组织好代码

2023年5月10日 9:25

```
[liao@ubuntu processPool]$ tree .
.
├── client
│   ├── client
│   ├── client.c
│   └── Makefile
└── server
    ├── head.h
    ├── main.c
    ├── main.o
    ├── Makefile
    ├── server
    ├── worker.c
    └── worker.o
```

```
1 client:client.c
2     gcc client.c -o client -lpthread -g
```

s/Makefile

```
1 SRCS:=main.c worker.c
2 OBJS:=$(SRCS:%.c=%.o)
3 server:$(OBJS)
4     gcc $^ -o $@ -lpthread
5 %.o:%.c
6     gcc -c $^ -o $@ -g
7 clean:
8     $(RM) server $(OBJS)
```

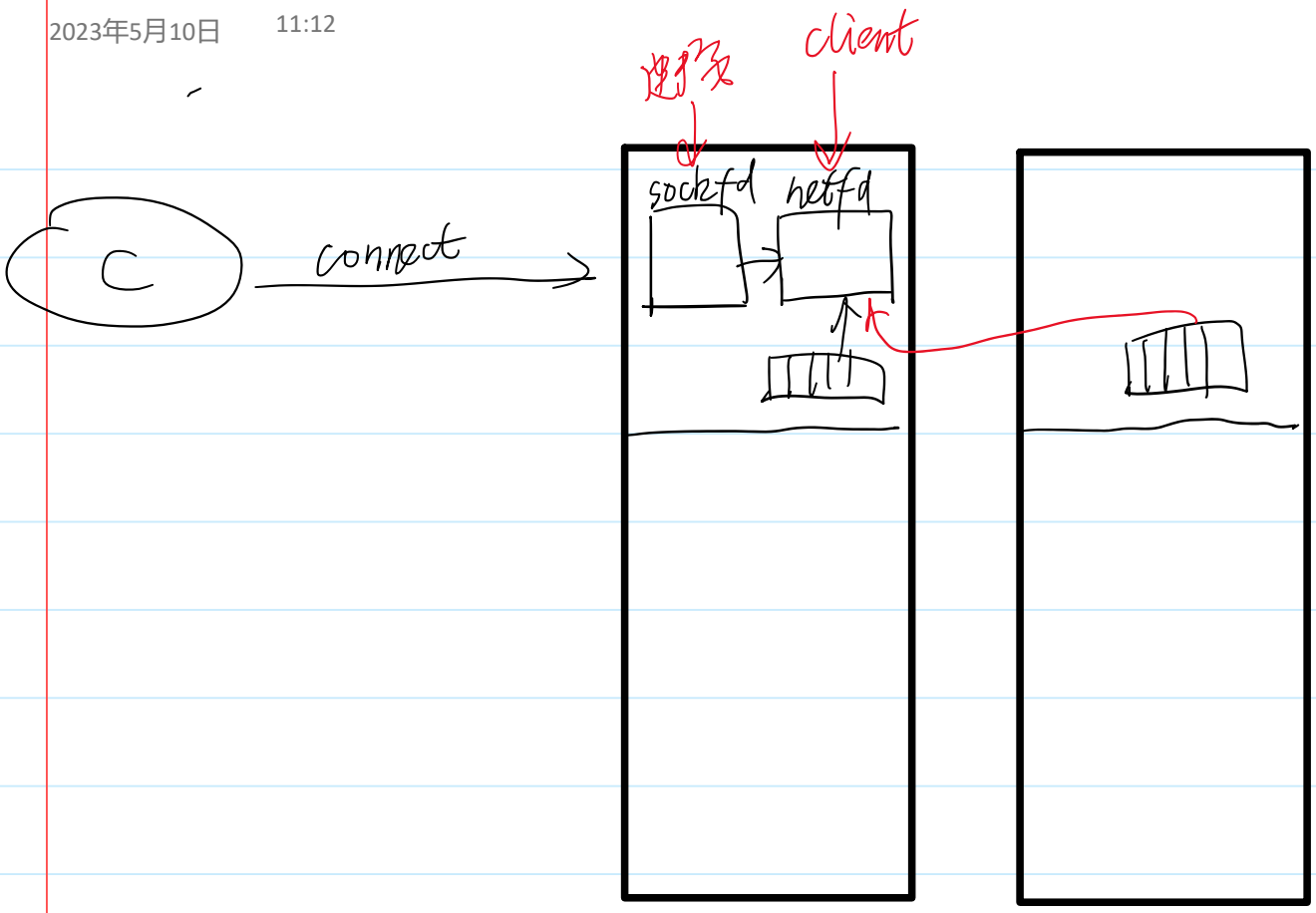
设计数据结构

2023年5月10日 10:19

```
enum {
    FREE,
    BUSY
};
typedef struct workerdata_s{//父进程使用，用来保存每个子进程的信息
    pid_t pid;
    int status;
}workerdata_t;
```

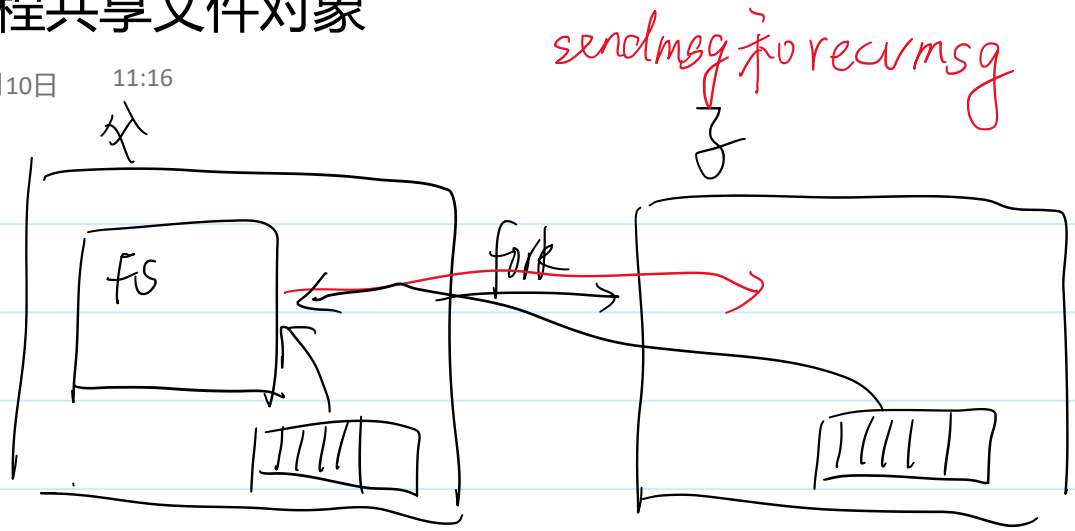
在进程之间传递文件描述符

2023年5月10日 11:12



跨进程共享文件对象

2023年5月10日 11:16




```
ssize_t sendmsg(int sockfd, const struct msghdr *msg, int flags);

ssize_t recvmsg(int sockfd, struct msghdr *msg, int flags);
```

必须用socket

在父子进程间建立本地socket

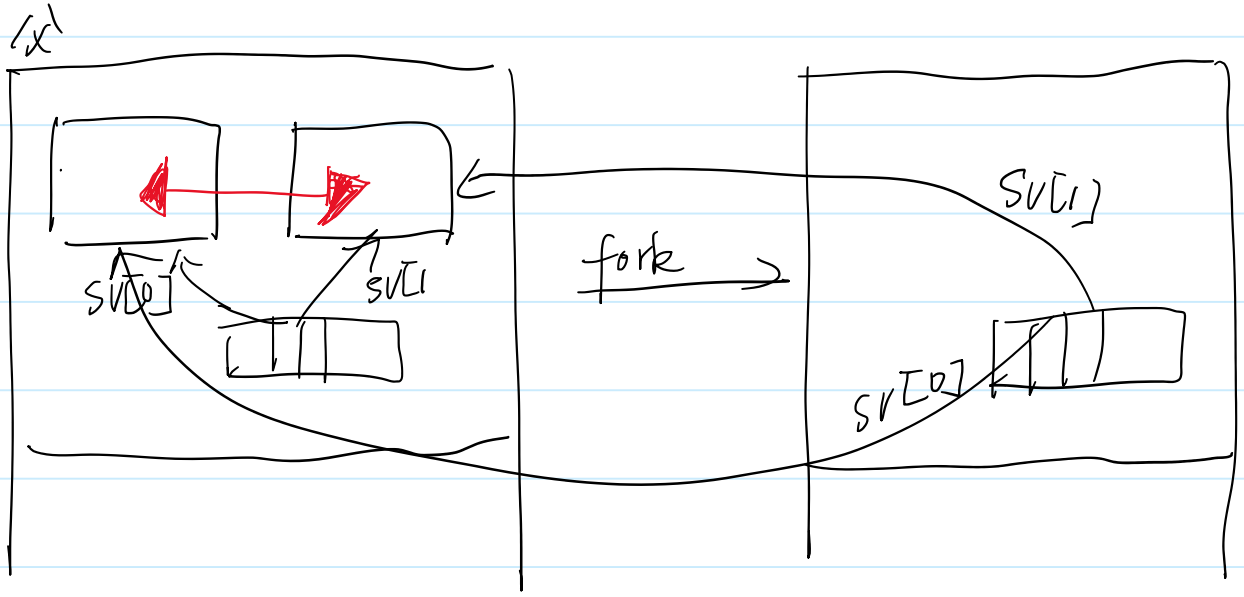
2023年5月10日 11:19

AF_LOCAL　SOCK_STREAM　0

```
int socketpair(int domain, int type, int protocol, int sv[2]);
```

← 可以使用sendmsg和recvmsg

使用方法和 pipe

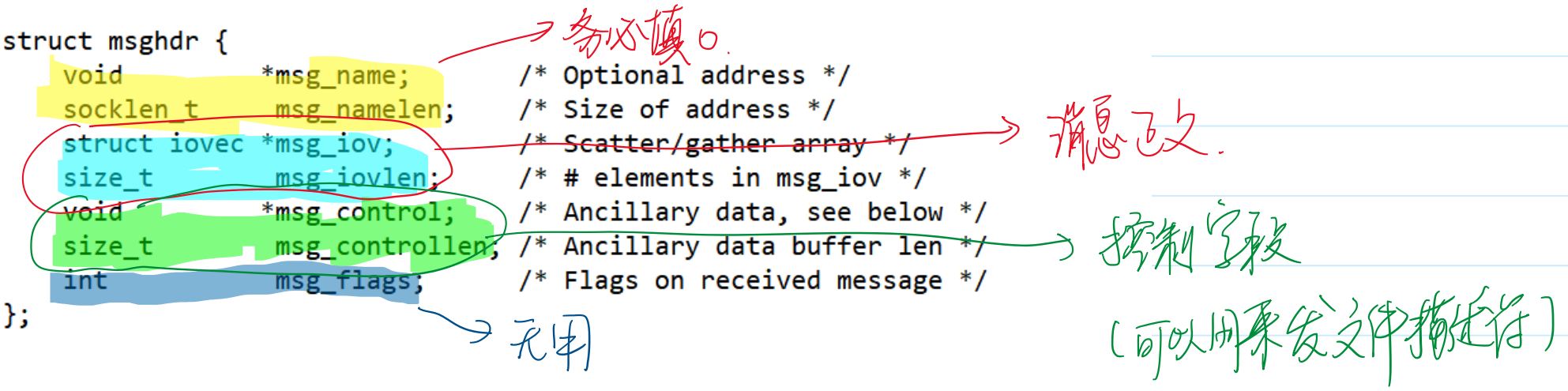


[0]可读可写　[1]可读可写.　[0]写[1]读
[1]写[0]读

struct msghdr

2023年5月10日 11:24

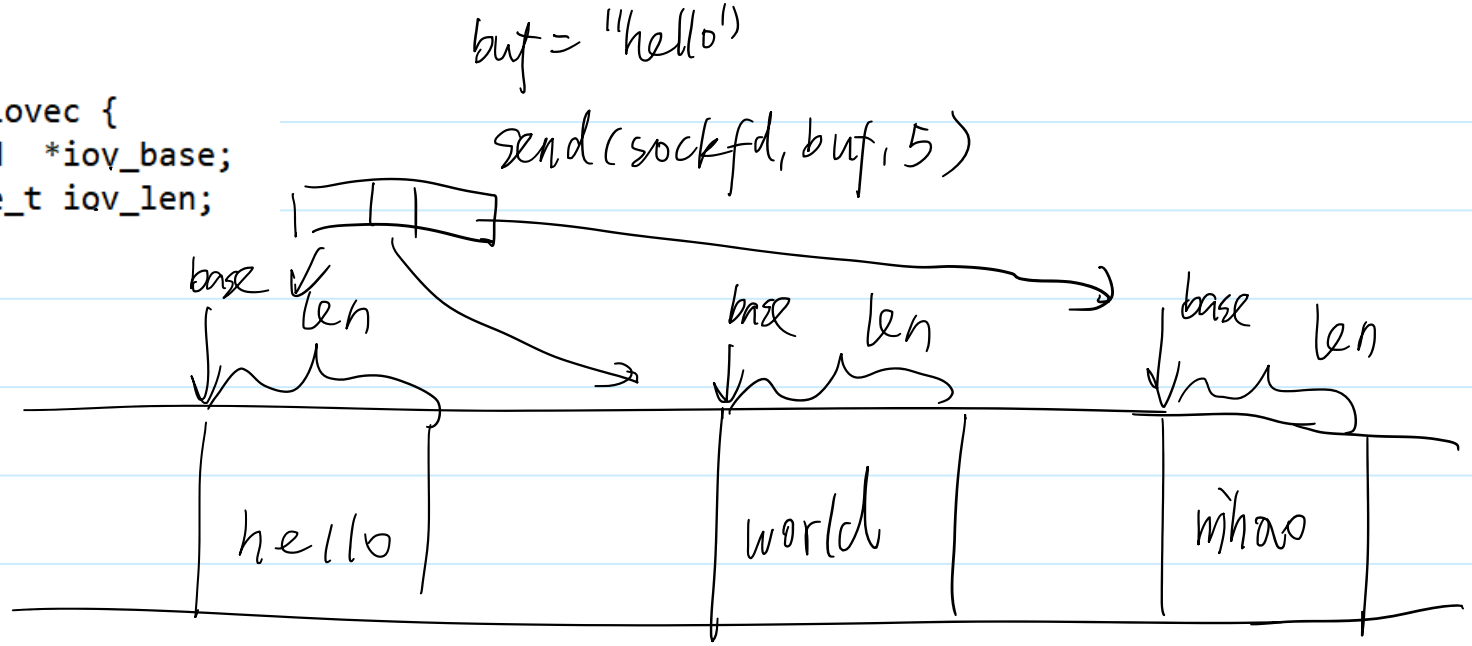
```
struct iovec {                    /* Scatter/gather array items */
    void  *iov_base;              /* Starting address */
    size_t iov_len;               /* Number of bytes to transfer */
};

struct msghdr {
    void         *msg_name;       /* Optional address */
    socklen_t     msg_namelen;    /* Size of address */
    struct iovec *msg_iov;        /* Scatter/gather array */
    size_t        msg_iovlen;     /* # elements in msg_iov */
    void         *msg_control;    /* Ancillary data, see below */
    size_t        msg_controllen; /* Ancillary data buffer len */
    int           msg_flags;      /* Flags on received message */
};
```



消息正文

2023年5月10日 11:28

```
struct iovec {
    void  *iov_base;
    size_t iov_len;
};
```



设置消息的正文部分

2023年5月10日 11:47

```
// 消息的正文部分
char buf[] = "hello";
struct iovec iov[1];
iov[0].iov_base = buf;
iov[0].iov_len = 5;
hdr.msg_iov = iov;
hdr.msg_iovlen = 1;
```

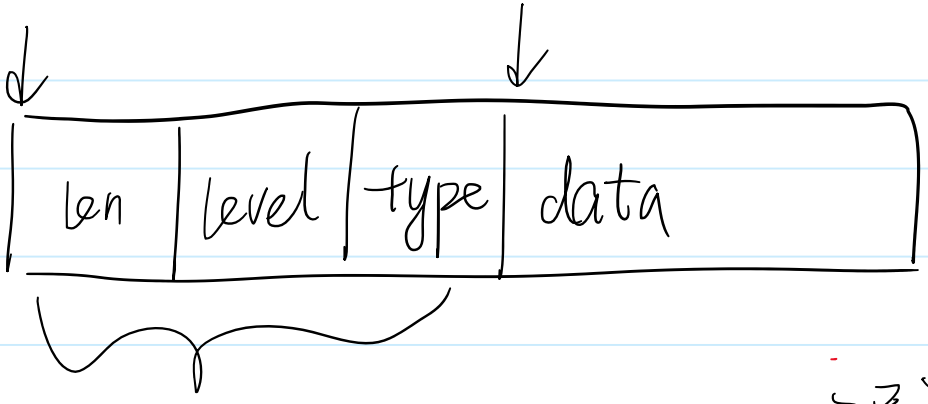
控制信息

2023年5月10日 11:47

```
struct cmsghdr {
    size_t cmsg_len;    /* Data byte count, including header
                           (type is socklen_t in POSIX) */
    int    cmsg_level;  /* Originating protocol */
    int    cmsg_type;   /* Protocol-specific type */
/* followed by
   unsigned char cmsg_data[]; */
};
```

结构体最后一个成员，是一个长度为 柔性0[1]的数组. ⟹ 变长数组

struct cmsghdr * p = malloc (足够的大小)



只能申请连续空间.

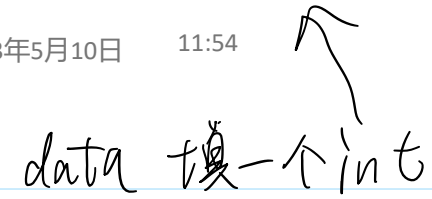
```
size_t CMSG_LEN(size_t length);
unsigned char *CMSG_DATA(struct cmsghdr *cmsg);
```

→ 已知data的length 算总长度.

→ 已知初始地址，找到 data 的地址

发送文件描述符

2023年5月10日 11:54




```c
#include <49func.h>
int sendfd(int sockfd ,int fdtosend){
    struct msghdr hdr;
    bzero(&hdr,sizeof(hdr));//不可省略
    // 消息的正文部分
    char buf[] = "hello";
    struct iovec iov[1];
    iov[0].iov_base = buf;
    iov[0].iov_len = 5;
    hdr.msg_iov = iov;
    hdr.msg_iovlen = 1;
    // 消息的控制字段
    // 堆空间存放变长结构体
    struct cmsghdr * pcmsghdr;
    pcmsghdr = (struct cmsghdr *)calloc(1,CMSG_LEN(sizeof(int)));
    pcmsghdr->cmsg_len = CMSG_LEN(sizeof(int));
    pcmsghdr->cmsg_level = SOL_SOCKET;
    pcmsghdr->cmsg_type = SCM_RIGHTS;//说明控制信息是文件描述符
    *(int *)CMSG_DATA(pcmsghdr) = fdtosend;// pcmsghdr找到data的首地址，把void *转成int *，再解引用赋值
    hdr.msg_control = pcmsghdr;
    hdr.msg_controllen = CMSG_LEN(sizeof(int));
    // sendmsg 发送正文
    int ret = sendmsg(sockfd,&hdr,0);
    ERROR_CHECK(ret,-1,"sendmsg");
}
```

接收文件描述符

2023年5月10日 12:09

```c
int recvfd(int sockfd,int *pfdtorecv){
    struct msghdr hdr;
    bzero(&hdr,sizeof(hdr));
    // 消息的正文部分
    char buf[6] = {0};
    struct iovec iov[1];
    iov[0].iov_base = buf;
    iov[0].iov_len = 5;
    hdr.msg_iov = iov;
    hdr.msg_iovlen = 1;
    // 消息的控制字段
    // 堆空间存放变长结构体
    struct cmsghdr * pcmsghdr;
    pcmsghdr = (struct cmsghdr *)calloc(1,CMSG_LEN(sizeof(int)));
    pcmsghdr->cmsg_len = CMSG_LEN(sizeof(int));
    pcmsghdr->cmsg_level = SOL_SOCKET;
    pcmsghdr->cmsg_type = SCM_RIGHTS;//说明控制信息是文件描述符
    hdr.msg_control = pcmsghdr;
    hdr.msg_controllen = CMSG_LEN(sizeof(int));
    // recvmsg 接收正文
    int ret = recvmsg(sockfd,&hdr,0);
    ERROR_CHECK(ret,-1,"recvmsg");
    printf("buf = %s\n", buf);
    *pfdtorecv = *(int *)CMSG_DATA(pcmsghdr);
    printf("fdtorecv = %d\n", *pfdtorecv);
}
```

单元测试的代码

2023年5月10日 12:10

```c
#include <49func.h>
int sendfd(int sockfd ,int fdtosend);
int recvfd(int sockfd,int *pfdtorecv);
int main(){
    int fds[2];
    //pipe(fds);
    socketpair(AF_LOCAL,SOCK_STREAM,0,fds);
    if(fork() == 0){
        int fd = open("file1",O_RDWR);
        write(fd,"hello",5);
        sendfd(fds[1],fd);
        printf("child fd = %d\n", fd);
    }
    else{
        int fd1 = open("file2",O_RDWR);
        int fd2;
        recvfd(fds[0],&fd2);
        printf("parent fd1 = %d, fd2 = %d\n",fd1,fd2);
        write(fd2,"world",5);
        wait(NULL);
    }
}
```