

深圳大学 计算机与软件学院

College of Computer Science and Software Engineering of Shenzhen University



系统编程

基于TaiShan服务器/openEuler OS 的实践

第二讲：进程间通信 – 命名管道

找bug...

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

int main(int argc, char *argv[])
{
    int i,cid[5];
    pid_t pid;

    pid = fork();
    cid[0] = pid;
    for (i = 1; i < 5; i++) {
        if (pid > 0) {
            pid = fork();
            cid[i] = pid;
        } else if (pid == 0){
            sleep(1);
            printf("Hello from child process %d. \n",getpid());
            return 0;
        } else {
            perror("Fail to fork a child.\n");
            return 0;
        }
    }
    printf("Hello from parent process with pid %d. \n",getpid());
    for (i = 0; i < 5; i++) {
        waitpid(cid[i],NULL,0);
        printf("Child process %d exited. \n",cid[i]);
    }
    printf("Parent process exit. \n");
    return 0;
}
```

```
[yuhong@FedoraDVD13 C程序]$ gcc -o findXILIeyes findXILIeyes.c
[yuhong@FedoraDVD13 C程序]$ ./findXILIeyes
Hello from parent process with pid 2525.
Hello from parent process with pid 2530.
Child process 2526 exited.
Child process 2527 exited.
Child process 2528 exited.
Child process 2529 exited.
Child process 0 exited.
Parent process exit.
Hello from child process 2529.
Hello from child process 2528.
Hello from child process 2526.
Hello from child process 2527.
Child process 2526 exited.
Child process 2527 exited.
Child process 2528 exited.
Child process 2529 exited.
Child process 2530 exited.
Parent process exit.
[yuhong@FedoraDVD13 C程序]$
```

两个...parent
process...?



叶家健分享的代码，请指出问题

```
[yejiajian2018132005@taishan01-vm-7 ex2]$ cat txt.txt
[yejiajian2018132005@taishan01-vm-7 ex2]$ cat try1.c
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(){
    FILE *fd;
    fd = fopen("txt.txt", "a");
    if(fd != 0){
        perror("fd\n");
        exit(23);
    }
    printf("printf__hahahhahahah\n");
    fprintf(fd, "fprintf__helloworld__beforedup2\n");
    dup2(fileno(fd),1);
    printf("printf__hahahhahahah_after_dup2\n");
    fprintf(fd, "fprintf__helloworld__after_dup2\n");
    fclose(fd);
    close(1);
    return 0;
}
```

```
[yejiajian2018132005@taishan01-vm-7 ex2]$ make try1
cc      try1.c  -o try1
try1.c: 在函数‘main’中:
try1.c:12:3: 警告: 隐式声明函数‘exit’ [-Wimplicit-function-declaration]
    exit(23);
    ^~~~
try1.c:12:3: 警告: 隐式声明与内建函数‘exit’不兼容
try1.c:12:3: 附注: include ‘<stdlib.h>’ or provide a declaration of ‘exit’
[yejiajian2018132005@taishan01-vm-7 ex2]$ ./try1
printf__hahahhahahah
[yejiajian2018132005@taishan01-vm-7 ex2]$ cat txt.txt
printf__hahahhahahah_after_dup2
fprintf__helloworld__beforedup2
fprintf__helloworld__after_dup2
```

```
2. 172.31.234.200 (szu)
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    FILE *fd;
    fd = fopen("text.txt","a");
    if (fd < 0) {
        perror("fopen text.txt error\n");
        exit(0);
    }
    printf("printf Hello World 1\n");
    fprintf(fd,"Hello world 2\n");
    dup2(fileno(fd),1);
    printf("Hello world 3\n");
    fprintf(fd,"Hello world 4\n");
    fclose(fd);
    close(1);
    return 0;
}
```

```
[szu@taishan02-vm-10 fifo]$ ./stdout2file2
printf Hello World 1
[szu@taishan02-vm-10 fifo]$ cat text.txt
Hello world 3
Hello world 2
Hello world 4
```

修改代码以更清晰的方式再现问题

```
2. 172.31.234.200 (szu)
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>

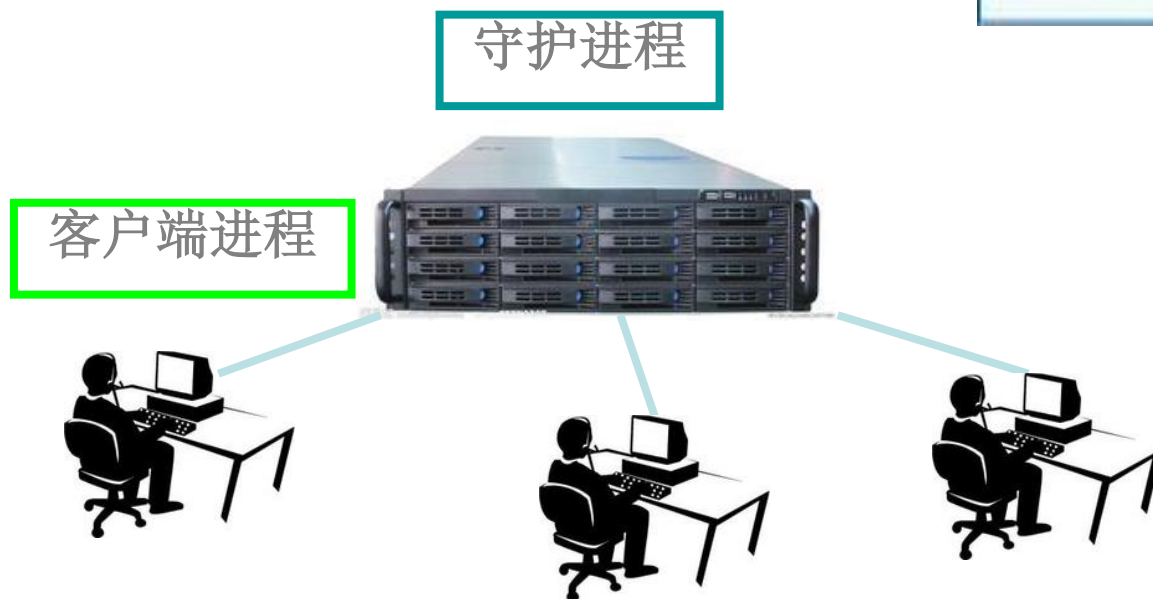
int main()
{
    FILE *fd;
    fd = fopen("text.txt","a");
    if (fd < 0) {
        perror("fopen text.txt error\n");
        exit(0);
    }
    printf("printf Hello World 1\n");
    fprintf(fd,"Hello world 2\n");
    fflush(fd);
    dup2(fileno(fd),1);
    printf("Hello world 3\n");
    fprintf(fd,"Hello world 4\n");
    fflush(fd);
    fclose(fd);
    close(1);
    return 0;
}
```

```
[szu@taishan02-vm-10 fifo]$ ./stdout2file3
printf Hello World 1
[szu@taishan02-vm-10 fifo]$ cat text.txt
Hello world 2
Hello world 3
Hello world 4
```

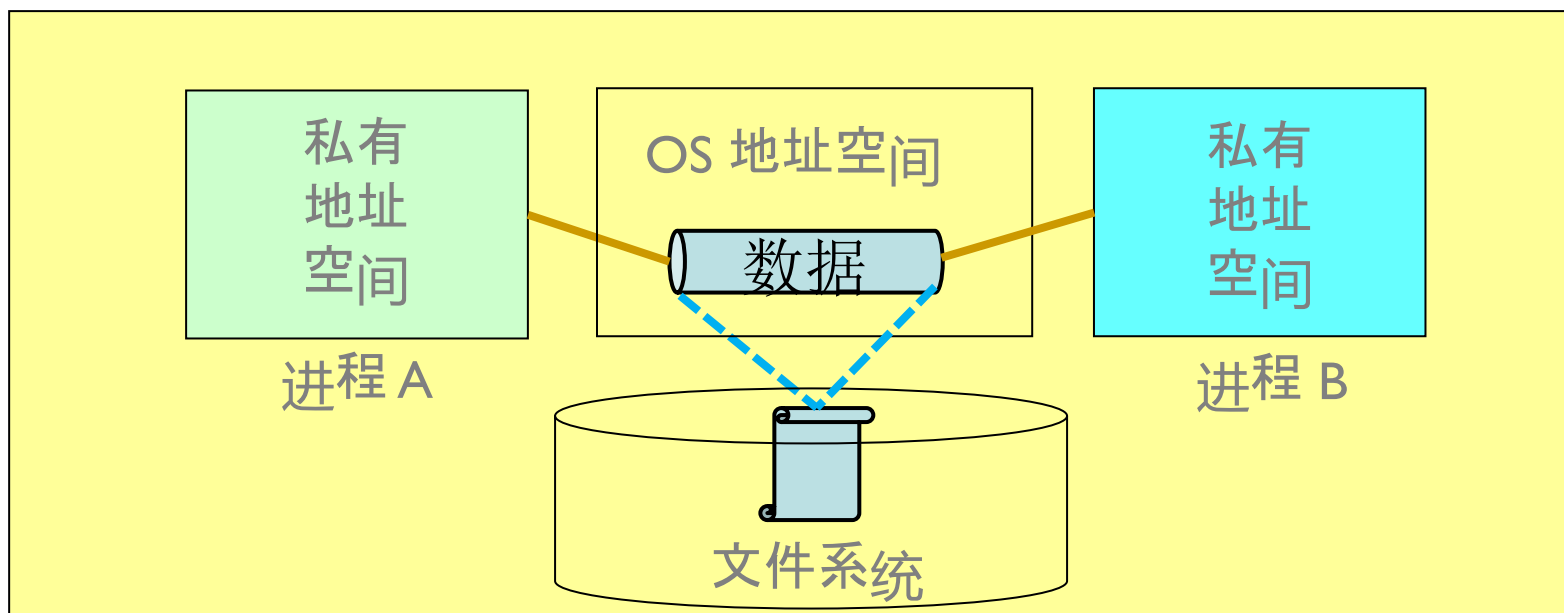
再修改代码以解决问题

管道的局限性

- 只用于创建管道的进程与其子孙进程间的通信
- 非持久化



通过命名管道通信



- 通信数据存在内核，先进先出
 - 不支持诸如lseek()等文件定位操作
- 在文件系统中路径与之相关联
- 可设置权限（Owner, Group, Other) (rwx) (rwx) (rwx)

创建FIFO的函数

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
int mkfifo(const char *pathname, mode_t mode);
```

pathname: 文件路径名 mode: 设定读写权限

返回值: 成功为0, 失败为-1

相关函数, 学有余力可以自己探索...

```
mknod(const char *pathname, mode_t mode, dev_t dev)
```

- 普通文件, 块文件, 字符型文件, 命名管道
- S_IRUSR | S_IWUSR | S_IFBLK | S_IFCHR | S_IFIFO

相关命令, 学有余力可以自己探索...

```
$mknod myfifo p
```


命名管道的读

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <fcntl.h>
```

```
int open(const char *pathname, int flags);
```

“flags” 参数

- O_RDONLY: 进程阻塞, 直到有进程以写方式打开管道
- O_WRONLY: 进程阻塞, 直到有进程以读方式打开管道
- O_RDWR: 进程不阻塞
- O_RDONLY|O_NONBLOCK: 进程不阻塞、读不阻塞
 - 有写者进程返回-1, 并设置errno为EAGAIN
 - 无写者进程返回0
- O_WRONLY|O_NONBLOCK:
 - 没有读者进程, 写进程退出 (SIGPIPE)
 - 读进程退出, 写操作导致写进程退出 (SIGPIPE)

这程序有错，你细看看，问题在哪？

```
1 #include "ClientInfo.h"
2
3 #define FIFO_NAME "tmp/server_test"
4 #define BUFF_SZ 100
5 char mypipename[BUFF_SZ];
6 CLIENTINFO userinfo, recvinfo;
7
8 int main()
9 {
10     int res;
11     int fifo_fd, my_fifo, fd;
12     char buffer[BUFF_SZ];
13
14     char send[20], rec[20];
15
16     if(access(FIFO_NAME, F_OK)==-1)
17     {
18         printf("Could not open FIFO %s\n", FIFO_NAME);
19         exit(EXIT_FAILURE);
20     }
21     fifo_fd = open(FIFO_NAME, O_WRONLY);
22     if(fifo_fd == -1)
23     {
24         printf("Could not open %s for write access \n", FIFO_NAME);
25         exit(EXIT_FAILURE);
26     }
27     printf("Your name please:");
28     scanf("%s", userinfo.username);
29     printf("The receiver please:");
30 }
```

"Client.c" 61L, 1235C 1,1 Top

命名管道应用例子（一）

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
```

创建一个命名管道，从其读数据即从标准输入读

```
int main(int argc, char** argv) {
    mkfifo(argv[1], S_IRWXU | S_IRWXG | S_IRWXO);

    int fifo = open(argv[1], O_RDONLY);

    dup2(fifo, 0); /* 0 is the file descriptor of stdin */

    char line[1024];
    while (fgets(line, 1024, stdin))
        printf("I got this: %s\n", line);
}
```

仿照上面的例子，改写Client/Server

1. Server从管道读数据，如从标准输入读
2. Client往管道写数据，如从标准输出写

学生分享的代码 (2)

```
client.c  server.c  server.c
1 #include <sys/types.h>
2 #include <sys/stat.h>
3 #include <fcntl.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <string.h>
7 #include <errno.h>
8 #include <unistd.h>
9 #define MSGSIZE 65
10 char *fifo = "myfifo";
11
12 int main(int argc, char *argv[]){
13     int fd, i, nwrite;
14     char msgbuf[MSGSIZE+1];
15
16     if(argc>2) {printf("Usage: receive message ... \n"); exit
17     if(mkfifo(fifo, 0666) == -1){
18         if(errno!=EEXIST) {perror("receive:mkfifo"); exit(6);}
19     }
20
21
22     if( (fd = open(fifo, O_RDWR)) < 0){
23         perror("fifo open problem");
24         exit(3);
25     }
26     dup2(fd, 0);
27     for(;;){
28         scanf("%s", msgbuf);
29         //if(read(fd, msgbuf, MSGSIZE+1)<0){
30         //    perror("problem in reading"); exit(5);
31         //}
32         printf("\n Message Received:%s \n", msgbuf);
33         fflush(stdout);
34     }
35 }
```

```
server.c  client.c
1 #include <sys/types.h>
2 #include <sys/stat.h>
3 #include <fcntl.h>
4 #include <unistd.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8 #include <error.h>
9 #define MSGSIZE 65
10 char *fifo = "myfifo";
11
12 int main(int argc, char *argv[]){
13     int fd, i, nwrite;
14     char msgbuf[MSGSIZE+1];
15
16     if(argc<2) {printf("Usage: send a message ... \n"); exit(1);};
17
18     if( (fd = open(fifo, O_WRONLY|O_NONBLOCK)) < 0){
19         perror("fifo open error");
20         exit(1);
21     }
22     dup2(fd, 1); // cout >>
23     for(i=1;i<argc; i++){
24         if(strlen(argv[i])>MSGSIZE){
25             printf("Message with Prefix %.*s Too long - Ignored\n", 10, argv[i]);
26             fflush(stdout);
27             continue;
28         }
29         strcpy(msgbuf, argv[i]);
30         printf("Message:%s\n", msgbuf);
31         // if((nwrite=write(fd, msgbuf, MSGSIZE+1))!= -1){
32         //     perror("Error in Writing"); exit(2);
33         // }
34     }
35     exit(0);
36 }
```

相关系统调用:

mkfifo,

open/close

read/write,

scanf/sscanf/printf

fopen/fclose

fgets/fprintf

fread/fwrite

fscanf/sscanf/fprintf

```
4. 172.31.234.200 (szu) (1)
Re-attach Fullscreen Stay on top Duplicate Hide toolbar Clos

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

#define MSGSIZE 1024

char *fifo = "msgchannel";

int main(int argc, char *argv[]){
    int fd;
    char msgbuf[MSGSIZE+1];

    printf("Please input message line by line:\n");

    if ((fd = open(fifo, O_WRONLY)) < 0){perror("fail to open fifo\n");exit(1);}
    dup2(fd, 1);
    for ( ; ; ){
        read(0,msgbuf,MSGSIZE+1);
        write(fd, msgbuf,MSGSIZE);
    }
}
```

```
2. 172.31.234.200 (szu) 4. 172.31.234.200 (szu) (1)
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

#define MSGSIZE 1024

char *fifo = "msgchannel";

int main(int argc, char *argv[]){
    int fd, nread;
    char msgbuf[MSGSIZE+1];

    if (access(fifo,R_OK|W_OK) != 0){
        printf("the file does not exist..\n");
        if (mkfifo(fifo, 0666) == -1) {perror("Fail to creat fifo\n"); exit(0);}
    }
    if ((fd = open(fifo, O_RDONLY)) < 0){perror("Fail to open fifo\n");exit(1);}
    dup2(fd, 0);
    for ( ; ; ){
        nread = read(fd,msgbuf,MSGSIZE+1);
        if (nread < 0 ) {perror("Fail to read fifo\n");exit(2);}
        if (msgbuf[0]=='\0') break;
        printf("\n Here is a message: %s\n",msgbuf);
        msgbuf[0]='\0';
    }
}
```

代码出错了。。。你将怎么做

- Debug
- 插入printf
- 写log
- 单元测试

C语言单元测试框架—Check（例子源代码）

<http://blog.chinaunix.net/uid-20147410-id-85936.html>

C编码规范

<https://wenku.baidu.com/view/02f313c15fbfc77da269b121.html>

谷歌代码规范

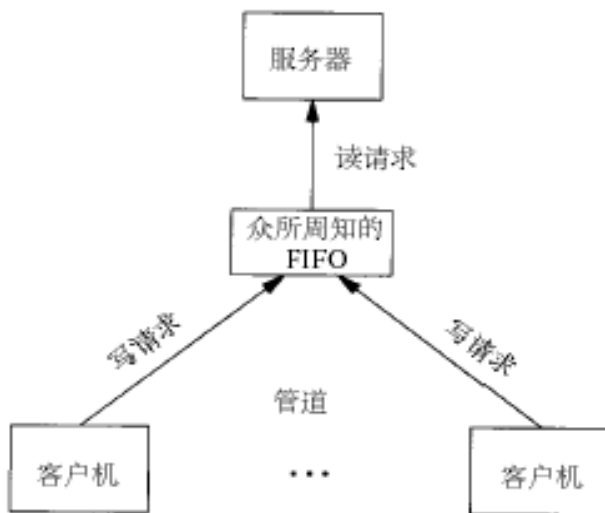
<https://blog.csdn.net/doubleintfloat/article/details/86552296>

命名管道的应用(二)

■ 简单的客户端-服务器架构

● 客户端(Client)

◆ 通过服务器命名管道发送消息到服务器然后退出



客户机用FIFO向服务器发送请求

命名管道的应用(二)

■ 简单的客户端-服务器架构

- 服务器(Server)

- ◆ 后台进程

- ◆ 从命名管道中读取客户端的消息

- ◆ 将客户端的消息输出到标准输出


```
yuhong@FedoraDVD13:~/C程序/fifo
文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#define MSGSIZE 65

char *fifo="myfifo";

int main(int argc, char *argv[])
{
    int fd, i, nwrite;
    char msgbuf[MSGSIZE+1];

    if (argc>2){
        printf("Usage: receive message & \n"); exit(1);
    }

    if ( mkfifo(fifo,0666) == -1){
        if (errno!= EEXIST) { perror("receiver: mkfifo"); exit(6);}
    }

    if ( (fd = open(fifo, O_RDWR))<0){
        perror("fifo open problem"); exit(3);
    }

    for (;;) {
        if ( read(fd,msgbuf,MSGSIZE+1) < 0){
            perror("problem in reading"); exit(5);
        }
        printf("\n Message Received: %s\n",msgbuf);
        fflush(stdout);
    }
}
```

server.c

文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <error.h>
#define MSGSIZE 65
char *fifo = "myfifo";

int main(int argc, char *argv[]){
    int fd, i, nwrite;
    char msgbuf[MSGSIZE+1];

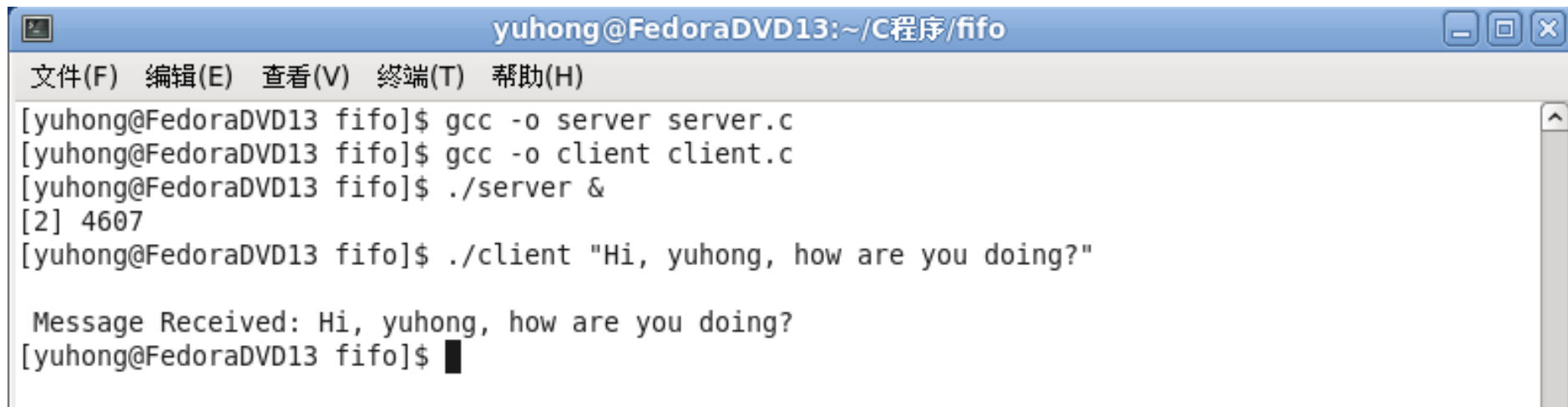
    if (argc<2) { printf("Usage: send a message ... \n"); exit(1);}

    if ( (fd = open(fifo, O_WRONLY | O_NONBLOCK)) < 0){ perror("fifo open error"); exit(1);}

    for (i = 1; i < argc; i++){
        if (strlen(argv[i])>MSGSIZE){
            printf("Message with Prefix %.*s Too long - Ignored\n",10,argv[i]);
            fflush(stdout);
            continue;
        }
        strcpy(msgbuf,argv[i]);
        if ((nwrite=write(fd,msgbuf,MSGSIZE+1))==-1){
            perror("Error in Writing"); exit(2);
        }
    }
    exit(0);
}
```

client.c

运行



A terminal window titled "yuhong@FedoraDVD13:~/C程序/fifo" with a menu bar containing "文件(F)", "编辑(E)", "查看(V)", "终端(T)", and "帮助(H)". The terminal shows the following commands and output:

```
[yuhong@FedoraDVD13 fifo]$ gcc -o server server.c
[yuhong@FedoraDVD13 fifo]$ gcc -o client client.c
[yuhong@FedoraDVD13 fifo]$ ./server &
[2] 4607
[yuhong@FedoraDVD13 fifo]$ ./client "Hi, yuhong, how are you doing?"

Message Received: Hi, yuhong, how are you doing?
[yuhong@FedoraDVD13 fifo]$
```

命名管道的应用(三)

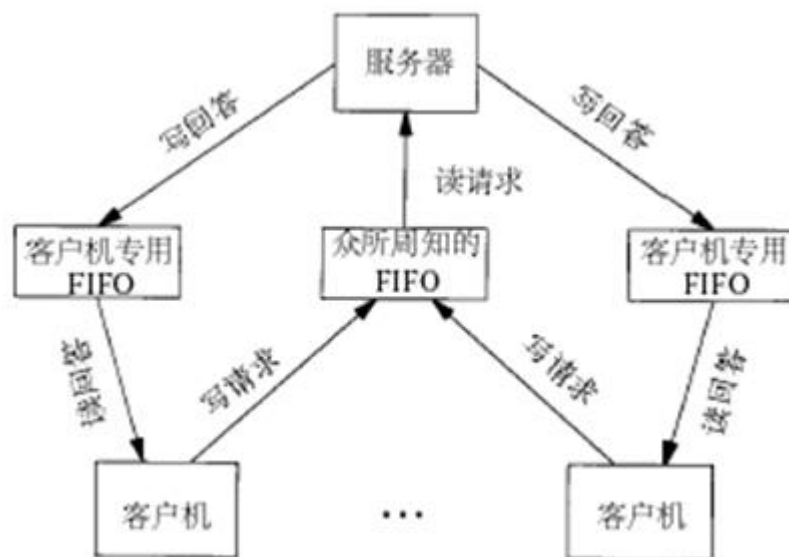
■ 简单的客户端-服务器架构

● 服务器(Server)

◆ 守护进程

◆ 从命名管道中读取客户端的消息

◆ 将客户端的消息输出到客户端



命名管道的应用(三)

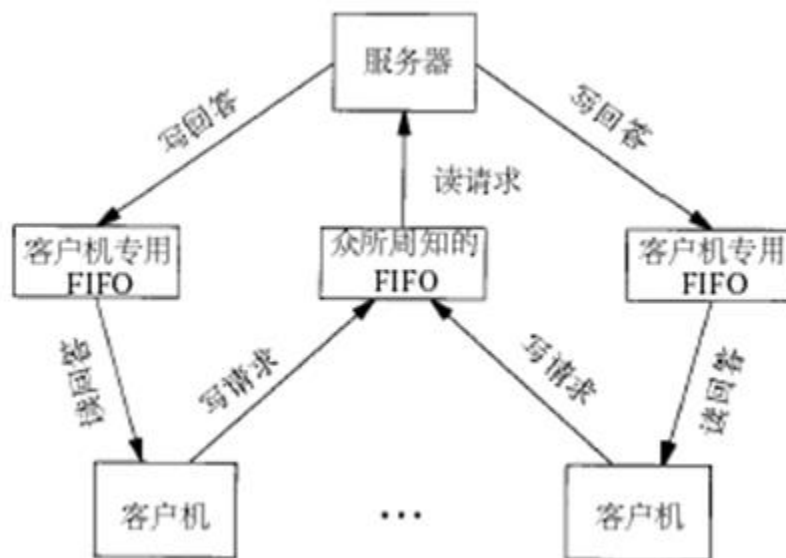
■ 简单的客户端-服务器架构

● 服务器(Server)

◆ 守护进程

◆ 从命名管道中读取客户端的消息

◆ 将客户端的消息输出到客户端



clientinfo.h

```
/* clientinfo.h */

#ifndef _CLIENTINFO_H
#define _CLIENTINFO_H

typedef struct {
    char myfifo[50]; /* client's FIFO name */
    int leftarg;      /* left argument of calculation */
    int rightarg;     /* right argument of calculation */
    char op;          /* operation: + - "*" / */
} CLIENTINFO, *CLIENTINFOPTR;

#endif
```

客户端发送给服务器的请求的消息格式

client.c - 1

```
/* client.c */

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <fcntl.h>
#include <signal.h>
#include "clientinfo.h"

#define FIFO_NAME  "/tmp/server_fifo"
#define BUFF_SZ  100

char mypipename[BUFF_SZ];

void handler(int sig) {    /* remove pipe if signaled */
    unlink(mypipename);
    exit(1);
}
```


client.c - 2

```
int main(int argc, char *argv[]) {
    int res;
    int fifo_fd, my_fifo;
    int fd;
    CLIENTINFO info;
    char buffer[BUFF_SZ];

    /* handle some signals */
    signal(SIGKILL, handler);
    signal(SIGINT, handler);
    signal(SIGTERM, handler);

    /* check for proper command line */
    if (argc != 4) {
        printf("Usage: %s op1 operation op2\n", argv[0]);
        exit(1);
    }
}
```

client.c - 3

```
/* check if server fifo exists */
if (access(FIFO_NAME, F_OK) == -1) {
    printf("Could not open FIFO %s\n", FIFO_NAME);
    exit(EXIT_FAILURE);
}

/* open server fifo for write */
fifo_fd = open(FIFO_NAME, O_WRONLY);
if (fifo_fd == -1) {
    printf("Could not open %s for write access\n",
        FIFO_NAME);
    exit(EXIT_FAILURE);
}
```

client.c - 4

```
/* create my own FIFO */
sprintf(mypipename, "/tmp/client%d_fifo", getpid());
res = mkfifo(mypipename, 0777);
if (res != 0) {
    printf("FIFO %s was not created\n", buffer);
    exit(EXIT_FAILURE);
}

/* open my own FIFO for reading */
my_fifo = open(mypipename, O_RDONLY | O_NONBLOCK);
if (my_fifo == -1) {
    printf("Could not open %s for read only access\n",
        FIFO_NAME);
    exit(EXIT_FAILURE);
}
```

client.c - 5

```
/* construct client info */
strcpy(info.myfifo, mypipename);
info.leftarg = atoi(argv[1]);
info.op = argv[2][0];
info.rightarg = atoi(argv[3]);

/* write client info to server fifo */
write(fifo_fd, &info, sizeof(CLIENTINFO));
close(fifo_fd);
```

client.c - 6

```
/* get result from server */
memset(buffer, '\0', BUFF_SZ);
while (1) {
    res = read(my_fifo, buffer, BUFF_SZ);
    if (res > 0) {
        printf("Received from server: %s\n", buffer);
        break;
    }
}

printf("Client %d is terminating\n", getpid());

/* delete fifo from system */
close(my_fifo);
(void)unlink(mypipename);

exit(0);
}
```

calcservice.c - 1

```
/* calcservice.c */

#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <fcntl.h>
#include "clientinfo.h"

#define FIFO_NAME  "/tmp/server_fifo"

void handler(int sig) {
    unlink(FIFO_NAME);
    exit(1);
}
```

calcservice.c - 2

```
int calc(CLIENTINFOPTR info) {
    switch(info->op) {
        case '+': return info->leftarg + info->rightarg;
        case '-': return info->leftarg - info->rightarg;
        case '*': return info->leftarg * info->rightarg;
        case '/': return info->leftarg / info->rightarg;
    }
    return 0;
}

int main() {
    int res;
    int i;
    int fifo_fd, fd1;
    CLIENTINFO info;
    char buffer[100];
```


calcservice.c - 3

```
signal(SIGKILL, handler);
signal(SIGINT, handler);
signal(SIGTERM, handler);

/* create FIFO, if necessary */
if (access(FIFO_NAME, F_OK) == -1) {
    res = mkfifo(FIFO_NAME, 0777);
    if (res != 0) {
        printf("FIFO %s was not created\n", FIFO_NAME);
        exit(EXIT_FAILURE);
    }
}

/* open FIFO for reading */
fifo_fd = open(FIFO_NAME, O_RDONLY);
if (fifo_fd == -1) {
    printf("Could not open %s for read only access\n",
        FIFO_NAME);
    exit(EXIT_FAILURE);
}
```

calcservice.c - 4

```
printf("\nServer is ready to go!\n");

while (1) {
    res = read(fifo_fd, &info, sizeof(CLIENTINFO));
    if (res != 0) {
        printf("Client arrived!!\n");
        sprintf(buffer, "The result is %d", calc(&info));
        fd1 = open(info.myfifo, O_WRONLY | O_NONBLOCK);
        write(fd1, buffer, strlen(buffer)+1);
        close(fd1);
    }
}
exit(0);
}
```

client/calcservice 执行

```
$ ./calcservice &  
[2] 3683  
$ ./client 3 + 45
```

```
Server is ready to go!  
Client arrived!!  
Received from server: The result is 48  
Client 3684 is terminating
```

```
$ ./client 3 "*" 2 &  
[3] 3685
```

```
$ Client arrived!!  
Received from server: The result is 6  
Client 3685 is terminating
```

```
kill 3683
```

```
[3]- Done ./client 3 "*" 2  
$
```

命名管道应用例子（四）：生产者-消费者

- 生产者

- 写FIFO

- 消费者

- 读FIFO
- 将数据输出到文件

- **OS确保FIFO 写操作的原子性**

当有多个生产者怎么办？

■ 场景

- 多个客户端（生产者）
- n类事件请求：注册、登录、发信息、……
- 服务器等待客户端的请求
- 尽快响应任一个客户端的请求

■ 实现方案一：进程

- 建立n个连接(pipe, fifo, socket, …)
- 监听每个连接(read, accept, receive, …) 的请求

◆ 阻塞式

问题：在某个连接上阻塞，即使另一个连接有数据就绪！

◆ 非阻塞式

问题：循环不停地检测，运行效率低，代码可读性差

解决问题需要新的通信方式：等待事件集中任意事件（1个或多个）的发生

select & poll、epoll

select()&poll():等待事件集中任意事件的发生

■ 参数

- 一集文件描述符
- 每个文件描述符对应一集事件
- Timeout长度

■ 返回值

- 一集文件描述符
- 每个描述符对应的事件

■ 注意事项

- Select 相对简单
- Poll 支持等待更多的事件类型
- 一些系统的更新的版本: epoll

select()

```
int select (int num_fds, fd_set* read_set,  
            fd_set* write_set, fd_set* except_set,  
            struct timeval* timeout);
```

- 等待可读/可写的文件描述符
- 返回:
 - 成功: 就绪的文件描述符数
 - 出错: -1, 并设置 **errno**
- 参数:
 - **num_fds**:
 - ◆ 需要监视的文件描述符集中最大的文件描述符+1
 - **read_set, write_set, except_set**:
 - ◆ **select** 函数监视需要监视的文件描述符的变化情况（位矢量， bit vectors）： read、write、或 except
 - ◆ 调用**select**函数后会阻塞，直到有描述符就绪（**fd_set**）或者超时
 - ◆ 返回后，可以通过遍历**fd_set**来找到就绪的描述符
 - **timeout**:
 - ◆ 等待描述符就绪的时间

文件描述符集struct fd_set

■ 数据结构：位矢量（**Bit vectors**），通常是**1024**位

■ 通过预定义宏进行操作

```
fds_set myset;
```

```
void FD_ZERO (&myset);      /*设置所有位为0，清空集合*/
```

```
void FD_SET (n, &myset);    /*设置位n为1，将文件描述符n加入集合*/
```

```
void FD_CLEAR (n, &myset); /*设置位n为0，将文件描述符n从集合中删去*/
```

```
int FD_ISSET (n, &myset); /* 检查集合中文件符n是否可读写? */
```

文件描述符集struct fd_set

■ 要检查的3个条件

- 可读

- ◆ 数据就绪，可读

- 可写

- ◆ 缓冲区空间就绪，可写

- 异常

- ◆ 带外数据就绪（TCP），异常

select() 应用例子

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <unistd.h>
int main()
{
    char buf[100];
    fd_set my_read;          //设置监听集合
    struct timeval tv; //设置等待时间, 0不等待, NULL一直等待
    tv.tv_sec = 5; //秒数
    tv.tv_usec = 0; //微秒数, 秒后面的零头

    FD_ZERO(&my_read);      //清空集合

/*
 * 通常, 一个进程启动时, 都会打开 3 个文件:
 * 标准输入 (0)、标准输出 (1) 和标准出错处理 (2)
 */
/*设置位0为1, 将文件描述符0加入集合*/
    FD_SET(0, &my_read);

    if (select(1, &my_read, NULL, NULL, &tv) == 1) {
        /* data ready on stdin */
        if (FD_ISSET(0, &my_read)){
            fgets(buf, 100, stdin);
            printf("Your input is: %s\n", buf);
        }
    } else {
        printf("Your missed your time ...\n");
    }
}
```

```
[yuhong@FedoraDVD13 IPC]$ gcc -o select_1 select_1.c
[yuhong@FedoraDVD13 IPC]$ ./select_1
Hellowtimeout ...
[yuhong@FedoraDVD13 IPC]$ ./select_1
Hello
Your input is: Hello

[yuhong@FedoraDVD13 IPC]$ ./select_1
timeout ...
[yuhong@FedoraDVD13 IPC]$ █
```

poll()

```
#include <poll.h>
```

```
int poll (struct pollfd* pfd, nfds_t nfds, int timeout);
```

- 判断pfd中的文件是否可读。

- 返回值:

- 成功: 可读fd的数量
- 出错: -1, 并且设置 `errno`

- 参数:

- `pfd`:

- ◆ 文件描述符结构体数组, 每个元素包含文件描述符、请求的事件和返回的事件。

- `nfds`:

- ◆ `pfd`数组的长度

- `timeout`:

- ◆ 超时 (毫秒)

文件描述符结构

■ 结构

```
struct pollfd {  
    /* 文件描述符 */  
    int fd;  
    /* 监视该文件描述符的事件位掩码，由用户来设置 */  
    short events;  
    /* 文件描述符的操作结果事件位掩码，内核在调用返回时设置 */  
    short revents;  
}
```

■ 注意事项:

- 任意 $fd < 0$ 的结构都将被忽略

事件标志

■ **POLLIN:**

- 数据就绪，可读

■ **POLLOUT:**

- 缓冲区空间可用，可写

■ **POLLERR:**

- 文件描述符有错误汇报

■ **POLLHUP:**

- 文件描述符挂起（连接断开）

■ **POLLVAL:**

- 文件描述符无效

这些标志不是互斥的，可同时设置，如某文件读和写操作都正常返回。

文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <sys/time.h>
#include <unistd.h>
#include <poll.h>
```

```
int main()
```

```
{
```

```
    char buf[100];
```

```
    struct pollfd my_pfds[1];
```

```
    my_pfds[0].fd = 0;
```

```
    my_pfds[0].events = POLLIN;
```

```
    if (poll(my_pfds, 1, 5000) == 1){
```

```
        if (my_pfds[0].revents & POLLIN) {
```

```
            fgets(buf, 100, stdin);
```

```
            printf("Your input is: %s\n", buf);
```

```
        }
```

```
    } else {
```

```
        printf("timeout ... \n");
```

```
    }
```

```
}
```

poll() 应用例子

文件(F) 编辑(E) 查看(V) 终端(T) 帮助(H)

```
[yuhong@FedoraDVD13 IPC]$ ./poll_1
timeout ...
```

```
[yuhong@FedoraDVD13 IPC]$ ./poll_1
Hello world
```

```
Your input is: Hello world
```

```
[yuhong@FedoraDVD13 IPC]$ █
```


完成作业1-3

■ 将实验3改为用select/poll实现

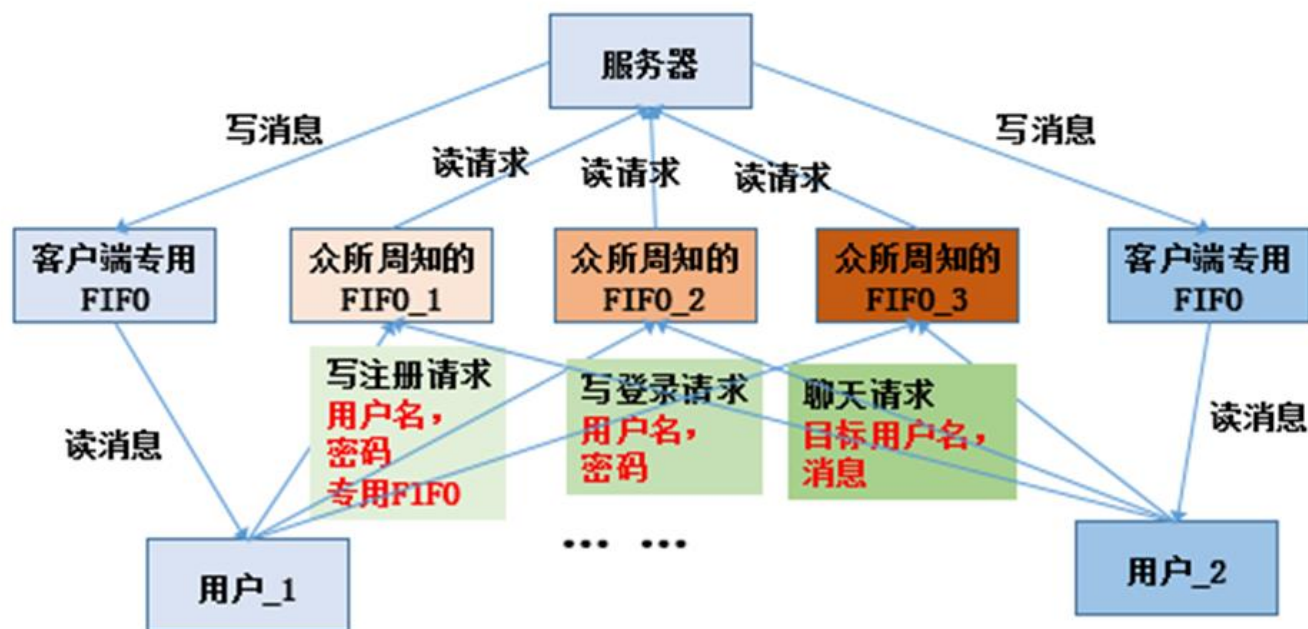


图 6. 多用户聊天系统框架图。

更多命名管道读/写函数

■ 读

```
pfp = fopen( "/tmp/cmd_pipe", "r" );  
...  
ret = fgets( buffer, MAX_LINE, pfp );
```

■ 写

```
pfp = fopen( "/tmp/cmd_pipe", "w+ );  
...  
ret = fprintf( pfp, "Here' s a test string!\n" );
```