linux新增的系统调用epoll

完成与select，poll等一样的功能，对多个描述符进行监听，但是效率很高

epoll共有三个api

**int epoll\_create(int size);**

创建一个epoll句柄，size告诉内核监听的描述符多大

返回一个句柄

**int epoll\_ctl(int epfd,int op,int fd,struct epoll\_event \*event);**

epoll的事件注册函数，注册要监听的事件类型

**epfd**:epoll\_create返回的句柄

op:动作

EPOLL\_CTL\_ADD：注册新的fd到epfd中；  
EPOLL\_CTL\_MOD：修改已经注册的fd的监听事件；  
EPOLL\_CTL\_DEL：从epfd中删除一个fd；

fd:需要监听的fd

event:告诉内核监听哪些事情

struct **epoll\_event**{

\_\_uint32\_t events;

/\*events可是如下宏的集合

EPOLLIN 表示对应的文件描述符可以读

EPOLLOUT 表示对应的文件描述符可以写

EPOLLPRI 表示对应的文件描述符有紧急数据可读

EPOLLERR 表示对应的文件描述符发生错误

EPOLLHUP 表示对应的文件描述符被挂断

\*/

epoll\_data\_t data;

}

typedef union epoll\_data {

void \*ptr;

int fd;

\_\_uint32\_t u32;

\_\_uint64\_t u64;

}**epoll\_data\_t**

**int epoll\_wait(int epfd,struct epoll\_event \*event,int maxevents,int timeout);**

等待事件的产生

event：从内核获取事件的集合

maxevents：告诉内核这个events多大，不大于epoll\_create()创建时的size

timeout:超时时间(ms),0表示立即返回,-1表示阻塞

返回值是需要处理的事件数目

用法例子

<http://blog.csdn.net/yusiguyuan/article/details/15026941>

int main(){

efd = eventfd(0,0);

int pthread\_t pid = pthread\_create(&pid,NULL,read\_thread,NULL);

for(i=0;i<5;i++){

count = 4;

write(efd,&count,sizeof(count));

printf("success write to efd, write %d bytes(%llu) at %lds %ldus\n",ret,count,tv.tv\_sec, tv.tv\_usec);

sleep(1);

}

}

void \*read\_thread(void \*dummy){

struct epoll\_event events[10];

int ep\_fd = **epoll\_create**(1024);

**struct epoll\_event read\_event;//加入监控描述符及事件**

**read\_event.events = EPOLLHUP | EPOLLERR | EPOLLIN;**

**read\_event.data.fd = efd;**

**epoll\_ctl**(ep\_fd,EPOLL\_CTL\_ADD,efd,&read\_event);

while(1){

ret = **epoll\_wait**(ep\_fd, &events[0], 10, 5000);

if(ret > 0){

for(;i<ret;i++){

if(events[i].events & EPOLLHUP){printf("epoll eventfd has epoll hup.\n");}

else if(events[i].events & EPOLLERR){printf("epoll eventfd has epoll error.\n");}

else if(events[i].events & EPOLLIN){

int event\_fd = events[i].data.fd;

ret = read(event\_fd, & count, sizeof(count));

printf("success read from efd, read %d bytes(%llu) at %lds %ldus\n",ret, count, tv.tv\_sec, tv.tv\_usec);

}

}

}else if(ret == 0){printf("epoll wait timed out.\n");break;}

}

1. success write to efd, write 8 bytes(4) at 1328805612s 21939us
2. success read from efd, read 8 bytes(4) at 1328805612s 21997us
3. success write to efd, write 8 bytes(4) at 1328805613s 22247us
4. success read from efd, read 8 bytes(4) at 1328805613s 22287us
5. success write to efd, write 8 bytes(4) at 1328805614s 22462us
6. success read from efd, read 8 bytes(4) at 1328805614s 22503us
7. success write to efd, write 8 bytes(4) at 1328805615s 22688us
8. success read from efd, read 8 bytes(4) at 1328805615s 22726us
9. success write to efd, write 8 bytes(4) at 1328805616s 22973us
10. success read from efd, read 8 bytes(4) at 1328805616s 23007us
11. epoll wait timed out.