

目 录

服务操作(术)

CSAPP-learnig-01

window常用操作

临时记录

代码规范

服务操作(术)

mindoc安装问题

安装后无法进入容器：请使用：

```
1. docker exec -it 328e6576e555 /bin/sh
```

无法修改数据库，则直接修改start.sh文件，示例：



也可以直接对conf/app.conf进行修改。

- 使用官网的镜像直接安装，安装成功后进入容器修改就行,这种是最方便的：

```
1. docker run --name=mindoc --restart=always -v /data/uploads:/mindoc/uploads -v /data/database:/data/database -e DB_ADAPTER=sqlite3 -e MYSQL_INSTANCE_NAME=./database/mindoc.db -e CACHE=true -e CACHE_PROVIDER=file -e ENABLE_EXPORT=true -e BASEURL=https://www.iminho.me/wiki -p 8181:8181 -d registry.cn-hangzhou.aliyuncs.com/mindoc/mindoc:v0.12
```

CSAPP-learnig-01

lldb

linux 调试工具

- 数字的临结时非常重要的，尤其是在重要的系统中
- using return to a sort of a modern technique called return to return oriented programming

移位操作：Shift Operations

- 左移:后面填充**0**:

```
0000 0001 << 2 0000 0100
```

- 右移: 逻辑右移: 最左端填充**0**, 算数右移: 最右端填充最高有效位

```
0000 1000 >> 3 0000 0001
1000 1000 >> 3 1111 0001
```

为什么要这么做?

原因:

无符号和有符号数字范围

示例以5位数字来表示:

unsign num:

4 3 2 1 0

16 8 4 2 1

max: 1 1 1 1 1 -> 31 = $2^w(\text{次方}) - 1$

min: 0 0 0 0 0 -> 0

Two's Complement Values:

4 3 2 1 0

16 8 4 2 1

max 0 1 1 1 1 -> 15 = $2^{(w-1)}(\text{次方}) - 1$

min 1 0 0 0 0 -> -16 = $-2^{(w-1)}(\text{次方})$

$TMax: 2^{w-1} - 1$

$TMin: -2^{w-1}$

$UMax: 2^w - 1$

无符号数和有符号数运算时转换规则：

1. 操作的变量全为有符号，则认为时有符号数运算
2. 操作的变量只要有一个无符号数，则将其余有符号先转换为无符号数，再进行运算

window常用操作

- 通过修改注册表修改环境变量
- **reg**所有命令集合
- 更新**git**子项目：
 - 第一种方法：
 - 第二种方法：
 - 删除远程分支：
 - 删除本地分支：
 - 切换分支：
- **QLatin1String**和**QStringLiteral**
- 为什么不能用**const**修饰类的静态成员函数：
- 写代码时在最开始就去做的事：
- **qbs**在写依赖第三方库的用法：
- **ssh**连接服务器命令：
- 打印端口的连接数量：
- 更改**window ssh**连接时默认**shell**
- **css**样式表参考网站
- **jinJa**拼接url
- **Qt**密码不使用小圆点
- **screen** 基本使用

通过修改注册表修改环境变量

```

1. @echo off
2.
3. set mysql_path=D:\mysql\bin
4.
5. set PATH=%PATH%;%mysql_path%;
6.
7. set RegV=HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment
8.
9. reg add "%RegV%" /v "Path" /t REG_EXPAND_SZ /d "%PATH%" /f

```

reg所有命令集合

<https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/reg-add>

更新**git**子项目：

第一种方法：

```

1. git submodule update --init --recursive
2. git status

```

```
3. git pull
```

第二种方法:

```
1. git submodule init
2. git submodule update
3. git pull origin master
```

删除远程分支:

```
1. git branch -a
2. git push origin --delete xxx
```

删除本地分支:

```
1. git branch -d xxx
```

切换分支:

```
1. git checkout xxx
```

QLatin1String和QStringLiteral

- a. 如果操作支持对const char那么QLatin1String的使用比QString的效率, 它和const char的效率接近, 仅仅时对其很浅的封装;
- b. 如果操作只支持QString,那么无论时传递const char* 和QLatin1String, 它都将隐士转换成QString,因此此时应使用QStringLiteral宏, 它是个lambda表达式, 会将只读QString字符串保存至程序的.rodata。由于QString是隐士共享的因此没有拷贝等开销。

扩展: .bss、.data、.rodata

首先全局变量放在全局内存中、static修饰的局部变量也放在全局内存中, 但是其作用域是局部的。在ELF格式的可执行文件中, 全局内存有三种: .bss、.data、.rodata

- a. bss指没有初始化和初始化为0的全局变量
- b. data指初始化过(非零)的非const的全局变量
- c. rodata(read only data)指只读数据(const)

rodata简单说明:

常量不一定都放在rodata, 有的直接编码在指令里, 存在代码段(.text)中, 对于字符串常量, 编译器自动去掉重复字符串, 保证只有一份拷贝在多进程间共享, 能够提高空间利用率, 有的嵌入式系统, 其直接放在ROM里, 运行时直接读取ROM, 无需加载到RAM

为什么不能用const修饰类的静态成员函数:

const限定符作用于非静态成员函数时, 会影响this指针, 使得this指针为const*, 而静态成员函数没有

this指针，因此没有意义

写代码时在最开始就去做的事：

- 创建命名空间，命名空间和代码添加空格
- 类的成员变量添加_
- 类成员函数返回添加const、函数形参尽可能传递const引用
- 抵制烂名字
- 抵制重复
- 范围for循环无脑使用const auto&

qbs在写依赖第三方库的用法：

```
1. cpp.includePaths: base.concat("xx/include/xx")
2. cpp.libraryPaths: base.concat("xx/lib/" + qbs.buildVariant)
3. cpp.staticLibraries: ["xxx", "xxx"]
4. cpp.dynamicLibraries: []
```

ssh连接服务器命令：

```
1. ssh root@allianceshan.top
```

打印端口的连接数量：

```
1. windows: netstat -ant | find /C "192`"
2. linux: netstat -nat | grep -i "80" | wc -l
```

更改window ssh连接时默认shell

```
1. New-ItemProperty -Path "HKLM:\SOFTWARE\OpenSSH" -Name DefaultShell -Value "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -PropertyType String -Force
```

css样式表参考网站

[CSS样式表](#)

jinJa拼接url

```
1. {%for pdbFile in g.vizPDBFiles %}
2. {%set url=url_for('package.downloader', filename=pdbFile) %}
3. <li><a href="{{url}}">{{pdbFile}}</a></li>
4. {%endfor%}
```

Qt密码不使用小圆点

```
1. lineedit-password-character:42;
```

==Qt开源项目地址: ==

<https://www.zhihu.com/question/25649197>

<https://inqlude.org/libraries/kddockwidgets.html>

<https://github.com/fffaraz/awesome-cpp#asynchronous-event-loop>

[[Category:Windows脚本]]

screen 基本使用

```
1. screen -ls
2. screen -S flask
3. screen -r flask
```


临时记录

qPrintable

- 将QString 转为const char*,类似于: str.toLocal8Bit().constData().

示例:

```
1. QString str ="hello";
2. qPrintable(str);
```

相关的函数还有: qUtf8Printable()、qUtf16Printable()等参见<https://doc.qt.io/qt-6/qtglobal.html#qUtf8Printable>

QReadLocker、QWriteLocker、QMutexLocker

- QMutexLocker (QMutex) 主要是一个方便类,在使用互斥锁时,能够避免在繁琐的加锁解锁操作: 构造(加锁)、析构(解锁)
- QReadLocker、QWriteLocker (QReadWriteLock) 同上,是方便的使用读写锁的类

示例:

```
1. //读写锁(互斥锁类似):
2. QReadWriteLock lock;
3.
4. void writeData(const QByteArray &data)
5. {
6.     QWriteLocker locker(&lock);
7.
8. }
9.
10. //相当于:
11. void writeData(const QByteArray &data)
12. {
13.     lock.lockForWrite();
14.
15.     lock.unlock();
16. }
```

C++ static initialization order fiasco

这个静态初始化失败记得之前面试的时候好像被遇到过

判断double类型是否相等:

```
1. if(std::abs(up - low) < std::numeric_limits<double>::epsilon())
2. {
3.
4. }
```

qt creator f1键不能使用

主要原因是禁用了window的F1~F12功能，开启和关闭使用 Fn + Esc

解决vscode 无法调试问题

[参考链接](#)

解决vscode python 代码格式化问题：

<https://www.code456.com/article/35047.html>

配置vs code 中python的版本：

- 点击 ctrl + shift + p
- 输入： Python: Select Interpreter

设置Qt dialog默认隐藏？

```
1. QCoreApplication::setAttribute(Qt::AA_DisableWindowContextHelpButton, true);
```

代码规范

前置声明

- 尽量避免使用前置声明，转而包含头文件