小白也能看懂的poll源码解析

原创 陈同学在搬砖 陈同学在搬砖 2020-04-17 15:44

用途

poll函数工作原理与select函数类似，
也是监管一系列的文件描述符，
阻塞的去轮询看这些文件描述符是否可读/可写/异常，
再去调用io函数读写

用法

看一下下面这段简单的代码
实现的功能就是把标准输入(即文件描述符为0)
纳入poll的监管
然后poll在5s内阻塞的轮询
看是否有读就绪事件
如果有的话就返回 然后对其进行处理
如果超时或者出错的也返回

```
#include<poll.h>
#include<signal.h>
#include<iostream>
  #include <unistd.h>
using namespace std;
int main(){
        /*第一步 poll开始监听之前要知道要监管哪些套接字*/
            struct pollfd fds[1];
            fds[0].fd=0;
            fds[0].events=POLL_IN;
            fds[0].revents=POLL_IN;

        /*第二步 poll开始工作 阻塞的轮询看监管的套接字是否就绪*/

          int ret=poll(fds,1,5000);

        /*第三部  poll完成工作  有套接字就绪或者时间超时返回*/
        if(ret<0){cout<<"error"<<endl;}
        else if(ret==0){cout<<"time out"<<endl;}
        else{
            if(fds[0].revents==POLL_IN)
                {
                    char message[10];
                    read(fds[0].fd,message,sizeof(message));
                    cout<<message<<endl;
                }

        }


}
```

接口

上面的使用设计到了下面几个接口

pollfd

和select监听采用fd_set位数组不同
poll监听采用的是pollfd事件结构体数组
也就是先定义一个事件结构体数组
然后在事件结构体数组中
设定好要监听事件的一个文件描述符
以及要监听的事件等等信息

事件结构体的原型如下

pollfd结构体的原型为：

```
struct pollfd {
   int   fd;         /* 文件描述符 */
   short events;     /* 注册的事件 */
   short revents;    /* 实际发生的事件，由内核填充 */
};
```

其中：
1.fd 表示你要监听的文件描述符

2.events参数表示你要监听的事件类型
比如POLL_IN表示你想监听该套接字的读就绪事件

3.reevents表示当该套接字的某一事件就绪的
内核就会将该参数置为该事件类型
比如POLL_IN表示该套接字读就绪了

总结来说每个结构体的 events 域是由用户来设置，
告诉内核我们关注的是什么，
而 revents 域是返回时内核设置的，
以说明对该描述符发生了什么事件
events和reevents具体可以取的一些宏如下
 多个宏是可以通过或运算符同时监听的

poll

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


参数情况：
（1）fds：指向一个结构体数组的第0个元素的指针，
每个数组元素都是一个struct pollfd结构，
用于指定测试某个给定的fd的条件
（2）nfds：表示fds结构体数组大小
（3）timeout：表示poll函数的超时时间，单位是毫秒

函数返回值：
（1）返回值小于0，表示出错
（2）返回值等于0，表示poll函数等待超时
（3）返回值大于0，表示poll由于监听的文件描述符就绪返回，
    并且返回结果就是就绪的文件描述符的个数。
        poll函数使用前面提到的pollfd结构体中的revents参数，
      revents变量在每一次poll函数调用完成后
   内核设置会设置revents的值，
                  这个值其实也就是上面列出来的那些events的宏，
   以说明对该描述符发生了什么事件
   比如 调用完poll函数后要
     查看某一个文件描述符是否处于激活状态(比如可读)
   是通过调用pollfd参数的revents参数与POLLIN做比较如果相等，
   则说明该文件描述符处现在是可读的
   使用if语句:if(poll_fd.revents==POLLIN)

就绪情况：

读就绪

1）socket内核中,接收缓冲区中的字节数，
大于等于低水位标记SO_RCVLOWAT.
此时可以无阻塞的读该文件描述符，
并且返回值大于0；

2）socket TCP通信中，对端关闭连接，
此时对该socket读，则返回0；

3）监听的socket上有新的连接请求；

4）socket上有未处理的错误；


写就绪
1）socket内核中，
发送缓冲区中的可用字节数
(发送缓冲区的空闲位置大小)，
 大于等于低水位标记 SO_SNDLOWAT，
 此时可以无阻塞的写，并且返回值大于0；

2）socket的写操作被关闭(close或者shutdown).
对一个写操作被关闭的socket进行写操作，
会触发 SIGPIPE信号；

3）socket使用非阻塞connect连接成功或失败之后；

4）socket上有未读取的错误；
异常就绪：
socket上收到带外数据.
```

源码

源码结构

特性

1、优点
由上面的sys_poll可以看出， poll底层使用poll_list来管理 ，
它没有最大连接数的限制，原因是它是基于链表来存储的。

2、缺点
(1) 由上面的do_poll可以看出，
poll采用轮询的方式扫描文件描述符，
文件描述符数量越多，性能越差；

(2)由上面的do_sys_poll可以看出,在轮询期间，
需要复制大量的句柄数据结构到内核空间，
产生巨大的开销；

(3)由上面的do_poll可以看出，
返回的是含有整个句柄的数组，
应用程序需要遍历整个数组才能
发现哪些句柄发生了事件；

(4)触发方式是水平触发，
应用程序如果没有完成
对一个已经就绪的文件描述符进行IO操作，
那么之后每次select调用
还是会将这些文件描述符通知进程。

往期精彩回顾

你的微信消息是怎么发出去的?

1个小时学会所有Linux核心命令
一个小时学会Git
Leetcode面试高频题汇总--链表
Leetcode面试高频题汇总--数组
【设计模式】可能是东半球最透彻的单例模式讲解

听说点击在看的今年都会暴富脱单，升职加薪