {
21         perror("Fork failed");
22         exit(EXIT_FAILURE);
23     }
24
25     if (pid == 0)
26     {
27         close(filed[0]); //关闭管道的读端口
```

```

28     printf("This is in the child process, here write a string to the
pipe.\n");
29     char s[] = "Is sending a message to parent!\n";
30     write(filedes[1], s, sizeof(s));
31     close(filedes[1]); // 关闭管道的写端口
32 }
33 else
34 {
35     close(filedes[1]); // 关闭管道的写端口
36     printf("This is in the parent process, here read a string from the
pipe.\n");
37     read(filedes[0], buf, sizeof(buf));
38     printf("%s\n", buf);
39     close(filedes[0]); // 关闭管道的读端口
40 }
41 return 0;
42 }

```

四、实验结果



```

z@ubuntu: ~
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main(void)
{
    int filedes[2];
    char buf[80];
    pid_t pid;

    if (pipe(filedes) == -1) {
        perror("Pipe creation failed");
        exit(EXIT_FAILURE);
    }

    pid = fork();

    if (pid == -1) {
        perror("Fork failed");
        exit(EXIT_FAILURE);
    }
}

```

```

z@ubuntu:~$ vi pipe.c
z@ubuntu:~$ gcc -o pipe pipe.c
z@ubuntu:~$ ./pipe
This is in the parent process, here read a string from the pipe.
This is in the child process, here write a string to the pipe.
Is sending a message to parent!

```

五、结果分析

程序使用了管道（pipe）来实现父子进程之间的进程间通信。程序创建了一个管道，然后通过fork函数创建了子进程。

在子进程中（pid == 0），首先关闭了管道的读端口（filedes[0]），然后使用write函数向管道的写端口（filedes[1]）写入了一个字符串"Is sending a message to parent!\n"。

在父进程中（pid != 0），首先关闭了管道的写端口（filedes[1]），然后使用read函数从管道的读端口（filedes[0]）读取数据，并将其存储在缓冲区buf中。接着，父进程打印了读取到的字符串。

管道是一种半双工的通信方式，子进程可以向管道写入数据，父进程可以从管道读取数据。父子进程之间使用管道进行通信的目的是实现数据的传递。子进程将数据写入管道，父进程从管道读取数据并进行处理。