特立独行MVP

【第三章：BAT等名企面试真题解析8讲】第14节：腾讯面试真题解析之epoll 已购

来自【《收割BAT：C++校招学习路线总结》】 | 162 浏览 | 1 回复 | 2020-02-22

特立独行MVP 专栏作者 +关注

前言

如果说虚函数是C++面试常考问题，那么epoll则是C++后台开发方向面试必问问题。以我的面试经验总结来说，epoll相关是出现频次最多的面试题。因为如果涉及高性能服务器开发方面，epoll是非常常用的系统调用，在很多的开源项目当中epoll都是核心技术，例如Nginx和Redis等等。本节我将结合腾讯面试真题讲解eopll的相关知识点。

IO多路复用

如果要理解epoll，首先需要知道Linux下的IO多路复用技术。I/O多路复用的本质是使用select,poll或者epoll函数，挂起进程，当一个或者多个I/O事件发生之后，将控制返回给用户进程。以服务器编程为例，传统的多进程(多线程)并发模型，在处理用户连接时都是开启一个新的线程或者进程去处理一个新的连接，而I/O多路复用则可以在一个进程(线程)当中同时监听多个网络I/O事件，也就是多个文件描述符。select、poll 和 epoll 都是 Linux API 提供的 IO 复用方式。

IO多路复用优点：

1.相比基于进程的模型给程序员更多的程序行为控制。

2.IO多路复用只需要一个进程就可以处理多个事件，单个进程内数据共享变得容易，调试也更容易。

3.因为在单一的进程上下文当中，所以不会有多进程多线程模型的切换开销。

IO多路复用缺点：

1.业务逻辑处理困难，编程思维不符合人类正常思维。

2.不能充分利用多核处理器。

select

特立独行MVP

历一遍文件描述符就知道哪些文件描述符上有读写事件发生，进行相应的处理。函数原型如下：

```
#include <sys/select.h>

int select(int nfds,fd_set *readset,fd_set *writeset,fd_set *exceptset, struct timeval *time
```

nfds参数指定被监听的文件描述符的总数；readset,writeset,exceptset参数分别是可读，可写和异常事件对应的文件描述符集合；而fd_set结构体由一个整形数组组成，该数组每一位标记了一个文件描述符，该数组的上限有宏FD_SETSIZE指定；timeout是select的超时时间。

select机制的缺点

1. 每次调用select，都需要把监听的文件描述符集合fd_set从用户态拷贝到内核态，从算法角度来说就是O(n)的时间开销。
2. 每次调用select调用返回之后都需要遍历所有文件描述符，判断哪些文件描述符有读写事件发生，这也是O(n)的时间开销。
3. 内核对被监控的文件描述符集合大小做了限制，并且这个是通过宏控制的，大小不可改变(限制为1024)。

poll

1997年出现了poll系统调用，和select类似，poll也是在指定时间内监听多个文件描述符。

函数原型：

```
#include <poll.h>
int poll(struct pollfd *fds, nfds_t nfds, int timeout);
```

优化：poll改变了文件描述符集合的描述方式，通过一个pollfd数组向内核传递需要关注的事件，没有描述符个数的限制.

```
typedef struct pollfd {
        int fd;
        short events;
        short revents;
} pollfd_t;
```

epoll

Linux2.6内核实现了epoll，函数原型如下：

```
#include <sys/epoll.h>

int epoll_wait(int epfd, struct epoll_event *events, int maxevents, int timeout);
```

epoll是Linux特有的I/O复用函数。对于select的几个问题，epoll的优化策略如下：

2.在epoll_wait函数返回时，无须遍历整个被侦听的描述符集，只要遍历那些被内核IO事件异步唤醒而加入就绪队列的描述符集合就行，这样就避免了无效的遍历。

简单来说，在用户进程通过epoll_ctl系统调用向内核注册需要监听的事件，使用epoll_wait开启I/O多路复用，epoll为每个文件描述符指定一个回调函数，当设备就绪，就会调用这个回调函数，而这个回调函数会把就绪的文件描述符加入一个就绪链表，唤醒等待在这个epoll_wait上的进程，用户进程只需要遍历这个就绪链表就可以进行下一步处理。同时内核通过红黑树的结构存储这些文件描述符，提升查找的效率。

腾讯面试真题

1.说一下epoll的好处

答：epoll解决了select和poll在文件描述符集合拷贝和遍历上的问题，能够在一个进程当中监听多个文件描述符，并且十分高效。

2.epoll和select之间的区别？

答：1.epoll不需要每次调用的时候都在用户态和内核态拷贝文件描述符集合，而select需要。

2.epoll在内核态通过红黑树和回调的方式处理文件描述符，返回时只需要遍历就绪链表即可，而select通过数 组的方式处理，每次需要遍历整个数组判断哪些文件描述符就绪。

3.epoll使用maxevents参数指定最多监听的文件描述符的个数，最多可以到达系统允许打开的最大文件描述 符个数，而select允许监听的最大文件描述符熟练有限制。

4.epoll可以工作在高效的ET模式，而select只能工作在相对低效的LT模式。

5.epoll使用的是回调的方式将就绪文件描述符插入就绪链表，返回时用户只需要O(1)的时间开销寻找就绪事 件，而select采用轮询的方式，需要O(n)的时间开销。

3.epoll需要在用户态和内核态拷贝数据么？

答：在注册监听事件时从用户态将数据传入内核态；当返回时需要将就绪队列的内容拷贝到用户空间。

4.epoll的实现知道么？在内核当中是什么样的数据结构进行存储，每个操作的时间复杂度是多少？

答：在内核当中是以红黑树的方式组织监听的事件，查询开销是O(logn)。采用回调的方式检测就绪事件，时间复杂的位O(1);

特立独行MVP

5.epoll的水平触发和边缘触发有什么区别

答：**水平触发(Level Trigger)：是eopll的**默认工作模式，当epoll_wait检测到某文件描述符上有事件发生，并通知应用程序之后，应用程序可以不立即处理该事件，当下次调用epoll_wait时，epoll_wait还会再次向应用程序通知此事件，直到该事件被应用程序处理。

边缘触发（Edge Trigger）： 当epoll_wait检测到某文件描述符上有事件发生，并通知应用程序之后，应用程序必须立即处理该事件。如果应用程序不处理，下次调用epoll_wait时，epoll_wait不会再次通知此事件。

ET模式很大程度上减少了epoll事件被重复触发的次数，因此效率比LT模式下高。

6.什么是EPOLLONESHOT事件？

答：注册了EPOLLONESHOT事件的文件描述符，操作系统只能触发该文件描述符上注册的最多一个就绪事件，并且只触发一次。该处理方式保证了当一个线程正在处理该文件描述符上的就绪事件时，其他线程无法操作该文件描述符。

总结

epoll的出现是由于历史的原因，在十几年前前并发量不大的情况下select和poll已经可以解决那时的问题，而随着并发量越来越大，select和poll就暴露出设计的问题，从而一步一步优化，这也是符合程序设计的思路，先解决当前问题，再考虑优化。三者对比总结如下：

系统调用	select	poll	epoll
查找就绪文件描述符的操作时间复杂的	O(n)	O(n)	O(1)
最大支持的文件描述符数量	有最大限制一般为1024	最多可达系统文件描述符上限	最多可达系统文件描述符上限
工作模式	LT	LT	LT,ET
效率对比	调用时需要O(n)的文件描述符集合拷贝	调用时需要O(n)的文件描述符集合拷贝	只在添加描述符时拷贝，调用无需额外拷贝开销

本节主要介绍了epoll和select,poll的区别，并结合腾讯的面试真题对epoll相关问题进行解答。

特立独行MVP

感谢阅读，如果文中有任何错误欢迎你给我留言指出，也欢迎分享给更多的朋友一起阅读。

举报

收藏 赞 1

相关专栏

《收割BAT：C++校招学习路线总结》

19篇文章 | 95订阅

已订阅

1条评论 默认排序

dragonlogin

昨天 18:11:35

1 0 1#

真题的第4个，复杂度写的有问题

请留下你的观点吧~

发布

专栏推荐

《收割BAT：C++校招学习路线总结》

《收割BAT：C++校招学习路线总结》，专栏共计17节。专栏分为五大主要内容，包括后台开发学习...

19篇文章 | 95阅读