OPERATING SYSTEMS

# Interprocess Communication

## Operating Systems

郑贵锋 博士
中山大学计算机学院
zhenggf@mail.sysu.edu.cn
https://gitee.com/code_sysu

中山大學 SUN YAT-SEN UNIVERSITY



## 目录


- IPC概述
- 共享内存系统
- 消息传递系统
- 管道Pipes
- 客户机-服务器系统中的通信
  - 套接字Sockets
  - 远程过程调用RPC

## Pipes



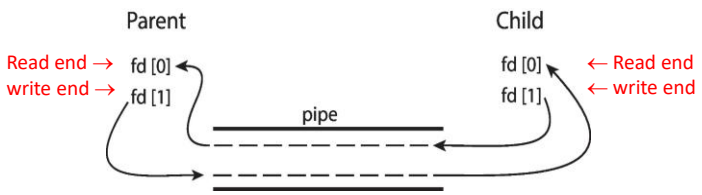
- **管道**
  - 管道充当导管，允许两个进程通信
    - 没有任何结构的字节流
  - 实现通信过程中的一些问题：
    - 通信是单向的还是双向的？
    - 在双向通信的情况下，是半双工还是全双工？
    - 通信进程之间是否必须存在关系（如父子关系）？
    - 管道是否可以通过网络使用，或者进程是否必须驻留在同一台主机上？

## Pipes



- **管道**
  - 无名管道
    - 无名管道(普通管道)允许以标准的生产者-消费者风格进行通信
      - 生产者写入一端（wirte-end）。
      - 消费者从另一端（read-end）读取数据。
    - 因此，普通管道是单向的(unidirectional). 如果需要双向通信，则必须使用两根管道。
    - 在UNIX系统上，普通管道是使用函数构造的

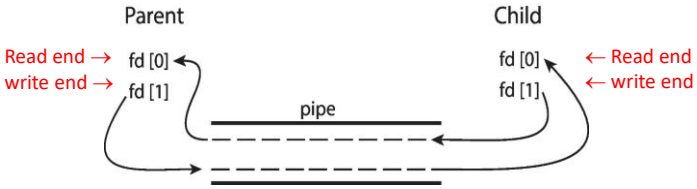
```
pipe(int fd[])
```



## Pipes



- **管道**
  - 无名管道
    - 无名管道被视为一种特殊类型的文件。可通过int fd[]文件描述符访问创建的管道。
      - fd[0]管道的读取端
      - fd[1]管道的写入端
    - 因此，可以使用普通的 read() 和 write() 系统调用访问管道。
    - 普通管道无法从创建它的进程外部访问。通常，父进程创建一个管道，并使用它与通过fork()创建的子进程通信。子进程继承管道就像从其父进程继承打开的文件一样。




## Pipes



- **管道**
  - 命名管道
    - 命名管道(Named Pipes) 比普通管道更强大。
      - 通信是双向的。
      - 通信进程之间不需要父子关系。
    - 几个进程可以使用已建立的命名管道进行通信。
    - 类UNIX和Windows系统上都提供了命名管道。

## Pipes

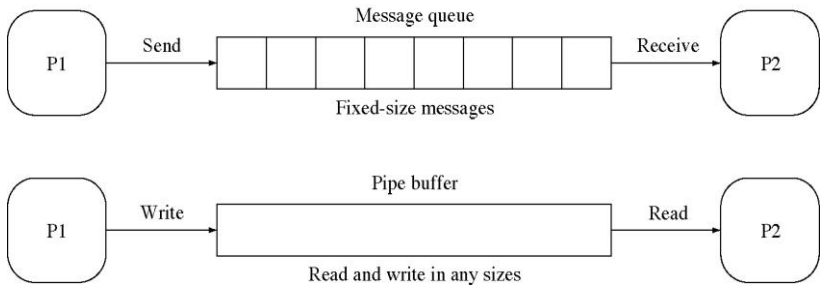


- **管道**
  - UNIX/Linux中的命名管道
    - 命名管道称为FIFO，是通过 mkfifo() 系统调用创建的。
      - 创建后，它们在文件系统中显示为典型文件，并通过普通的 open(), read(), write(), 和close() 系统调用进行操作。
    - FIFO将继续存在，直到从文件系统中明确删除。
    - FIFO允许双向通信和半双工传输。
      - 如果数据必须双工传输，通常使用两个FIFO。
    - 通信进程必须位于同一台机器/主机上。
      - 如果需要机器间通信，请使用*套接字socket*

## Pipes



- **管道**
  - 消息传递 vs. 管道



## Examples



- **Linux：管道**
  - Linux用户级别限制
    - 可以在/etc/security/limits.conf中重新定义

```
isscgy@ubuntu:/mnt/os-2020$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 7645
max locked memory       (kbytes, -l) 65536
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 7645
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
isscgy@ubuntu:/mnt/os-2020$
```

    - 管道容量为512（字节）*8=4096（字节）=4KB=1页

## Examples



- **Linux：管道**
  - 函数
    - 无名管道

```
int pipefd[2];
int pipe(int pipefd);

int fcntl(int pipefd[0|1], int cmd);
int fcntl(int pipefd[0|1], int cmd, long arg);

ssize_t write(int pipefd[1], void* buf, size_t count);
ssize_t read(int pipefd[0], void* buf, size_t count);

close(pipefd[0]);
close(pipefd[1]);
```

## Examples



- **Linux：管道**
  - **函数**
    - **命名管道**

```
#define FIFO pathname /* pathname: "/tmp/my_fifo" */

unlink(FIFO); /* 删除命名管道，即删除文件 */

mkfifo(FIFO, 0666);
int fdw = open(FIFO, O_RDWR);

mkfifo(FIFO, 0444);
int fdr = open(FIFO, O_RDONLY);

ssize_t write(int fdw, void* buf, size_t count);
/* ssizt_t = signed int, sizt_t = unsigned int */

ssize_t read(int fdr, void* buf, size_t count);

close(fdw);
close(fdr);
```

## Examples



- **Linux：管道**
  - **管道缓冲区大小和管道容量**
    - **算法** 10-1: single pipe buffer (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#define ERR_EXIT(m) \
    do { \
        perror(m); \
        exit(EXIT_FAILURE); \
    } while (0)

int main(int argc, char *argv[])
{
    int pipefd[2];
    int bufsize;
    char *buffer;
    int flags, ret, lastwritten, count, totalwritten;

    if(pipe(pipefd) == -1) /* 创建无名管道 */
        ERR_EXIT("pipe()");
    flags = fcntl(pipefd[1], F_GETFL);
    fcntl(pipefd[1], F_SETFL, flags | O_NONBLOCK); /* 设置write_end非阻塞 */
    bufsize = atoi(argv[1]);
    printf("testing buffer size = %d\n", bufsize);
    buffer = (char *)malloc(bufsize*sizeof(char));
    if(buffer == NULL || bufsize == 0)
        ERR_EXIT("malloc()");
```

## Examples



- **Linux：管道**
  - 管道缓冲区大小和管道容量
    - 算法 10-1: single pipe buffer(2)

```
    count = 0;
    while (1) {
        ret = write(pipefd[1], buffer, bufsize);
            /* bufsize is better to be 2^k */
        if(ret == -1) {
            perror("write()");
            break;
        }
        lastwritten = ret;
        count++;
    }
    totalwritten = (count-1)*bufsize + lastwritten;
    printf("single pipe buffer count =  %d, last written = %d bytes\n", count,
lastwritten);
    printf("total written = %d bytes = %d KiB\n", totalwritten,
totalwritten/1024); /* pipe buffer */

    return 0;
}
```

## Examples



- **Linux：管道**
  - 管道缓冲区大小和管道容量
    - 算法 10-1: single pipe buffer(2)

```
isscgy@ubuntu:/mnt/os-2020$ ./a.out 1024
testing buffer size = 1024
write(): Resource temporarily unavailable
single pipe buffer count =  64, last written = 1024 bytes
total written = 65536 bytes = 64 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 4096
testing buffer size = 4096
write(): Resource temporarily unavailable
single pipe buffer count =  16, last written = 4096 bytes
total written = 65536 bytes = 64 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 4097
testing buffer size = 4097
write(): Resource temporarily unavailable
single pipe buffer count =  11, last written = 4096 bytes
total written = 45066 bytes = 44 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 32768
testing buffer size = 32768
write(): Resource temporarily unavailable
single pipe buffer count =  2, last written = 32768 bytes
total written = 65536 bytes = 64 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 32769
testing buffer size = 32769
write(): Resource temporarily unavailable
single pipe buffer count =  2, last written = 28673 bytes
total written = 61442 bytes = 60 KiB
isscgy@ubuntu:/mnt/os-2020$
```

, count,

## Examples



- **Linux：管道**
  - 管道缓冲区大小和管道容量

```
isscgy@ubuntu:/mnt/os-2020$ ./a.out 1024
testing buffer size = 1024
write(): Resource temporarily unavailable
single pipe buffer count =  64, last written = 1024 bytes
total written = 65536 bytes = 64 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 4096
testing buffer size = 4096
write(): Resource temporarily unavailable
single pipe buffer count =  16, last written = 4096 bytes
total written = 65536 bytes = 64 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 4097
testing buffer size = 4097
write(): Resource temporarily unavailable
single pipe buffer count =  11, last written = 4096 bytes
total written = 45066 bytes = 44 KiB
isscgy@ubuntu:/mnt/os-2020$ ./a.out 32768
testing buffer size = 32768
write(): Resource temporar
single pipe buffer count =
total written = 65536 byte
isscgy@ubuntu:/mnt/os-2020
testing buffer size = 3276
write(): Resource temporar
single pipe buffer count =
total written = 61442 byte
isscgy@ubuntu:/mnt/os-2020$
```

, count,

与PIPE_BUF的不同之处在于PIPE_SIZE是实际内存分配的长度，而PIPE_BUF保证原子性。
检查头文件include/linux/pipi_fs_i.h
#define PIPE_SIZE PAGE_SIZE /* 4 KiB */
#define PIPE_DEF_BUFFERS 16 /* 4*16 = 64 KiB */

## Examples



- **Linux：管道**
  - 管道缓冲区大小和管道容量
    - 算法 10-2: Total pipes capacity.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#define PIPE_SIZE 64*1024 /* 64 KiB */
```

```
int main(void)
{
    char buf[PIPE_SIZE];
    int testfd[600][2];
    int i;
    long int ret;

    for (i = 0; i < 600; i++) {
        if(pipe(testfd[i]) == -1) {
            perror("pipe()");
            break;
        }
        fcntl(testfd[i][1], F_SETFL, O_NONBLOCK);
        ret = write(testfd[i][1], buf, PIPE_SIZE);
        if(ret == -1 || ret != PIPE_SIZE) {
            perror("write()");
            break;
        }
    }
    printf("\nsingle pipe buffer = 64 KiB, pipes created: %d\n", i);
    printf("total used size: %ld bytes = %ld KiB, or %.0f MiB\n", (long
int)i*PIPE_SIZE, (long int)i*PIPE_SIZE/1024, (double)i*PIPE_SIZE/1024/1024);
    return 0;
}
```

## Examples



- **Linux：管道**
  - 管道缓冲区大小和管道容量
    - 算法 10-2: Total pipes capacity.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#define PIPE_SIZE 64*1024 /* 64 KiB */
```

```
int main(void)
{
    char buf[PIPE_SIZE];
    int testfd[600][2];
```

```
isscgy@ubuntu:/mnt/os-2020$ gcc alg.10-2-pipecapacity.c
isscgy@ubuntu:/mnt/os-2020$ ./a.out
pipe(): Too many open files

single pipe buffer = 64 KiB, pipes created: 510
total used size: 33423360 bytes = 32640 KiB, or 32 MiB
isscgy@ubuntu:/mnt/os-2020$
```

```
            if (ret == -1 || ret != PIPE_SIZE) {
                perror("write()");
                break;
            }
        }
        printf("\nsingle pipe buffer = 64 KiB, pipes created: %d\n", i);
        printf("total used size: %ld bytes = %ld KiB, or %.0f MiB\n", (long
    int)i*PIPE_SIZE, (long int)i*PIPE_SIZE/1024, (double)i*PIPE_SIZE/1024/1024);
        return 0;
    }
```

## Examples



- **Linux：管道**
  - UNIX/Linux中的无名管道
    - 算法 10-3: pipe-ord-1.c(1)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#define READ_END 0
#define WRITE_END 1
```

```
/* 阻塞读取版本 */
int main(void)
{
    char write_msg[BUFSIZ]; /* BUFSIZ = 8192bytes, saved from stdin */
    char read_msg[BUFSIZ];
    int pipefd[2]; /* pipefd[0] for READ_END, pipefd[1] for WRITE_END */
    int flags;
    pid_t pid;

    if(pipe(pipefd) == -1) { /* create a pipe */
        perror("pipe()");
        exit(EXIT_FAILURE);
    }
    flags = fcntl(pipefd[WRITE_END], F_GETFL);
    fcntl(pipefd[WRITE_END], F_SETFL, flags | O_NONBLOCK); /* 非阻塞 write */
    flags = fcntl(pipefd[READ_END], F_GETFL);
    fcntl(pipefd[READ_END], F_SETFL, flags); /* 阻塞 read */

    pid = fork(); /* fork a child process */
    if(pid < 0) {
        perror("fork()");
        exit(EXIT_FAILURE);
    }
```

## Examples



- **Linux：管道**
  - UNIX/Linux中的无名管道
    - 算法 10-3: pipe-ord-1.c(2)

```
    if(pid > 0) { /* parent process */
        while (1) {
            printf("Enter some text: ");
            fgets(write_msg, BUFSIZ, stdin);
            write(pipefd[WRITE_END], write_msg, BUFSIZ);
            if (strncmp(write_msg, "end", 3) == 0)
                break;
        }
    }
    else { /* child process */
        while (1) {
            read(pipefd[READ_END], read_msg, BUFSIZ);
            printf("\n\t\t\t\t\tpipe read = %s", read_msg);
            if(strncmp(read_msg, "end", 3) == 0)
                break;
        }
    }

    wait(0);
    close(pipefd[WRITE_END]);
    close(pipefd[READ_END]);
    exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - UNIX/Linux中的无名管道
    - 算法 10-3: pipe-ord-1.c(2)

```
isscgy@ubuntu:/mnt/os-2020$ gcc alg.10-3-pipe-ord-1.c
isscgy@ubuntu:/mnt/os-2020$ ./a.out
Enter some text: hello world
                                        pipe read = hello world
Enter some text: good morning
                                        pipe read = good morning
Enter some text: end
                                        pipe read = end
isscgy@ubuntu:/mnt/os-2020$
```

```
        }
    }

    wait(0);
    close(pipefd[WRITE_END]);
    close(pipefd[READ_END]);
    exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - UNIX/Linux中的无名管道
    - 算法 10-3: pipe-ord-2.c(1)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#define READ_END 0
#define WRITE_END 1
```

```
/* 非阻塞读取版本 */
int main(void)
{
    char write_msg[BUFSIZ]; /* BUFSIZ = 8192bytes, saved from stdin */
    char read_msg[BUFSIZ];
    int pipefd[2]; /* pipefd[0] for READ_END, pipefd[1] for WRITE_END */
    int flags;
    pid_t pid;

    if(pipe(pipefd) == -1) { /* create a pipe */
        perror("pipe()");
        exit(EXIT_FAILURE);
    }
    flags = fcntl(pipefd[WRITE_END], F_GETFL);
    fcntl(pipefd[WRITE_END], F_SETFL, flags | O_NONBLOCK); /* 非阻塞写 */
    flags = fcntl(pipefd[READ_END], F_GETFL);
    fcntl(pipefd[READ_END], F_SETFL, flags | O_NONBLOCK); /* 非阻塞读 */

    pid = fork(); /* fork a child process */
    if(pid < 0) {
        perror("fork()");
        exit(EXIT_FAILURE);
    }
```

## Examples



- **Linux：管道**
  - UNIX/Linux中的无名管道
    - 算法 10-3: pipe-ord-2.c(2)

```
    if(pid > 0) { /* parent process */
        while (1) {
            printf("Enter some text: ");
            fgets(write_msg, BUFSIZ, stdin);
            write(pipefd[WRITE_END], write_msg, BUFSIZ);
            if(strncmp(write_msg, "end", 3) == 0)
                break;
        }
    }
    else { /* child process */
        while (1) {
            ret = read(pipefd[READ_END], read_msg, BUFSIZ);
                /* success: ret = 8192; failure: ret = -1 */
            if(ret > 0) {
                printf("\n%*spipe read = %s", 40, " ", read_msg);
                if(strncmp(read_msg, "end", 3) == 0)
                    break;
            }
            else //sleep(1); /* check every second */
        }
    }

    wait(0);
    close(pipefd[WRITE_END]); close(pipefd[READ_END]);
    exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - UNIX/Linux中的无名管道
    - 算法 10-3: pipe-ord-2.c(2)

```
isscgy@ubuntu:/mnt/os-2020$ gcc alg.10-3-pipe-ord-2.c
isscgy@ubuntu:/mnt/os-2020$ ./a.out
Enter some text: hello world
                                        pipe read = hello world
Enter some text: good morning
                                        pipe read = good morning
Enter some text: end
                                        pipe read = end
isscgy@ubuntu:/mnt/os-2020$
```

```
                    break;
                }
                else //sleep(1); /* check every second */
            }
        }

        wait(0);
        close(pipefd[WRITE_END]); close(pipefd[READ_END]);
        exit(EXIT_SUCCESS);
    }
```

## Examples



- **Linux：管道**
  - 父进程和子进程之间的命名管道
    - 算法 10-4: pipe-nam.c (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <sys/stat.h>

#define TEXT_SIZE 1024
```

```
int main(int argc, char *argv[])
{
    char write_msg[TEXT_SIZE];
    char read_msg[TEXT_SIZE];
    char fifoname[80];
    int fdw, fdr;
    pid_t pid;

    if(argc < 2) {
        printf("Usage: ./a.out pathname\n");
        return EXIT_FAILURE;
    }
        /* pathname can not in current directory */
    strcpy(fifoname, argv[1]);
    if(access(fifoname, F_OK) == -1) {
        if(mkfifo(fifoname, 0666) != 0) { /* creat a named pipe */
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else
            printf("new fifo %s created …\n", fifoname);
    }
```

## Examples



- **Linux：管道**
  - **父进程和子进程之间的命名管道**
    - **算法** 10-4: pipe-nam.c (2)

```
pid = fork(); /* fork a child process */
if(pid < 0) {
    perror("fork()");
    exit(EXIT_FAILURE);
}
if(pid > 0) { /* parent process */
    fdw = open(fifoname, O_WRONLY); /* 阻塞写 */
    if(fdw < 0)
        perror("pipe write open()");
    else {
         while (1) {
            printf("Enter some text: ");
            fgets(write_msg, TEXT_SIZE, stdin);
            write(fdw, write_msg, TEXT_SIZE);
            if(strncmp(write_msg, "end", 3) == 0)
                break;
            sleep(1);
        }
    }
}
```

## Examples



- **Linux：管道**
  - **父进程和子进程之间的命名管道**
    - **算法** 10-4: pipe-nam.c (3)

```
else { /* child process */
     fdr = open(fifoname, O_RDONLY);  /* 阻塞读 */
     if(fdr < 0)
         perror("pipe read open()");
     else {
         while (1) {
             read(fdr, read_msg, TEXT_SIZE);
             printf("\n\t\t\t\t\tpipe read_end = %s", read_msg);
             if(strncmp(read_msg, "end", 3) == 0)
                 break;
         }
     }
}

wait(0);

close(fdw);
close(fdr);
unlink(fifoname);

exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - 父进程和子进程之间的命名管道
    - 算法 10-4: pipe-nam.c (3)

```
isscgy@ubuntu:/mnt/os-2020$ gcc alg.10-4-pipe-nam.c
isscgy@ubuntu:/mnt/os-2020$ ./a.out /tmp/mypipe
Enter some text: Hello World!

                                        pipe read_end = Hello World!
Enter some text: Goodmorning

                                        pipe read_end = Goodmorning
Enter some text: end

                                        pipe read_end = end
isscgy@ubuntu:/mnt/os-2020$ ls -l /tmp/mypipe
ls: cannot access '/tmp/mypipe': No such file or directory
isscgy@ubuntu:/mnt/os-2020$
```

```
            close(fdw);
            close(fdr);
            unlink(fifoname);

            exit(EXIT_SUCCESS);
        }
```

## Examples



- **Linux：管道**
  - 父进程和子进程之间的命名管道
    - 算法 10-4: pipe-nam.c (3)

```
isscgy@ubuntu:/mnt/os-2020$ gcc alg.10-4-pipe-nam.c
isscgy@ubuntu:/mnt/os-2020$ ./a.out /tmp/mypipe
Enter some text: Hello World!

                                        pipe read_end = Hello World!
Enter some text: Goodmorning

                                        pipe read_end = Goodmorning
Enter some text: end

                                        pipe read_end = end
isscgy@ubuntu:/mnt/os-2020$ ls -l /tmp/mypipe
ls: cannot access '/tmp/mypipe': No such file or directory
isscgy@ubuntu:/mnt/os-2020$
```

```
            close(fdw);
            close(fdr);
            unlink(fifoname);

            exit(EXIT_SUCCESS);
        }
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法** 10-5: pipe-nam-write1.c (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>

#define TEXT_SIZE 1024
```

```
/* 读写版本 */
int main(int argc, char *argv[])
{
    char fifoname[80], write_msg[TEXT_SIZE];
    int fdw;

    if(argc < 2) {
        printf("Usage: ./a.out pathname\n");
        return EXIT_FAILURE;
    }
    strcpy(fifoname, argv[1]);
    if(access(fifoname, F_OK) == -1) {
        if(mkfifo(fifoname, 0666) != 0) {
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else
            printf("new fifo %s created ...\n", fifoname);
    }

    fdw = open(fifoname, O_RDWR); /* 非阻塞发送与接收 */
//  fdw = open(fifoname, O_WRONLY);
        /* 阻塞发送，等待接收端接收 */
//  fdw = open(fifoname, O_WRONLY | O_NONBLOCK);
        /* 非阻塞发送，如果接收端未准备好则返回错误！ */
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法** 10-5: pipe-nam-write1.c (2)

```
    if(fdw < 0) {
        perror("pipe write open()");
        exit(EXIT_FAILURE);
    }
    else {
        while (1) {
            printf("\nEnter some text: ");
            fgets(write_msg, TEXT_SIZE, stdin);
            write(fdw, write_msg, TEXT_SIZE); /* 非阻塞写 */
            if (strncmp(write_msg, "end", 3) == 0)
                break;
            sleep(1);
          }
    }

    close(fdw);
    exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-6:** pipe-nam-write2.c (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>

#define TEXT_SIZE 1024
```

```
/* 只写且非阻塞发送版本 */
int main(int argc, char *argv[])
{
    char fifoname[80], write_msg[TEXT_SIZE];
    int fdw;

    if(argc < 2) {
        printf("Usage: ./a.out pathname\n");
        return EXIT_FAILURE;
    }
    strcpy(fifoname, argv[1]);
    if(access(fifoname, F_OK) == -1) {
        if(mkfifo(fifoname, 0666) != 0) {
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else
            printf("new fifo %s named pipe created\n", fifoname);
    }
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-6:** pipe-nam-write2.c (2)

```
    int count = 10;
    while (count) {
        fdw = open(fifoname, O_WRONLY | O_NONBLOCK);
        /* 非阻塞发送，如果接收端未准备好则返回错误！ */
        if(fdw < 0) {
            printf("waiting for receiver ... %d\n", count);
            sleep(1);
            /* do something, and query again, or exit(EXIT_FAILURE) when time out  */
            count--;
        }
        else
            break;
    }

    while (count) {
        printf("\nEnter some text: ");
        fgets(write_msg, TEXT_SIZE, stdin);
        write(fdw, write_msg, TEXT_SIZE); /* 非阻塞写 */
        if (strncmp(write_msg, "end", 3) == 0)
            break;
        sleep(1);
    }

    close(fdw);
    exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-7:** pipe-nam-read.c (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>

#define TEXT_SIZE 1024
```

```
/* 阻塞读取版本 */
int main(int argc, char *argv[])
{
    char fifoname[80], read_msg[TEXT_SIZE];
    int fdr;

    if(argc < 2) {
        printf("Usage: ./a.out pathname\n");
        return EXIT_FAILURE;
    }
    strcpy(fifoname, argv[1]);
    if(access(fifoname, F_OK) == -1) {
        if (mkfifo(fifoname, 0666) != 0) {
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else
            printf("new fifo %s named pipe created\n", fifoname);
    }
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-7:** pipe-nam-read.c (2)

```
    fdr = open(fifoname, O_RDONLY);  /* 阻塞读 */
    if (fdr < 0) {
        perror("pipe read open()");
        exit(EXIT_FAILURE);
    }
    else {
        while (1) {
            read(fdr, read_msg, TEXT_SIZE);
            printf("\npipe read_end = %s",read_msg);
            if (strncmp(read_msg, "end", 3) == 0)
                break;
        }
    }
    close(fdr);
    exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **终端1运行**write1.o **（非阻塞写）**

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-5-pipe-nam-write1.o /tmp/myfifo

Enter some text: hello world

Enter some text: end
isscgy@ubuntu:/mnt/os-2020$
```

    - **终端2运行**read.o**（阻塞读）**

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-7-pipe-nam-read.o /tmp/myfifo

pipe read_end = hello world

pipe read_end = end
isscgy@ubuntu:/mnt/os-2020$
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **终端1运行** write2.o**（只写且非阻塞发送版本）**

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-6-pipe-nam-write2.o /tmp/myfifo
waiting for receiver ... 10
waiting for receiver ... 9
waiting for receiver ... 8
waiting for receiver ... 7
waiting for receiver ... 6

Enter some text: hello world

Enter some text: end
isscgy@ubuntu:/mnt/os-2020$
```

    - **终端2运行**read.o**（阻塞读）**

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-7-pipe-nam-read.o /tmp/myfifo

pipe read_end = hello world

pipe read_end = end
isscgy@ubuntu:/mnt/os-2020$
```

## Examples



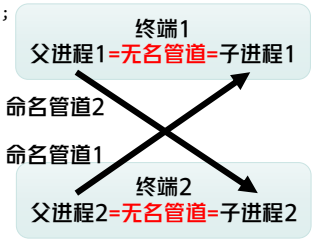
- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-8:** pipe-nam-rdwr1.c (1)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/wait.h>

#define TEXT_SIZE 1024
```

```
/* 在两个任意进程间创建两个命名管道进行通信       */
/* 终端1运行 ./a.out pathname 1
   终端2运行 ./a.out pathname 2                  */
/* 在父进程（写）与子进程（读）间创建两个无名管道
   建立连接，以通知进程结束 */
int main(int argc, char *argv[])
{
    char fifoname_1[80], fifoname_2[80];
    char write_msg[TEXT_SIZE], read_msg[TEXT_SIZE];
    int fdr, fdw, ret;
    pid_t pid;
    int pipefd1[2], pipefd2[2], flags; char msg_str[2];

    if(argc < 3) {
        printf("Usage: ./a.out pathname 1|2\n");
        return EXIT_FAILURE;
    }
    if(pipe(pipefd1) == -1) {/* 创建无名管道 */
        perror("pipe()");
        exit(EXIT_FAILURE);
    }
    flags = fcntl(pipefd1[1], F_GETFL);
    fcntl(pipefd1[1], F_SETFL, flags | O_NONBLOCK);
    flags = fcntl(pipefd1[0], F_GETFL);
    fcntl(pipefd1[0], F_SETFL, flags | O_NONBLOCK);
```




## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-8:** pipe-nam-rdwr1.c (2)

```
    if(pipe(pipefd2) == -1) {  /* 创建无名管道 */
        perror("pipe()");
        exit(EXIT_FAILURE);
    }
    flags = fcntl(pipefd2[1], F_GETFL);
    fcntl(pipefd2[1], F_SETFL, flags | O_NONBLOCK);
    flags = fcntl(pipefd2[0], F_GETFL);
    fcntl(pipefd2[0], F_SETFL, flags | O_NONBLOCK);

    strcpy(fifoname_1, argv[1]); strcat(fifoname_1,"-1");
    strcpy(fifoname_2, argv[1]); strcat(fifoname_2,"-2");
    if(access(fifoname_1, F_OK) == -1) {
        if((mkfifo(fifoname_1, 0666)) != 0) {/* 创建命名管道 */
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else printf("new fifo %s created ...\n", fifoname_1);
    }
    if(access(fifoname_2, F_OK) == -1) {
        if((mkfifo(fifoname_2, 0666)) != 0) {/* 创建命名管道 */
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else printf("new fifo %s created ...\n", fifoname_2);
    }
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-8:** pipe-nam-rdwr1.c (3)

```
printf("\n==== pipe write end ====          ==== pipe read end ====\n");
pid = fork();
if(pid < 0) {
    perror("fork()");
    exit(EXIT_SUCCESS);
}
if(pid == 0) {  /* 子进程读命名管道 */
    if(argv[2][0] == '1')
        fdr = open(fifoname_1, O_RDONLY | O_NONBLOCK);
    else fdr = open(fifoname_2, O_RDONLY | O_NONBLOCK);
    if(fdr < 0)
        perror("fdr open()");
    else
        while (1) {
            ret = read(fdr, read_msg, TEXT_SIZE); /* 非阻塞读 */
            if(ret > 0) {
                printf("\n%*.s%s", 40, " ", read_msg);
                if(strncmp(read_msg, "end", 3) == 0)
                    break;
            }
            ret = read(pipefd2[0], msg_str, 2);
            if(ret > 0 && msg_str[0] == '1')
                break;
        }
    write(pipefd1[1], "1", 2);
    exit(0);
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-8:** pipe-nam-rdwr1.c (4)

```
} else {  /* 父进程写命名管道 */
    if(argv[2][0] == '1')
        fdw = open(fifoname_2, O_RDWR);
    else fdw = open(fifoname_1, O_RDWR);
    if(fdw < 0)
        perror("fdw open()");
    else
        while (1) {
            printf("\n");
            fgets(write_msg, TEXT_SIZE, stdin);
            ret = write(fdw, write_msg, TEXT_SIZE); /* 非阻塞写 */
            if(ret <= 0)
                break;
            if(strncmp(write_msg, "end", 3) == 0)
                break;
            ret = read(pipefd1[0], msg_str, 2);
            if(ret > 0 && msg_str[0] == '1')
                break;
        }
    write(pipefd2[1], "1", 2);
}
wait(0);
close(fdr); close(fdw);
close(pipefd1[1]); close(pipefd1[0]); close(pipefd2[1]); close(pipefd2[0]);
exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - 两个任意进程之间的命名管道
    - 终端1运行 rdwr1.o

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-8-pipe-nam-rdwr1.o /tmp/myfifo 1

==== pipe write end ====          ==== pipe read end ====

hello world

                                        goodmorning

                                        end

isscgy@ubuntu:/mnt/os-2020$
```

    - 终端2运行 rdwr1.o

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-8-pipe-nam-rdwr1.o /tmp/myfifo 2

==== pipe write end ====          ==== pipe read end ====

                                        hello world

goodmorning

end
isscgy@ubuntu:/mnt/os-2020$
```

## Examples



- **Linux：管道**
  - 两个任意进程之间的命名管道
    - 终端1运行rdwr1.o

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-8-pipe-nam-rdwr1.o /tmp/myfifo 1

==== pipe write end ====          ==== pipe read end ====

hello world

                                        goodmorning

                                        end

isscgy@ubuntu:/mnt/os-2020$
```

    - 终端2运行 rdwr1.o

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-8-pipe-nam-rdwr1.o /tmp/myfifo 2

==== pipe write end ====          ==== pipe read end ====

                                        hello world

goodmorning

end
isscgy@ubuntu:/mnt/os-2020$
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-8:** pipe-nam-rdwr2.c (1)

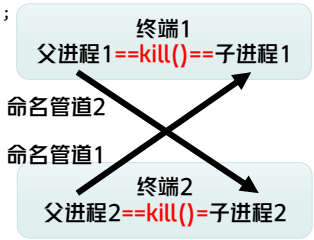
```
/* 在两个任意进程间创建两个命名管道进行通信       */
/* 终端1运行 ./a.out pathname 1
   终端2运行 ./a.out pathname 2                 */
/* 使用kill(, SIGKILL)终止进程                  */

int main(int argc, char *argv[])
{
    char fifoname_1[80], fifoname_2[80];
    char write_msg[TEXT_SIZE], read_msg[TEXT_SIZE];
    int fdr, fdw, ret;
    pid_t pid;
    int pipefd1[2], pipefd2[2], flags; char msg_str[2];

    if(argc < 3) {
        printf("Usage: ./a.out pathname 1|2\n");
        return EXIT_FAILURE;
    }
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/wait.h>

#define TEXT_SIZE 1024
```




## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法 10-8:** pipe-nam-rdwr2.c (2)

```
    strcpy(fifoname_1, argv[1]); strcat(fifoname_1,"-1");
    strcpy(fifoname_2, argv[1]); strcat(fifoname_2,"-2");
    if(access(fifoname_1, F_OK) == -1) {
        if((mkfifo(fifoname_1, 0666)) != 0) {
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else printf("new fifo %s created ...\n", fifoname_1);
    }
    if(access(fifoname_2, F_OK) == -1) {
        if((mkfifo(fifoname_2, 0666)) != 0) {
            perror("mkfifo()");
            exit(EXIT_FAILURE);
        }
        else printf("new fifo %s created ...\n", fifoname_2);
    }
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法** 10-8: pipe-nam-rdwr2.c (3)

```
printf("\n==== pipe write end ====          ==== pipe read end ====\n");
pid = fork();
if(pid < 0) {
    perror("fork()");
    exit(EXIT_SUCCESS);
}
if(pid == 0) {
    if(argv[2][0] == '1')
        fdr = open(fifoname_1, O_RDONLY);
    else fdr = open(fifoname_2, O_RDONLY);
    if(fdr < 0)
        perror("fdr open()");
    else
        while (1) {
            ret = read(fdr, read_msg, TEXT_SIZE); /* 阻塞读 */
            if(ret <= 0) /* if write-end error */
                break;
            printf("\n%*.s%s", 40, " ", read_msg);
            if(strncmp(read_msg, "end", 3) == 0)
                break;
        }
        kill(getppid(), SIGKILL);
        exit(0);
```

## Examples



- **Linux：管道**
  - **两个任意进程之间的命名管道**
    - **算法** 10-8: pipe-nam-rdwr2.c (4)

```
} else {
    if(argv[2][0] == '1')
        fdw = open(fifoname_2, O_RDWR);
    else fdw = open(fifoname_1, O_RDWR);
    if(fdw < 0)
        perror("fdw open()");
    else
        while (1) {
            printf("\n");
            fgets(write_msg, TEXT_SIZE, stdin);
            ret = write(fdw, write_msg, TEXT_SIZE); /* 非阻塞写 */
            if(ret <= 0)
                break;
            if(strncmp(write_msg, "end", 3) == 0)
                break;
        }
        kill(pid, SIGKILL);
}
wait(0);
close(fdr);
close(fdw);
exit(EXIT_SUCCESS);
}
```

## Examples



- **Linux：管道**
  - 两个任意进程之间的命名管道
    - 终端1 运行rdwr2.o

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-8-pipe-nam-rdwr2.o /tmp/myfifo 2

==== pipe write end ====          ==== pipe read end ====

hello world

                                        goodmorning

                                        end

Killed
isscgy@ubuntu:/mnt/os-2020$
```

    - 终端2运行rdwr2.o

```
isscgy@ubuntu:/mnt/os-2020$ ./alg.10-8-pipe-nam-rdwr2.o /tmp/myfifo 1

==== pipe write end ====          ==== pipe read end ====

                                        hello world
goodmorning

end
isscgy@ubuntu:/mnt/os-2020$
```

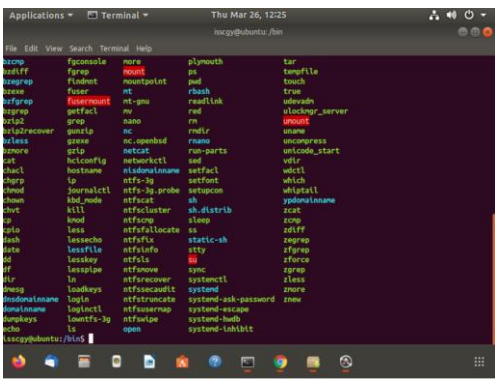
## Examples



- **Linux：管道**
  - UNIX CLI中的管道
    - 可以在UNIX命令行上使用|字符构造管道。完整的命令是

      ls | less

    - ls和less命令作为单个进程运行。ls的输出作为less的输入。
    - ls的结果：



## Examples



- **Linux：管道**
  - UNIX CLI中的管道
    - 可以在UNIX命令行上使用|字符构造管道。完整的命令是

      ls | less
    - ls和less命令作为单个进程运行。ls的输出作为less的输入。
    - ls | less的结果：