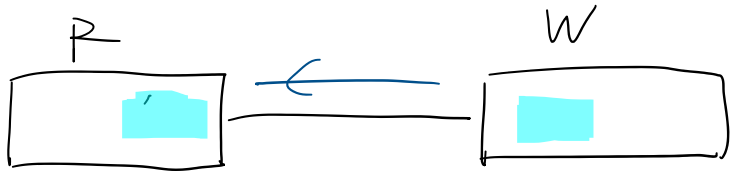
管道的read和write

2023年4月21日 10:35




```c
#include <49tunc.h>
int main(int argc, char *argv[])
{
    // ./4_write_pipe 1.pipe
    ARGS_CHECK(argc,2);
    int fdw = open(argv[1], O_WRONLY);
    ERROR_CHECK(fdw,-1,"open");
    printf("write side is opened!\n");
    //sleep(20);
    //printf("sleep over!\n");
    write(fdw,"hello",5);
    close(fdw);
    return 0;
}
```

W 目前不引发阻塞

```c
#include <49tunc.h>
int main(int argc, char *argv[])
{
    // ./4_read_pipe 1.pipe
    ARGS_CHECK(argc,2);
    int fdr = open(argv[1], O_RDONLY);
    ERROR_CHECK(fdr,-1,"open");
    printf("read side is opened!\n");
    char buf[4096] = {0};
    sleep(20);
    printf("sleep over!\n");
    ssize_t sret = read(fdr,buf,sizeof(buf));
    ERROR_CHECK(sret,-1,"read");
    printf("sret = %ld,buf=%s\n", sret, buf);
    close(fdr);
    return 0;
}
```

如果R缓冲区无数据 读阻塞

如果遇到了close

2023年4月21日 11:43

写端关了，读端调用read，read不会阻塞，read会返回0.

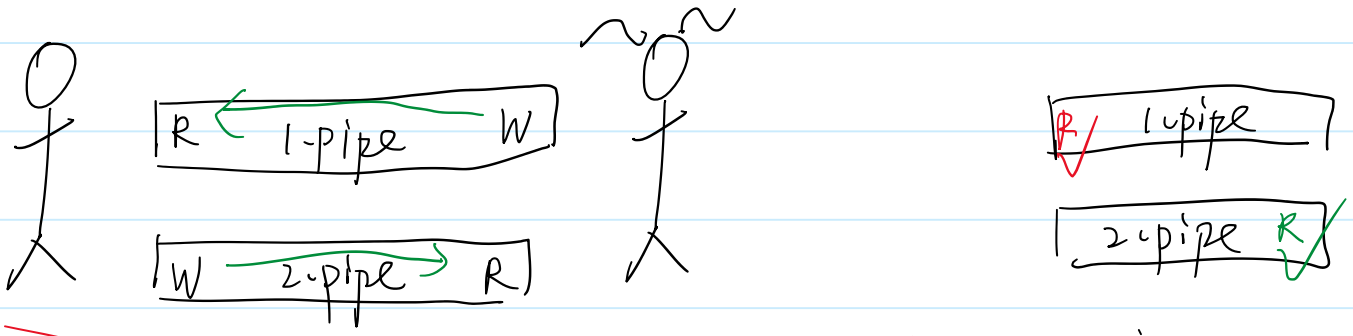
读端关闭，写端调用write，会触发SIGPIPE信号

```
(gdb) r
Starting program: /home/liao/49code/2_Linux/Linuxday_08/4_write_pipe 1.pipe
write side is opened!

Program received signal SIGPIPE, Broken pipe.
0x00007ffff7ed2077 in __GI___libc_write (fd=3, buf=0x55555555602c, nbytes=5)
    at ../sysdeps/unix/sysv/linux/write.c:26
26      ../sysdeps/unix/sysv/linux/write.c: No such file or directory.
```

利用管道实现全双工

2023年4月21日 14:34



死锁

5_aqiang.c

```c
#include <49func.h>
int main(int argc, char *argv[])
{
    // ./5_azhen 2.pipe 1.pipe
    ARGS_CHECK(argc,3);
    int fdr = open(argv[1],O_RDONLY);
    ERROR_CHECK(fdr,-1,"open fdr");
    int fdw = open(argv[2],O_WRONLY);
    ERROR_CHECK(fdw,-1,"open fdw");
    printf("chat is established!\n");
    close(fdw);
    close(fdr);
    return 0;
}
```

5_azhen.c

```c
#include <49func.h>
int main(int argc, char *argv[])
{
    // ./5_aqiang 1.pipe 2.pipe
    ARGS_CHECK(argc,3);
    int fdr = open(argv[1],O_RDONLY);
    ERROR_CHECK(fdr,-1,"open fdr");
    int fdw = open(argv[2],O_WRONLY);
    ERROR_CHECK(fdw,-1,"open fdw");
    printf("chat is established!\n");
    close(fdw);
    close(fdr);
    return 0;
}
```

解决死锁问题

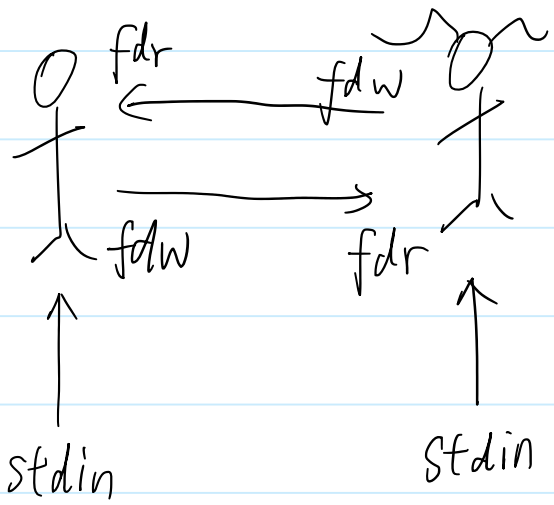
2023年4月21日 14:43

调整获取资源的顺序

```c
// ./5_azhen 1.pipe 2.pipe
ARGS_CHECK(argc,3);
//int fdr = open(argv[1],O_RDONLY);   存在死锁问题
//ERROR_CHECK(fdr,-1,"open fdr");
//int fdw = open(argv[2],O_WRONLY);
//ERROR_CHECK(fdw,-1,"open fdw");
int fdw = open(argv[1],O_WRONLY);
ERROR_CHECK(fdw,-1,"open fdw");
int fdr = open(argv[2],O_RDONLY);
ERROR_CHECK(fdr,-1,"open fdr");
printf("chat is established!\n");
close(fdr);
close(fdw);
return 0;
```

简陋即时聊天

2023年4月21日 14:48



无情发送数据

```c
char buf[4096];
while(1){
    // 读取对面发送的消息
    memset(buf,0,sizeof(buf));
    read(fdr,buf,sizeof(buf));
    printf("buf = %s\n", buf);
    // 阿珍发消息
    memset(buf,0,sizeof(buf));
    read(STDIN_FILENO,buf,sizeof(buf));
    write(fdw,buf,strlen(buf));
}
```

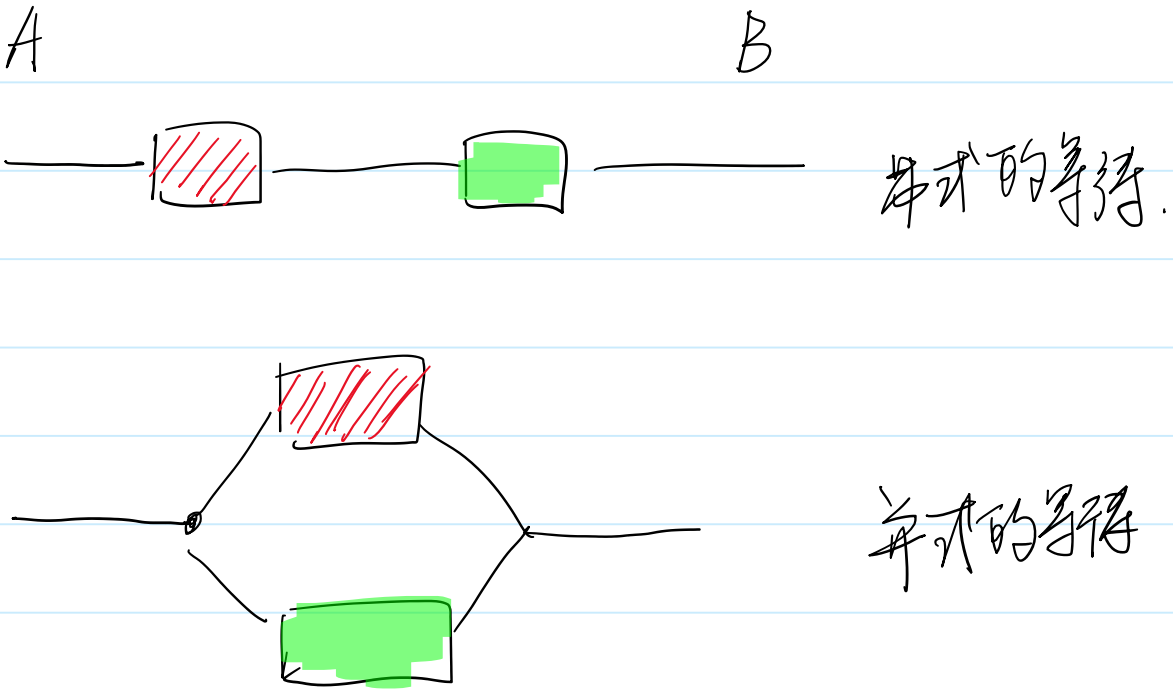
fdr

stdin

```c
12      while(1){
13          // 阿强发消息
14          memset(buf,0,sizeof(buf));
15          read(STDIN_FILENO,buf,sizeof(buf));
16          write(fdw,buf,strlen(buf));
17          // 读取对面发送的消息
18          memset(buf,0,sizeof(buf));
19          read(fdr,buf,sizeof(buf));
20          printf("buf = %s\n", buf);
21      }
22      close(fdw);
```

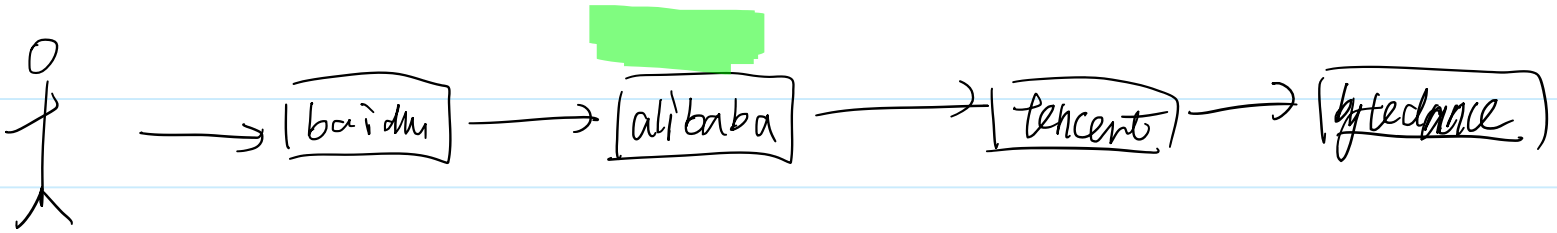
IO多路复用

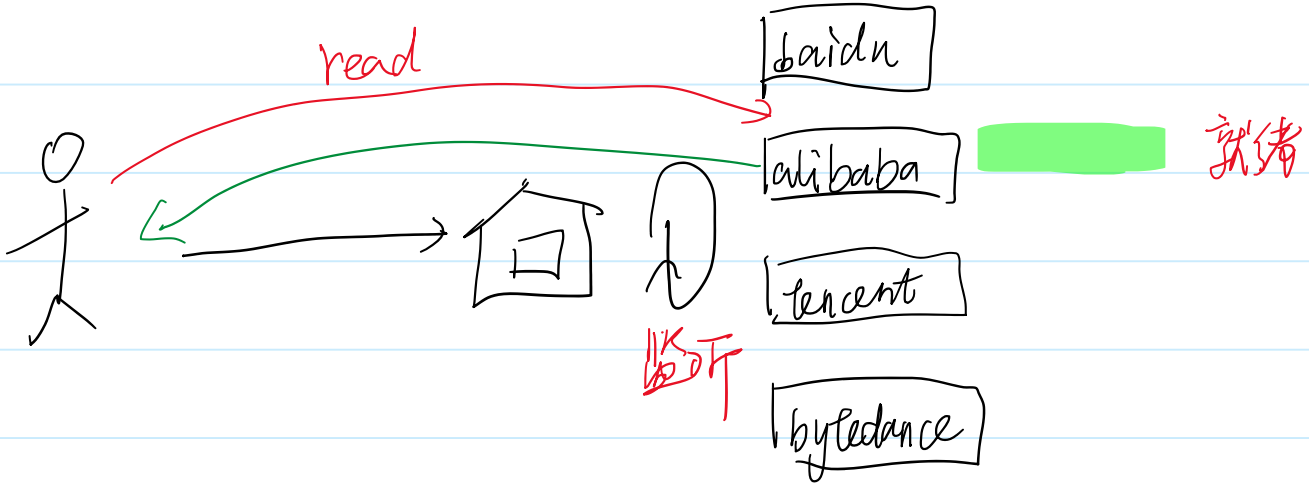
2023年4月21日 14:58



IO多路复用

2023年4月21日 15:01





select

2023年4月21日 15:11

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);

void FD_CLR(int fd, fd_set *set);
int  FD_ISSET(int fd, fd_set *set);
void FD_SET(int fd, fd_set *set);
void FD_ZERO(fd_set *set);
```

最大的fd再+1　读集合　写集合　异常　超时时限 NULL 永久等待

→ select调用会修改指向的内容

fd_set 资源的集合

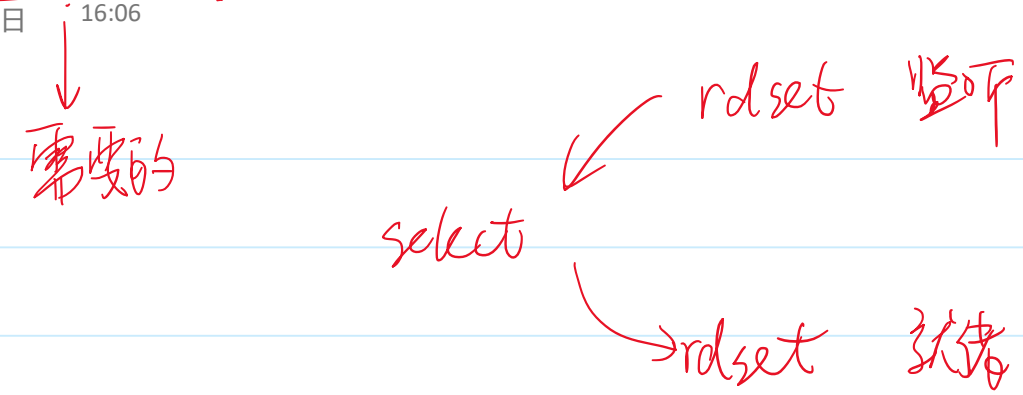
1. 创建监听集合　fd_set rdset;
2. 初始化　FD_ZERO
3. 把关心的fd加入监听.　FD_SET
4. 调用select函数. 本进程陷入阻塞, 内核检查fd是否就绪.
   监听集合 → 就绪集合
   任一fd就绪, select返回
5. 遍历就绪集合　FD_ISSET ⟹ read

2023年4月21日 16:05

```c
char buf[4096];
// 创建一个监听集合
fd_set rdset;
// 初始化
FD_ZERO(&rdset);
// 把管道读端和stdin加入监听
FD_SET(STDIN_FILENO,&rdset);
FD_SET(fdr,&rdset);
// 调用select函数，使进程阻塞
select(fdr+1,&rdset,NULL,NULL,NULL);
// select返回以后，rdset里面是本次的就绪集合
if(FD_ISSET(STDIN_FILENO,&rdset)){
    //stdin就绪
    memset(buf,0,sizeof(buf));
    read(STDIN_FILENO,buf,sizeof(buf));
    write(fdw,buf,strlen(buf));
}
if(FD_ISSET(fdr,&rdset)){
    //fdr就绪
    memset(buf,0,sizeof(buf));
    read(fdr,buf,sizeof(buf));
    printf("buf = %s\n", buf);
}
```

每次循环都重新设置一下监听

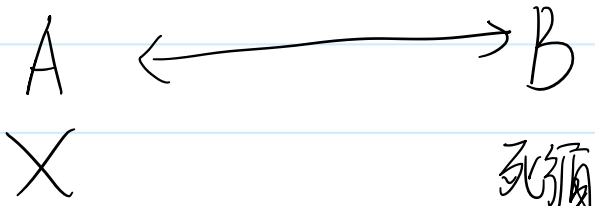
2023年4月21日 16:06




```c
fd_set rdset;
while(1){
    // 初始化
    FD_ZERO(&rdset);
    // 把管道读端和stdin加入监听
    FD_SET(STDIN_FILENO,&rdset);
    FD_SET(fdr,&rdset);
    // 调用select函数，使进程阻塞
    select(fdr+1,&rdset,NULL,NULL,NULL);
    // select返回以后，rdset里面是本次的就绪集合
    if(FD_ISSET(STDIN_FILENO,&rdset)){
        //stdin就绪
        memset(buf,0,sizeof(buf));
        read(STDIN_FILENO,buf,sizeof(buf));
        write(fdw,buf,strlen(buf));
    }
    if(FD_ISSET(fdr,&rdset)){
        //fdr就绪
        memset(buf,0,sizeof(buf));
        read(fdr,buf,sizeof(buf));
        printf("buf = %s\n", buf);
    }
}
```

聊天的关闭

2023年4月21日 16:13



死循环 ← select每次监听fd都为0.

管道写端关闭了，读端处于空的状态，每次用read立刻返回0。

2023年4月21日 16:23

```c
if(FD_ISSET(STDIN_FILENO,&rdset)){
    //stdin就绪
    memset(buf,0,sizeof(buf));
    ssize_t sret = read(STDIN_FILENO,buf,sizeof(buf));
    if(sret == 0){
        write(fdw,"nishigehaoren",13);
        break;
    }
    write(fdw,buf,strlen(buf));
}
if(FD_ISSET(fdr,&rdset)){
    //fdr就绪
    memset(buf,0,sizeof(buf));
    ssize_t sret = read(fdr,buf,sizeof(buf));
    if(sret == 0){
        printf("Hehe!\n");
        break;
    }
    printf("buf = %s\n", buf);
}
```

select的超时机制

2023年4月21日 16:24

timeout 限制最长等待时间

```
int select(int nfds, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout);
```

select返回 { ① 监听的fd就绪 ② timeout超时 } ← 传入传出

may be zero if the timeout expires before anything interesting happens.

```
struct timeval {
    long    tv_sec;         /* seconds */
    long    tv_usec;        /* microseconds */
};
```

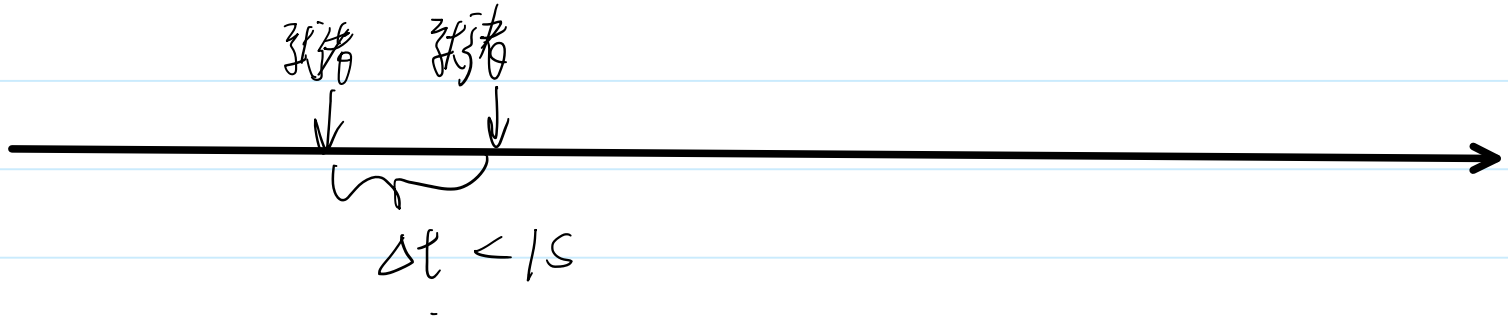
```
timeout.tv_sec = 3;
timeout.tv_usec = 0;
int ret = select(fdr+1,&rdset,NULL,NULL,&timeout);
if(ret == 0){
    time_t now = time(NULL);
    printf("timeout! curtime = %s\n", ctime(&now));
    continue;
}
```

作业的思路

2023年4月21日 16:41

实现即时聊天，如果对方连续10s没有发送任何消息则断开连接。（注意，即使本方在10s内从标准输入当中输入数据也不行）。

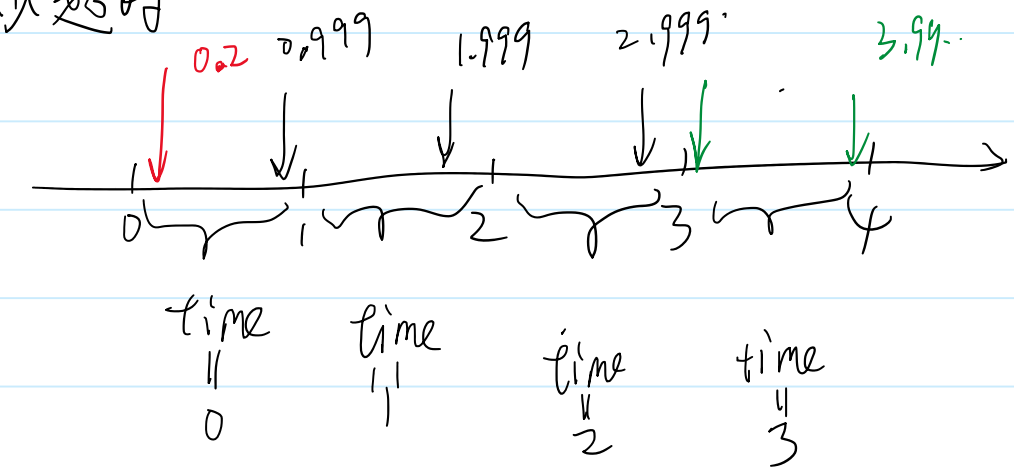
① 超时时间是 10秒



② 每次张伟 记录当前时间

2023年4月21日 16:56

3秒超时



写阻塞

2023年4月21日 17:21

7_read.c buffers

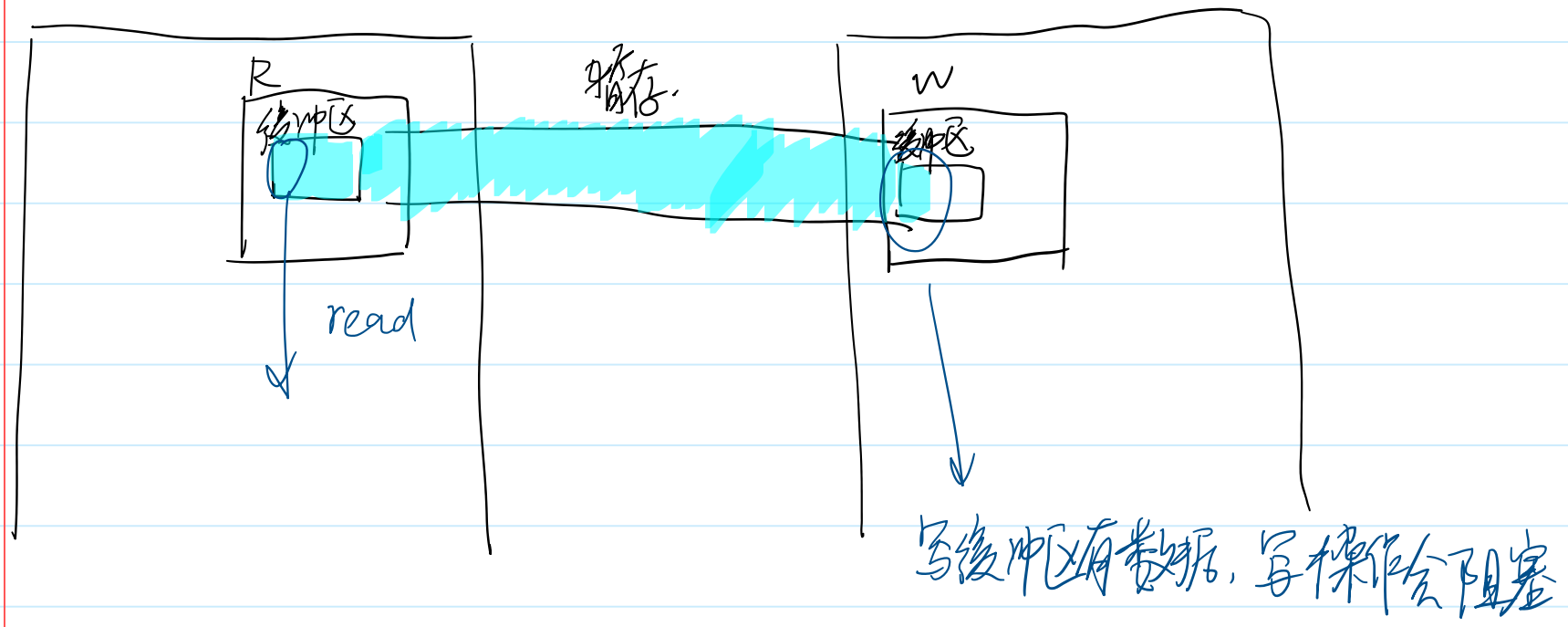
```c
#include <49func.h>
int main(int argc, char *argv[])
{
    // ./7_read 1.pipe
    ARGS_CHECK(argc,2);
    int fdr = open(argv[1],O_RDONLY);
    ERROR_CHECK(fdr,-1,"open");
    while(1){
        sleep(1);
    }
    close(fdr);
    return 0;
}
```

7_write.c

```c
#include <49func.h>
int main(int argc, char *argv[])
{
    // ./7_write 1.pipe
    ARGS_CHECK(argc,2);
    int fdw = open(argv[1],O_WRONLY);
    ERROR_CHECK(fdw,-1,"open");
    char buf[4096] = {0};
    ssize_t total = 0;
    while(1){
        ssize_t sret = write(fdw,buf,sizeof(buf));
        ERROR_CHECK(sret,-1,"write");
        total += sret;
        printf("sret = %ld, total = %ld\n", sret, total);
    }
    return 0;
}
```

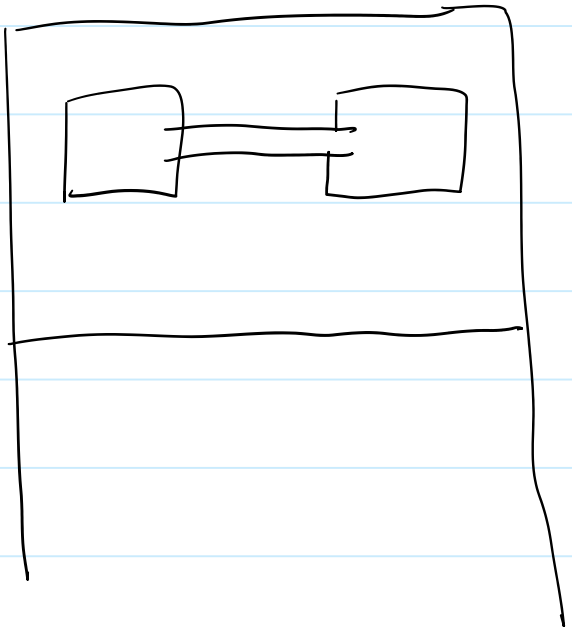
写阻塞的原因

2023年4月21日 17:30



用select 监听写阻塞

2023年4月21日 17:38



open O_RDWR, 非阻塞打开管道一端