# 一、I/O复用概述

## I/O复用概念：

进程不阻塞于某个特定的 I/O 系统调用

## I/O复用常用函数：

select、poll

# 二、select()函数

## select函数介绍：

int select(int maxfd, fd\_set \*readset, fd\_set \*writeset, fd\_set \*exceptset, const struct timeval \*timeout);

功能：

如果读取描述符（如套接字）数组有一个可读或写入描述符有一个可写或异常描述符有一个发送异常，则函数返回，否则等到超时

参数：

|  |  |  |
| --- | --- | --- |
| 参数 | 名称 | 说明 |
| maxfd | 指定要检测的描述符的范围 | 所检测描述符最大值+1 |
| readset | 可读描述符集 | 监测该集合中的任意描述符是否有数据可读 |
| writeset | 可写描述符集 | 监测该集合中的任意描述符是否有数据可写 |
| exceptset | 异常描述符集 | 监测该集合中的任意描述符是否发生异常 |
| timeout | 超时时间 | 超过规定时间后唤醒 |

返回值：

0：超时

-1：出错

>0：准备好的文件描述符数量

头文件：

#include <sys/select.h>

#include <sys/time.h>

超时时间：

//该结构体表示等待超时的时间

struct timeval{

long tv\_sec;//秒

long tv\_usec;//微秒

};

//比如等待10.2秒

struct timeval timeout;

timeoout.tv\_sec = 10;

timeoout.tv\_usec = 200000;

//将select函数的timeout参数设置为NULL则永远等待

## 描述符集合的操作：

select可传入读，写，异常3种描述符集合

如下描述符集合操作函数

//初始化描述符集

void FD\_ZERO(fd\_set \*fdset);

//将一个描述符添加到描述符集

void FD\_SET(int fd, fd\_set \*fdset);

//将一个描述符从描述符集中删除

void FD\_CLR(int fd, fd\_set \*fdset);

//检测指定的描述符是否有事件发生

int FD\_ISSET(int fd, fd\_set \*fdset);

## select()函数整体使用框架：

示例：检测 0、4、5 描述符是否准备好读

while(1)

{

    fd\_set rset;//创建一个描述符集rset

    FD\_ZERO(&rset);//对描述符集rset清零

    FD\_SET(0, &rset);//将描述符0加入到描述符集rset中

    FD\_SET(4, &rset);//将描述符4加入到描述符集rset中

    FD\_SET(5, &rset);//将描述符5加入到描述符集rset中

    // 等他 rset 集合可读

    if(select(5+1, &rset, NULL, NULL, NULL) > 0)

    {

        // 是否是 0 可读

        if(FD\_ISSET(0, &rset))

        {

            //描述符0可读及相应的处理代码

        }

        // 是否是 4 可读

        if(FD\_ISSET(4, &rset))

        {

            //描述符4可读及相应的处理代码

        }

        // 是否是 5 可读

        if(FD\_ISSET(5, &rset))

        {

            //描述符5可读及相应的处理代码

        }

    }

}

# 三、select函数的应用对比

我们通过udp同时收发的例子来说明select的妙处。

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/select.h>

#include <sys/time.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int main()

{

    int udpfd = 0;

struct sockaddr\_in saddr;

    bzero(&saddr,sizeof(saddr));

    saddr.sin\_family = AF\_INET;

    saddr.sin\_port   = htons(8000);

    saddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    //创建套接字

    if( (udpfd = socket(AF\_INET,SOCK\_DGRAM, 0)) < 0)

    {

        perror("socket error");

        exit(-1);

    }

    //套接字端口绑字

    if(bind(udpfd, (struct sockaddr\*)&saddr, sizeof(saddr)) != 0)

    {

        perror("bind error");

        close(udpfd);

        exit(-1);

}

    printf("input: \"sayto 192.168.220.X\" to sendmsg to somebody\033[32m\n");

    while(1)

    {

        char buf[100]="";

        fd\_set rset;    //创建文件描述符的聚合变量

        FD\_ZERO(&rset); //文件描述符聚合变量清0

        FD\_SET(0, &rset);//将标准输入添加到文件描述符聚合变量中

        FD\_SET(udpfd, &rset);//将udpfd添加到文件描述符聚合变量中

        write(1,"UdpQQ:",6);

        if(select(udpfd + 1, &rset, NULL, NULL, NULL) > 0)

        {

            if(FD\_ISSET(0, &rset))//测试0是否可读

            {

                struct sockaddr\_in caddr;

                bzero(&caddr,sizeof(caddr));

                caddr.sin\_family  = AF\_INET;

                caddr.sin\_port    = htons(8000);

                fgets(buf, sizeof(buf), stdin);

                buf[strlen(buf) - 1] = '\0';

                if(strncmp(buf, "sayto", 5) == 0)

                {

                    char ipbuf[16] = "";

                    inet\_pton(AF\_INET, buf+6, &caddr.sin\_addr);//给addr套接字地址再赋值.

                    printf("\rsay to %s\n",inet\_ntop(AF\_INET,&caddr.sin\_addr,ipbuf,sizeof(ipbuf)));

                    continue;

                }

                else if(strcmp(buf, "exit")==0)

                {

                    close(udpfd);

                    exit(0);

                }

                sendto(udpfd, buf, strlen(buf),0,(struct sockaddr\*)&caddr, sizeof(caddr));

            }

            if(FD\_ISSET(udpfd, &rset))//测试udpfd是否可读

            {

                struct sockaddr\_in addr;

                char ipbuf[INET\_ADDRSTRLEN] = "";

                socklen\_t addrlen = sizeof(addr);

                bzero(&addr,sizeof(addr));

                recvfrom(udpfd, buf, 100, 0, (struct sockaddr\*)&addr, &addrlen);

                printf("\r\033[31m[%s]:\033[32m%s\n",inet\_ntop(AF\_INET,&addr.sin\_addr,ipbuf,sizeof(ipbuf)),buf);

            }

        }

    }

    return 0;

}