

部分代码规范

TIPS:部分代码规范

命名约定

函数命名, 变量命名, 文件命名要有描述性: **少用缩写**.

尽可能使用描述性的命名, 别心疼空间, 毕竟相比之下让代码易于新读者理解更重要. 不要用只有项目开发能理解的缩写, 也不要通过砍掉几个字母来缩写单词.

好的命名习惯:

```
int price_count_reader; // 无缩写
int num_errors; // "num" 是一个常见的写法
```

坏的命名习惯:

```
int nerr; // 含糊不清的缩写。
int n_comp_conns; // 含糊不清的缩写。
int wgc_connections; // 只有你才知道是什么意思。
int pc_reader; // "pc" 有太多可能的解释了。
int cstmr_id; // 删减了若干字母。
```

注意, 一些特定的广为人知的缩写是允许的, 例如用 `i`、`j` 表示迭代变量;

文件名要全部小写, 可以包含下划线 (`_`) 或连字符 (`-`), 依照项目的约定. 如果没有约定, 那么 `""` 更好.

例如:

```
my_useful_class.cpp
my-useful-class.cpp
myusefulclass.cpp
```

所有类型命名 —— 类, 结构体, 类型定义 (**`typedef`**), 枚举, 类型模板参数 —— 均使用相同约定, 即以大写字母开始, 每个单词首字母均大写, 不包含下划线. 例如:

```
// 类和结构体
class UrlTable { ...
class UrlTableTester { ...
struct UrlTableProperties { ...
// 类型定义
typedef hash_map<UrlTableProperties *, string> PropertiesMap;
// using 别名
using PropertiesMap = hash_map<UrlTableProperties *, string>;
```

```
// 枚举
enum UrlTableErrors { ...
```

变量 (包括函数参数) 和数据成员名一律小写, 单词之间用下划线连接. 类的成员变量以下划线结尾, 但结构体的就不用

普通变量命名

```
string table_name; // 好 - 用下划线.
string tablename; // 好 - 全小写.
string tableName; // 差 - 混合大小写
```

类数据成员

不管是静态的还是非静态的, 类数据成员都可以和普通变量一样, 但要接下划线.

```
class TableInfo {
...
private:
string table_name_; // 好 - 后加下划线.
string tablename_; // 好.
static Pool<TableInfo>* pool_; // 好.
};
```

结构体变量

不管是静态的还是非静态的, 结构体数据成员都可以和普通变量一样, 不用像类那样接下划线:

```
struct UrlTableProperties {
string name;
int num_entries;
static Pool<UrlTableProperties>* pool;
};
```

一般来说, 非类的成员函数名的每个单词首字母大写 (即“大驼峰命名”), 没有下划线. 对于首字母缩写的单词, 更倾向于将它们视作一个单词进行首字母大写 (例如, 写作 **StartRpc()** 而非 **StartRPC()**).

```
AddTableEntry()
DeleteUrl()
OpenFileOrDie()
```

类的成员函数名采用小驼峰命名法 (第一个单词首字母小写, 之后的单词首字母大写)

```
class TableInfo {
...
public:
void addTableEntry(); // 好
void deleteUrl()//好
```

```
void openFileOrDie(); //好  
};
```