知识储备

理论知识

### 计算机体系

#### cpu相关

##### 系统调用

　syscall是执行一个系统调用，根据指定的参数number和所有系统调用的接口来确定调用哪个系统调用，用于用户空间跟内核之间的数据交换

cpus = sysconf(\_SC\_NPROCESSORS\_CONF);

##### 含参数的宏与函数

宏替换不占运行时间，只占编译时间；而函数调用则占运行时间（分配单元、保留现场、值传递、返回），所以每次执行都要载入所以执行起来比较慢一些。。

? 定义宏的时候不要在宏及其参数之间键入空格，因为宏替换的时候会把你不经意打的空格当作宏的一部分进去。

? 在宏定义中把每个参数都用括号括起来，同时把整个结果也用括号（对于单个表达式的宏，可以使用小括号(),对于宏定义的复合语句可以使用{}，Linux内核中有一个比较好的宏定义，do {…}while(0)）括起来，以防止当宏用于一个更大的表达式时可能出现的问题。

? 使用宏次数多时，宏展开后源程序长，因为每展开一次都使程序增长，但是执行起来比较快一点（这也不是绝对的，当有很多宏展开，目标文件很大，执行的时候运行时系统换页频繁，效率就会低下）。而函数调用不使源程序变长。

? 函数调用时，先求出实参表达式的值，然后带入形参。而使用带参的宏只是进行简单的字符替换。

? 函数调用是在程序运行时处理的，分配临时的内存单元；而宏展开则是在编译时进行的，在展开时并不分配内存单元，不进行值的传递处理，也没有“返回值”的概念。

? 对函数中的实参和形参都要定义类型，二者的类型要求一致，如不一致，应进行类型转换；而宏不存在类型问题，宏名无类型，它的参数也无类型，只是一个符号代表，展开时带入指定的字符即可。宏定义时，字符串可以是任何类型的数据。

? 调用函数只可得到一个返回值，且有返回类型，而宏没有返回值和返回类型，但是用宏可以设法得到几个结果。

? 函数体内有Bug，可以在函数体内打断点调试。如果宏体内有Bug，那么在执行的时候是不能对宏调试的，即不能深入到宏内部。

? C++中宏不能访问对象的私有成员，但是成员函数就可以。

? 宏的定义很容易产生二义性，如：定义#define S(a) (a)\*(a)，代码S(a++)，宏展开变成(a++)\*(a++)这个大家都知道，在不同编译环境下会有不同结果。

\*\*\*\*内联函数和宏的区别（内联函数的优点）\*\*\*\*\*\*

**\*内联函数和宏的区别在于，宏是由预处理器对宏进行替代，而内联函数是通过编译器控制来实现的。\***而且内联函数是真正的函数，只是在需要用到的时候，内联函数像宏一样的展开，**所以取消了函数的参数压栈，减少了调用的开销。**你可以象调用函数一样来调用内联函数，而不必担心会产生于处理宏的一些问题。

我们可以用Inline来定义内联函数，不过，任何在类的说明部分定义的函数都会被自动的认为是内联函数。（当然内联函数的识别对编译器来说是一个复杂的工作，后继可能有专门的论述）

当然，**内联函数也有一定的局限性。就是函数中的执行代码不能太多了**，如果，内联函数的函数体过大，一般的编译器会放弃内联方式，而采用普通的方式调用函数。这样，内联函数就和普通函数执行效率一样了。

内联函数通过避免被调用的开销来提高执行效率，尤其是它能够通过调用（“过程化集成”）被编译器优化。

**\*如何选择使用宏还是函数：以下情况可以选择宏，其他情况最好选用函数\***

2 一般来说，用宏来代表简短的表达式比较合适。

2 在考虑效率的时候，可以考虑使用宏，或者内联函数。

2 在头文件保护（防止重复包含编译），条件编译中的#ifdef，#if defined以及assert的实现。

##### 线程和cpu

###### 发生线程切换

​ x86是不会的，不管是虚拟寻址的tlb还是物理寻址的cache都不会。

因为线程共享进程的线性地址

空间，所以不需要清tlb。进程切换会清tlb，因为会涉及到地址空间的切换。

据我所知mips也不会，arm就不清楚了

##### POSIX Thread Library (NPTL)

##### 协程概念

#### 解决问题思路

##### DPDK-VPP 一次mbuf地址被踩的定位思路

https://blog.csdn.net/sjin\_1314/article/details/106090582

#### 算法与数据结构

##### 局部敏感性hash

### 协议和接口

##### RESTful API

一.概念和区别：

1.关于rest

rest  :representational state transfer      表述性状态转移,是一种架构风格。

rest原则：

   <1>网络上的所有事物都被抽象为资源

   <2> 每个资源都有一个唯一的资源标识符

   <3> 同一个资源具有多种表现形式(xml,json等)

   <4>  对资源的各种操作不会改变资源标识符

   <5> 所有的操作都是无状态的

2.关于restful

restful: 遵守了rest 原则 的web服务

理解：rest与restful相比，多了一个ful,就英语层面来说是一个形容词，restful翻译为中文为： “rest式的”

是rest式的什么呢？答案是 rest式的应用，rest风格的web服务也是rest式的应用，rest式的web服务是一种ROA(The Resource-Oriented Architecture)(面向资源的架构)，ROA听起来很高大上有没有。。

3.两者的联系与区别

restful是由rest派生出来的。

二.restful用法：

在Restful之前的操作：

http://127.0.0.1/user/query/1 GET  根据用户id查询用户数据

http://127.0.0.1/user/save POST 新增用户

http://127.0.0.1/user/update POST 修改用户信息

http://127.0.0.1/user/delete GET/POST 删除用户信息

RESTful用法：

http://127.0.0.1/user/1 GET  根据用户id查询用户数据

http://127.0.0.1/user  POST 新增用户

http://127.0.0.1/user  PUT 修改用户信息

http://127.0.0.1/user  DELETE 删除用户信息

之前的操作是没有问题的,大神认为是有问题的,有什么问题呢?你每次请求的接口或者地址,都在做描述,例如查询的时候用了query,新增的时候用了save,其实完全没有这个必要,我使用了get请求,就是查询.使用post请求,就是新增的请求,我的意图很明显,完全没有必要做描述,这就是为什么有了restful.

三.优点：

精简，一看就明白。