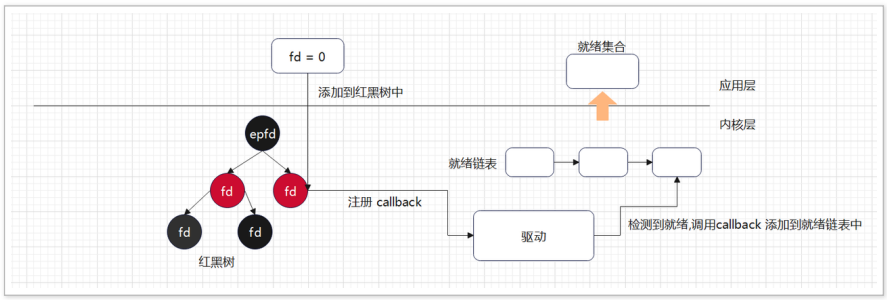
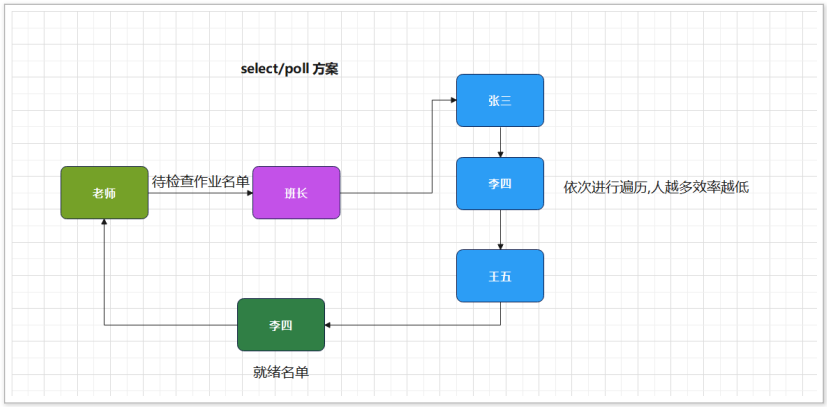
12.6 多路复用 io-epoll(一) 基本原理与应用_物联网 / 嵌入式工程师 - 慕课网

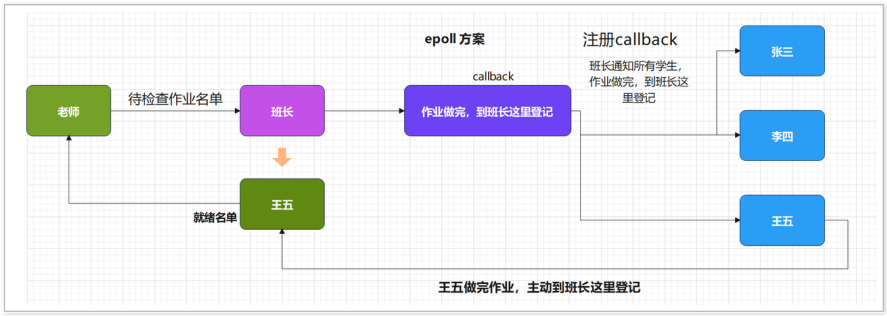
> 慕课网慕课教程 12.6 多路复用 io-epoll(一) 基本原理与应用涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- epoll 相对于 select 与 poll 有较大的不同，主要是针对前面两种多路复用 IO 接口的不足
  - 与 select/poll 方案对比
    - select 方案使用数组存储文件描述符，最大支持 1024
    - select 每次调用都需要将描述符集合拷贝到内核中，非常消耗资源
    - poll 方案解决文件描述符存储数量限制问题，但其他问题没有得到解决
    -
    - select / poll 底层使用轮询的方式检测文件描述符是否就绪，文件描述符越多，则效率越低
    -
    - epoll 底层使用红黑树，没有文件描述符数量的限制，并且可以动态增加与删除节点，不用重复拷贝
    - epoll 底层使用 callback 机制，没有采用遍历所有描述符的方式，效率较高
    -



- 下面以老师检查学生作业为例，来看两种方案
  - select/poll 方案
    -





- epoll 创建需要调用 epoll_create 函数, 用于创建 epoll 实例

函数头文件 #include <sys/epoll.h>

函数原型 int epoll_create(int size);

函数功能 创建一个 epoll 实例, 分配相关的数据结构空间

函数参数 size : 需要填一个大于 0 的数，从 Linux 2.6.8 开始，size 参数被忽略

函数返回值

- 成功 : 返回 epoll 文件描述符
- 失败 : 返回 –1, 并设置 errno

示例

创建一个 epoll 实例

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


int main(void)
{
    int epfd;

    epfd = epoll_create(1);
    if (epfd == -1){
        perror("[ERROR] epoll_create(): ");
    }

    printf("epfd = %d\n",epfd);


    return 0;
}
```

1. 理解 epoll 的基本原理, 并创建 epoll 实例

全文完