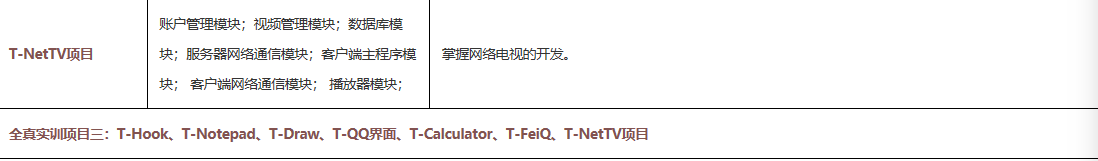


4





**C++学习之路**

# 一、C++编程

## 1、数据类型、变量、内存布局、指针基础；

1. Typede

a）给struct起别名，typede strut student{}stu；stu={}；

B）判断数据类型

C）提高代码移植性：typede int MYINT；

Void test{MYINT a;MYINT b;}

1. Void:万能指针

不需要强制转化就可以赋值

1. Sizeof

非函数，是一个操作符

对于数据类型，要加（）使用，对于变量直接使用

返回类型是 unsigned int 无符号整型

可以统计数组长度

#ifdef DEBUG//如果存在DEBUG就运行下面代码 #if 1//自定义编译器

#else//否则运行下面代码

#endif

\_FILE\_(文件宏) \_LINE\_(行宏) \_DATA\_

1. 内存分区

运行前：代码区和数据区

运行后：栈区和堆区

堆区：

Malloc（sizeof(int)\*10）初始化乱码

Calloc（10，sizeof(int)）初始化0

Realloc （p,sizeof(int)\*20）大于p的空间时 后面重新分配空间

引用：类型 &别名=原名

## 2、字符串、一维数组、二维数组；

宏函数在一定程度比普通的函数效率高：#define MADadd(x,y) x+y

1. 字符串

Char 定义

字符串反转：

字符串格式化：char buf[1024]；memset(buf,0,1024)//将字符数组清零

Sprint(buf,”在%s那个地方”,”公园”)

字符串筛选：sscanf（str,%\*s%d,buf）

\*+忽略部分 //忽略

%[宽度]s //读取指定宽度|

%[a-z] //匹配到选定范围字符，一个匹配失败后续不再匹配

%[^a] //匹配到非a的字符，一个匹配失败后续不再匹配

1. 一位数组

## 3、一级指针，二级指针，三级指针，N级指针概念，指针数组和数组指针；

3.1，一级指针：

不允许像NULL和非法地址赋值，

指针的步长：指针跳一个字节

间接赋值

指针输入输出：输入主调函数分配，被调函数使用，输出反之

字符串后面结束字符\0，sizeof为一个 strlen为结束

一级指针易错点：

3.2，二级指针

输入特性：在主调函数分配内存被调函数使用

输出特性：在被调函数分配内存主调函数使用

练习：FILE \*file=fopen(“./”,”r”);//获取文件

3.3

## **4、结构体、文件的使用**；

结构体：struct preu{};

文件读写：

## 5、动态库的封装和设计；

静态库的创建：创建.h文件初始化一个函数，创建.c的函数，项目属性常规->配置 类型，运行即可创建一个lib文件，再上一级debug里面。

静态库的使用：创建一个项目，将lib文件和.h文件复制进去。直接使用。

静态库的优缺点：移植啊方便，浪费空间。

动态库创建：同上，在.h中初始化函数时\_\_declspec (dllexport) int Sub(int x,int y);（导入函数只能在项目内部使用，导出函数能在外部使用，属于后者）

动态库使用：创建项目，将lib dll .h文件复制进去。在主函数前调用

#pragma comment(lib,"./mydll.lib")，即可使用

## **6、函数指针回调函数**。

函数指针： 声明：void func(){}

//Typedef void(FUNC\_TYPE)();

//FUNC\_TYPE \*pFunc=func;

Void (\* pFunc)()=func;

pFunc();

函数指针数组：

Void func1（）{} Void func1（）{}

Void func1（）{}Void func1（）{}

Void (\* pFunc [4])()=func;

回调函数：提供一个函数可以打印任何数据类型，提供一个通用函数可以打印

## **7、面向对象编程思想**

面向接口编程：

## 8、类的封装，构造和析构、静态成员、对象管理

构造函数的分类：

(参数)有参构造函数和无参构造函数

（类型）普通构造函数和拷贝构造函数

在拷贝函数是参数加const防止修改别人的文件还要加&

静态成员：

静态成员变量：static修饰，对象创建之前就已经分配空间、并且共享、在类 内声明在外初始化。通过对象或者类名进行访问；

静态成员函数：在函数前修饰。访问方式有对象调用和类名访问

对象：

空类占一个空间，成员属性占类空间，其他不占空间；

## 9、类的构造(有参构造函数、无参构造、拷贝构造、默认构造函数)和析构；

系统默认两个构造函数：无参构造函数和拷贝构造函数（const Person p1）

深拷贝和浅拷贝：在进行char\* 分配时 之后堆区分配内存在析构出现问题

## 10、对象动态管理、友元函数、友元类、操作符重载；

## 11、C++编译器对象管理模型分析；

## 12、类对象的动态管理(new/delete)；

new：动态分配内存

Person\* p1=new Person;//相当于分分配队空间，判断分配成功 执行构造函数

Person\* person=new Person（a）;//调用有参构造

在堆区创建类数组时候必须要存在一个默认的构造函数

在堆区开辟数组 可以指定哪个构造函数来进行初始化（Person ps[2]={Person(1),Person(1)}）

delete释放对象

delete p1;//释放单一类

delete[] ps;//释放数组类中存在多个 类友元函数和友元类

## 13运算符重载(一元运算符、二元运算符、运算符重载难点、项目开发中的运算符重载)；

加号运算符的重载

Person operator+(Person &p){} person p3=p1.operator+(p2)

可以直接p3=p1+p2;

左移运算符的重载(全局)

ostream operator<<(ostream &cout,Person &p1)

{cout<<"m\_A"=<<p1.m\_A<<"m\_b"<<p1.m\_B;return cout;}

cout<<p1;

递增运算符重载（ ）

前置++：Person& operator++()

{m\_num++;return \*this;}

cout<<++p1;

后置++：Person operator++(int)

{Person p=\*this;

this->m\_num++;

return p;}

cout<<++p1;

指针运算符的重载（只能指针：托管new出来的对象的释放）

赋值运算符重载(非全局构造函数)

系统会给一个类添加四个函数：默认构造，拷贝构造，析构，operator=;

void operator=（const Person &p）

{//先判断原来堆区是否有数据，有先释放

if（）delete；=NULL；

this->=new char[strlen(name+1)]}

[]运算符重载

动态申请情况加 int& operator[](int index){return this->num;}\

可以直接使用arr[2];

关系运算符重载

直接添加void operator(Person &p)，直接使用类等于

函数运算符重载

类的仿函数，使用和函数一样void operator(string str){cout<<str}

不要重载&&和||

## 14，类的继承、多继承及其二义性、虚继承；

类继承：继承

利用继承可以减少重复单吗的实现

语法：class 子类 :public 父类

继承方式

一，public继承：子类原装继承，父类私有不可以继承

一，protect继承：子类继承，权限改为protec，父类私有不可以继 承

三，private继承：子类继承，权限改为private，父类私有不可以继 承

继承中的对象模型

父类中的私有属性，子类也继承下去，只不过被编译器隐藏了，访问 不到。

继承中的构造和析构

继承中先调用父类的构造，在调用子类的构造，析构的顺序和构造顺 序相反的。

可以利用初始化列表语法，显示指定出调用父类的哪个析构函数，

子类不会继承父类中的构造和析构函数，只有父类才知道如果构造和 析构自己的额属性。

继承中同名的处理

如果子类和父类成员同名，可以通过添加作用域进行调用；

继承中同名静态的处理

优先调用子类成员函数，可以通过类名进行访问

多继承：语法：class a:public B1,public b2

虚继承：利用虚继承Virtual public 类进行修改，可以是类继承变为一个；

## 15、多态(概念、意义、原理剖析、多态案例)；

概念：父子之间继承+虚函数

意义：方便调用不同的类

多态的满足条件：父类中有虚函数、子类重写父类的虚函数、父类的指针或者引用指向子类的对象

多态原理：当父类中存在虚函数后，内部发生结构变化。多了指针vfptr指向虚函数表。

多态案例：计算器

多态好处：对扩展性提高，组织性强，可读性强。

如果父类存在虚函数，那么子类必须要重写父类的虚函数

开闭原则，对扩展进行开放，对修改进行关闭。

## 16、虚函数（virtual）、纯虚函数、抽象类(面向抽象类编程思想案例)；

如果子类中有属性创建在堆区，在多态的情况下不会调用子类的析构代码，导致内存泄漏。

虚析构在析构前面加关键字virtual

纯虚析构 virtual ~函数名（）=0

## 17、函数模板、类模板，模板的继承；

函数模板

基本语法：template<typename T>//告诉编译器T是一个万能数据类型

调用函数模板： 自动类型--让编译器推导出一致的T类型才能使用

显示指定类型--告诉编译器T的类型mywao<int>(a,b)

模板函数和普通函数的区别

普通函数可以隐式类型转换。函数模板如果自动类型不可以隐式转换

调用规则:同时存在优先调用普通函数，可以发生重载，函数模板可以 产生更好的匹配优先使用函数模板

模板的局限性--类的类型（具体化person函数，走这个代码 ）

template<> bool myCamp<Person>(Person a, Person b)

类模板

基本语法：template下面跟着类，这个类称之为类模板

类模板和函数模板的区别：类模板使用时候不可以自动类型推导，可有没人参数

泛型编程--体现在模板技术--将类型参数化

成员函数的创建时机：不是一开始创建，在运行时候创建

类模板作为函数参数：

指定传入类型：void dowork(Person<string, int> &p)

参数模板化：void dowork(Person<T1,T2> &p)

整个类模板化：void dowork(T &p)

类模板的继承问题：如果父类是一个类模板子类子啊做继承时候必须指定父类中的类型 否则无法给父类分配。

class son :public Person<T2>

类模板类外实现成员函数：template<class T1,class T2>

person<T1,T2>::person(T1 name,T2 age)

类模板的份文件编写问题：类模板不生成 需要调用CPP文件。

类模板碰到友元函数：

类模板的数组封装：

## 18、C++类型转换；

## 19、C++输入输出流(标准I/O文件I/O字符流I/O)；

C语言：

文件按字符输入输出（fputc和fgetc）：

输入： FILE \* f\_write = fopen("./text01.txt","w");

char buf[] = "hello world";

for (int i = 0; i < strlen(buf); i++)

{

fputc(buf[i],f\_write);

}

输出：FILE \* f\_read = fopen("./text01.txt","r");

char ch;

while(ch=fgetc(f\_read)!=EOF)//判断是否读到最后

{printf(“%c”,ch);}

C++：



void text01()

{

//打开制定文件

ofstream ofs("./tet.txt",ios::out|ios::trunc);

//判断打开成功

if (!ofs.is\_open())

{

return;

}

//写文件

ofs << "名字：Mingzi " << endl;

ofs << "年龄：19 " << endl;

ofs << "性别：男" << endl;

ofs.close();

}

//读取文件

void text2()

{

ifstream ifa("./tet.txt", ios::in);

if (!ifa.is\_open())

{

return;

}

//第一种方式

char buf[1024] = { 0 };

while (ifa>>buf)

{

cout << buf << endl;

}

//第三种

char buf[1024] = { 0 };

while (!ifa.eof())

{

ifa.getline(buf,sizeof(buf));

cout << buf << endl;

}

//第三种 单个字符读取

char c;

while ((c=ifa.get())!=EOF)

{

cout << c;

}

ifa.close();

}

## 20、C++异常处理(异常机制、异常类型、异常变量、异常层次结构、标准异常库)；

基本语法：try{函数}

catch(捕获类型){}

如果不想处理加throw向上抛出

可抛出一个自定义的数据类型异常 class MyExtion{ void myprint()}

thy{}

catch(MyExtion e){ e.myprint()}

接口声明：如果只允许函数抛出某种类型异常，可以使用异常的接口声明。

void func()throw(类型) 如果throw（空），代表不抛出任何异常

异常变量的声明周期：

MyExcetion e会调用拷贝构造

MyExcetion &e引用方式，接受，建议用这种方式，节省开销

MyExcetion \*e指针方式，接受，抛出，

异常的多态使用：抛出哪种异常就打印出哪种异常

C++标准异常：系统有一些标准的抛出异常，可以使用。

编写自己异常类：class OutOfException:public exception

重写what（

virtual char const\* what() const

{

return this->out.c\_str();

}

）

# 二、数据结构和算法

## 1、这是所有编程语言中最应该学习的部分，程序组成的基础之一。

## 2、顺序存储、链式存储、循环链表；

2.1，顺序储存

2.2，链式储存

链表声明（数据+指针）：struct linkNode

{

int num;

struct lincNode \*next;

};

增，删，改，查。

## 3、双向链表、栈(顺序和链式)、队列(顺序和链式)；

## 4、栈的应用、树基本概念及遍历、二叉树；

## 5、排序算法、并归算法、选择、插入、快速、希尔。

# 三、模板和STL

## 1、STL知识

六大组件：容器、算法、迭代器、仿函数、适配器、空间适配器

容器：序列式容器、关联式容器（有key起到索引作用）

算法：质变算法、非质变算法

迭代器：双向、随机访问

## 2、容器算法迭代器

vector<类型> 别名 容器

尾插 v.push\_buck()

遍历 for\_each加入头文件#include<algorithm>

## 3、常用容器

string容器：构造、赋值。

字符存取：[]和at区别at会抛出异常

拼接、查找、替换：+=append拼接，find查找。replace替换

比较compare：

子串：string substr（int pos=0，int n=npos）const//返回有pos开始 的n个字符组成的字符串

插入：insert

删除：earse

大小写转换：str[i]= toupper(str[i]);

string tmp = lineData.substr(start,pos-1);//substr从strt位置读到pos-1

pos = lineData.find(",",start);//find函数从start查找，返回值int

vector容器：与数组类似但是动态数组，动态分配内存 不是原有空间分配，重新找

交换：swap（）容器大小：size 容量：capacity 重新指定容量：resize

reserve:预留空间但不会初始化

push\_buck插入，pop\_buck删除

vector<int> (v).swap(v);//缩小空间

deque容器：双端数组,没有容量，可以对头部和尾部操作，头部push\_front

list容器：双向链表，

删除remove（elem）删除容器与elem匹配的元素

反转reverse

排序sort //默认从小到大，对于自定义数据类型，必须指定排序规则。

set容器：根据键值自动排序（默认从小到大，可以修改 ），不允许有相同的键值， 贴迭代器只能访问

排序：插入之前通过仿函数指定排序规则。

earse（elme）删除值为elme的元素

count（key）查找key的个数

对组声明：pair<int, string> p(19,string("dsas"));

pair<int, string> p2=make\_pair(19,"dsas");

//第一个元素first 第二个元素second

map容器：关联式容器，根据键值自动排序，无法通过迭代器修改键值，有key和 value每个元素都是队组。

map<int,int> m;

m.insert(pair<int,int>(1,10));//第一种

m.insert(make\_pair(1, 10));//第二种

m.insert(map<int, int>::value\_type(1, 10));//第三种

按照key的值进行删除和查找。



## 4、常用算法

函数对象：类似与函数调用方式，内部可以拥有自己的状态，可以作为函数的参数

谓词：返回值为bool可以成为谓词

内建函数对象：头文件#include<functional>

sort(v.begin(),v.end(),greater<int>());//大于

适配器：将参数进行绑定，bind2nd（）->添加头文件#include<functional>->仿函 数添加参数继承binary\_function<int,int,void>，后面添加const

取反适配器：同上 加not1->添加头文件->仿函数继承un...function,并且 添加const

函数指针适配器：

算法： for\_each(v.begin(),v.end(),函数/仿函数/适配器) //遍历算法

find(v.begin(), v.end(), 3);//查找算法，返回值为值的位置，类型为迭代器

adjacent\_find（v.begin(), v.end()）//查找相邻的重复值 必须为有序序列

binary\_searxh(v.begin(), v.end(), 3); //二分法查找

count(v.begin(), v.end(), 3) //按值统计元素出现的次数，返回元素个数

count\_if(v.begin(), v.end(), 回调函数/仿函数) //按条件进行统计出现的 次数，返回元素个数

merge(v1.begin(), v1.end(),v2.begin(), v2.end(),v3.begin() )//容器元素合并

sort(v.begin(), v.end()) //默认从小到大排序，添加greater<int>()修改为从 大到小

random\_shuffle(v.begin(), v.end()) //随机打乱

reverse（v.begin(), v.end()） //反转

copy(v.begin(), v.end(), v1.begin()) //拷贝到另一个容器

replace(v.begin(), v.end(), oldval,newval) //将指定元素替新元素

replace\_if(v.begin(), v.end(),仿函数 ,3) //将限定条件的元素换成 指定值

swap(v1,v2) //互换两个容器的元素

sccumulate(v.begin(), v.end(), 0)//计算容器元素类总和，添加头文件 numeric，第三个参数是起始累加值

fill（v.begin(), v.end(), 3）//向容器中添加元素

set\_intersection(v1.begin(), v1.end()，v2.begin(), v2.end()，v3.begin())//求 两个set集合的交集，两个集合必须为有序序列

set\_union(v1.begin(), v1.end()，v2.begin(), v2.end()，v3.begin())//求两 个 set集合的并集，两个集合必须为有序序列

set\_difference(v1.begin(), v1.end()，v2.begin(), v2.end()，v3.begin())//求两 个set集合的差集，两个集合必须为有序序列

copy（v1.begin(), itend，ostream\_iterator<int>(cout," ")）;//输出打印操作

## 5、STL综合案例

# 四、UBIX/LINUX网络服务器

## 1、linux简介

优点：开放源、多任务、良好的用户界面、优异的性能和稳定性。

时间戳：从1970年初开始到现在的秒数。

关机：shutdown -h now init 0

备份：

快照（又称还原点，保存拍照时候的系统状态，后期随时可以恢复。侧重在于短期备份，需要频繁备份时候才可以使用快照，一般系统会处于开启状态）

克隆（就是复制，侧重长期备份，做克隆时候必须关闭）

目录：

|  |  |
| --- | --- |
| bin | binary含义二进制，该目录储存一些二进制文件，文件可以被运行 |
| dev | 存放外接设备，如盘、其他光盘。在期中的外接设备是不能直接被使用的，需要挂载（类似windows下的分盘符） |
| etc | 主要存储一些配置文件， |
| home | 表示除了root以外的其他用户的家目录，类似于windows下的User |
| proc | 表示进程，存储linux的进程 |
| sbin | 该目录存放一些可以被执行的二进制文件，但是必须有超级权限的人才能执行 |
| tmp | 表示“临时”的，当系统运行时候产生的临时文件会在这个目录存着。 |
| user | 存放用户自己安装的软件， |
| var | 存放的程序/系统的日志文件的目录。 |
| mnt | 当外接设备需要挂载时候，需要挂载到mnt下。 |

## 2、指令

基础指令

ls：（list）直接使用则列出当前工作目录下的所有文件和文件夹名称。加路径 则列出...下的文件的名称。

-l（#ls -l）表示以详细列表的形势进行展示、

-a（#ls -la）表示显示所有的文件夹（包含隐藏的文件）

-h（#ls -lh）表示列出指定文件下所有文件的名称，列表显示文档大小以可读性较高的形势显示。

pwd：打印当前的绝对路径。

cd+路径：切换当前工作目录

cd ~：表示切换当前用户的家目录

mkdir 路径：创建目录

-p（mkdir -p）同时创建多层不存在的目录。

Touch 文件路径:创建文件。

cp 路径：复制文件到指定位置。

mv 移动的文档路径 保存的位置路径：移动一个文档到新的位置。可以重命名

rm 选项 文档：删除文档

-f；强制删除，不提示

-r：删除递归的目录。

vim 文件路径：打开一个文件，不退出。

输出重定向：

>:覆盖输出

>>:追加输出。

cat：直接打开一个文件，全部输出并且退出

对文件进行合并

进阶指令

df -h:查看磁盘的空间

free -m ：查看内存使用情况。

head -n 文件路径：查看一个文件的前n行。

tail -n 文件路径：查看文件的后n行

tail -f 文件路径：查看一个文件的动态变化

less 文件路径：查看文件，以较少的内容输出，按下辅助功能查看更多。

wc：统计文件内容信息（包含行数，单词书，字节数）

wc -lwc 需要统计的文件路径；(l：表示行数，w：表示单词数，c：表述字节数)

data：表示操作时间日期（读取，设置）。

date +%F：格式显示

date “+%F %T”

date -d “-1 day”显示一天前的时间。

cal：以日历形势显示时间

cal -3：显示三个月的；

cal -y 年份：表示输出某一年份的日历。

clear/ctrl+L：清除终端中已经存在的命令和结果。

管道符 |：一般用于过滤，特殊，扩展处理。。不能单独使用

ls /|grep y：查询有y的文件

ls /|wc -l：统计文件数目

高级指令

honstname：操作服务器的主机名（读取、设置）

-f：输出当前主机名中的FQDN（全限定域名）

id：查询一个用户的基本信息（用户id，用户租id，附加租id）

whoami：我是谁，显示当前登陆的用户名，一般用于shell脚本。

ps -ef：

ps：只要查看服务器的进程信息。

ps -e：等价于“-A”，表示列出全部的进程。

ps -f：显示全部的列

ps -ef|grep 进程名称：查看想看的进程。

top：查看服务器的进程占的资源

du -sh：查看目录的真实大小

find：查找

-name：按照文档名称搜索（支持模糊搜索）

-type：按照文档类型进行搜索（f表示文件 d表示文件夹）

service：用于控制一些软件的服务启动/停止/重启

service 服务器 start/stop/reastart

kill 进程PID：表示杀死进程

ifconfig：用于操作网卡相关的指令，获取网卡的信息

reboot：重新启动计算机

reboot -w：模拟重启但是不重启。只写关机和开机的日志

shutdown：关机

shutdown -h now "关机提示“或者shutdown -h 15：25

uptime：输出计算机的持续在线时间

uname：获取计算机操作系统的相关信息

uname -a：获取完整的信息

netstat-tnlp：查看网络的连接状态

-t：表示只列出tcp协议的连接

-n：表示将字母组合转化为ip地址，将协议转化成端口号来显示。

-l：过滤出state

-p：表示显示发起连接的PID和进程的名称

man：手册（包含了linux的全部命令手册）

如何删除光标前后的内容：ctrl+u ctrl+k

## 3、高级指令

honstname：操作服务器的主机名（读取、设置）

-f：输出当前主机名中的FQDN（全限定域名）

id：查询一个用户的基本信息（用户id，用户租id，附加租id）

whoami：我是谁，显示当前登陆的用户名，一般用于shell脚本。

ps -ef：

ps：只要查看服务器的进程信息。

ps -e：等价于“-A”，表示列出全部的进程。

ps -f：显示全部的列

ps -ef|grep 进程名称：查看想看的进程。

top：查看服务器的进程占的资源

du -sh：查看目录的真实大小

find：查找

-name：按照文档名称搜索（支持模糊搜索）

-type：按照文档类型进行搜索（f表示文件 d表示文件夹）

service：用于控制一些软件的服务启动/停止/重启

service 服务器 start/stop/reastart

kill 进程PID：表示杀死进程

ifconfig：用于操作网卡相关的指令，获取网卡的信息

reboot：重新启动计算机

reboot -w：模拟重启但是不重启。只写关机和开机的日志

shutdown：关机

shutdown -h now "关机提示“或者shutdown -h 15：25

uptime：输出计算机的持续在线时间

uname：获取计算机操作系统的相关信息

uname -a：获取完整的信息

netstat-tnlp：查看网络的连接状态

-t：表示只列出tcp协议的连接

-n：表示将字母组合转化为ip地址，将协议转化成端口号来显示。

-l：过滤出state

-p：表示显示发起连接的PID和进程的名称

man：手册（包含了linux的全部命令手册）

如何删除光标前后的内容：ctrl+u ctrl+k

## 4、Vim

简介：相当于文本编辑器

vim的三种模式：

命令模式，该模式下是不能对文件直接编辑，可以输入快捷键进行删 除、复制行、移动光标、粘贴等等。

编辑模式，在该模式下可以对文件的内容进行编辑；

末行模式，可以在末行输入命令对文件进行操作（搜索、替换、保存、 退出、撤销、高亮等等）

vim打开文件的方式：

vim 文件路径 打开指定文件

vim +数字 文件路径 打开指定文件，光标定位带指定行

vim +/关键词 文件路径 打开指定文件 并且高亮显示关键词

vim 文件路径1 文件路径2 打开多个文件

模式间 的切换

命令模式->末行模式：按“：”进入，按1下Esc/按两下/删除所有命令。 按/表示查找，N/n切换上/下

命令模式->编辑模式：按“i，a”进入，按1下Esc退出。

命令模式

光标移动：

移到行首，shift+6

移到行尾，shift+4或$

移到首行，gg

移到末行，G

移到指定行，数字 G

翻屏,向上翻屏ctrl+b（pgup）或者向下翻屏ctrl+f（pgdn）

复制、粘贴/

复制光标所在行，yy，在想要粘贴的地方按p

以光标所在位置，复制指定行数，数字 yy。

可视化复制，ctrl+v

剪贴/删除

剪贴/删除所在行，dd

剪贴/删除指定行，数字 dd

剪贴/删除，但不上移，D

撤销/恢复

撤销，：+u

恢复：ctrl+r

快速移动，数字+移动键。

末行模式

保存：w

另存：w 路径

退出文件：q

强制退出，修改不保存：q！

调用外部命令：！+外部命令

取消高亮：“：nohl”

替换：

替换光标所在的第一处：s/搜索关键词/替换的词

替换光标所在的一行：s/搜索关键词/替换的词/g

替换所有行中第一个符合的地方：%s/搜索关键词/替换的词

替换所有地方：%s/搜索关键词/替换的词/g

显示行号：set nu/set nonu

在vim打开多个文件，在末行下切换：

查看打开的文件名称：“files”

切换指定文件：“open 文件名”

切换上下文件：“bp/bn”

编辑模式

在光标之后插入：“i”

在光标之前插入：“a”

实用功能

代码着色:显示“syntax on”/：关闭显示“：syntax off”

计算器的使用：进入编辑模式->按下ctrl+r->输入=->输入计算的内 容。

扩展

vim的配置： a.文件打开的时候再末行的配置，临时的

b、个人配置文件（~/.vimrc，如果没用需要新建）

新建.vimrc->进入编辑模式->显示行号“set nu”

c、全局配置文件（/etc/vimrc）

异常退出：删除交换文件（rm -f .文件.swp）可以解决。

别名机制：创建一些属于自己的自定义命令

： ->添加修改->保存退出 ->重启

退出方式：出了“wq”，还支持另外一种保存退出方法“：x”（注意“X”表示给文件加密，再次输入空格，解密）

## 5、linux自有服务（自己存在的，不需要独立去安装的软件服务）

运行模式（运行级别）

打开配置文件：cat /etc/inittab

存在6个运行级别/模式：0 -表示关机级别；

1 -单用户模式；

2-多用户模式，不带NFS;

3-多用户模式，完整的 用户模式

4-没用被使用的模式；

5-X11，完整的图形化界面模式；

6 -表示重启级别（）

切换：init 3/init 5，来切换（临时）。永久切换（vim /etc/inittab，修改5 为3）

用户与用户组的管理

用户管理：

文件：/etc/passwd(存储用户的关键信息)、/etc/group(存储用户组的 关键信息)、/etc/shawn(存储密码的关键信息)

添加用户：“useradd 选项 用户名”

选项：-g：指定用户的用户主组，选项的值可以是用户组的id， 也可以是组名。

-G：指定用户的用户附加组，选项的值可以是用户组的id， 也可以是组名。

-u：uid,用户的id，系统默认从500之后分配，可以自定 义。

修改用户：“usermod 选项 用户名”

选项：-g：指定用户的用户主组，选项的值可以是用户组的id， 也可以是组名。

-G：指定用户的用户附加组，选项的值可以是用户组的id， 也可以是组名。

-u：uid,用户的id，系统默认从500之后分配，可以自定 义。

-l：修改用户名（语法 新的用户名 旧的用户名）

设置密码：“passwd 用户名”

切换用户：su 用户

删除用户：“userdel -r 用户名”（删除用户同时，删除家目录）

用户组的管理：

用户组添加：“groupadd 选项 用户组名”

选项：-g：选择自己设置一个自定义用户组id数字，不指定默 认为500之后。

用户组编辑：“groupmod 选项 用户名”

选项：-g：选择自己设置一个自定义用户组id数字

-n:表示设置新的用户组的名称

用户组删除：“groupdel 用户名”（如果这个组是用户的主组，不允许 删除）

网络设置：网卡位置（etc/sysconfig/network-scripts）

重启网卡：/etc/init.d/network restart

重启单个网卡：“ifdown 网卡名”/"ifup 网卡名"：停止网卡/开启网卡。

软链接（创建快捷方式）：ln -s 目标 新快捷名（ls -l查看快捷方式 和来源）

ssh服务（secure shell安全外壳协议）：远程连接协议、远程文件传输协 议。

端口号：etc/ssh/ssh\_config(注意：注意范围0-65535、不能使用别的 服务已经占用的端口)

服务启动/停止/重启：“service sshd start/stop/restart”，“etc/init.d/sshd start/stop/restart”

远程终端：终端工具主要帮助连接远程服务器，常见终端工具有： xshell、secureCRT、Putty。

获取服务器ip（ifconfig），顺手测试->

ssh服务文件传输：填写ip管理员信息，连接 ->

通过命令行传输文件

## 6、自有服务

设置主机名：（查看“hostname”）

临时设置主机名：“hostname 设置主机名”

永久设计主机名：“vim /etc/sysconfig/network”修改里面主机名内容->修改服 务器的host（etc/hosts）文件，将xbb指向本地

chkconfig:(相当于开机启动项)

开机启动服务查询：“chkconfig --list”

删除服务：“chkconfig --del 服务名”

添加开机启动服务：“chkconfig --add 服务名”

设置服务在某个级别下开机启动：“chkconfig --level 连在一起的启动级别 服 务器 开 /关”

ntp服务（对计算机的时间同步管理操作）

一次性同步：“ntpdate ntp.aliyun.com”

服务自动同步；启动ntpd服务“service ntpd start”->设置自启“chkconfig -- list|grep ntpd； chkconfig --level 35 ntpd on”

防火墙

查看是否开机启动：“chkconfig --list|grep iptables”

状态：“service iptables status”

简单设置防火墙规则：“iptables -A INPUT -p tcp --dport 80 -j ACCEPT”->保 存："“etc/inti.d/iptables save”.

rpm管理（类似于windows的电脑管家的软件管理，）

查询某个软件的安装情况：“rpm -qa|grep 关键词”

卸载软件：“rpm -e 软件的名称”（如果加 --nodeps有依赖关系也会删除）

软件的安装：“rpm -ivh 软件包的完整名称(mnt/dvd/packages/)”

扩展：光盘的挂载和解挂

解挂：“umount 当前设备的挂载点”

挂载：“mount 设备的原始地址(统一在dev下“dev/sr0”) 要挂载的位 置路径()”

cron/crontab计划任务

crontab 选项

-l：列出指定用户的计划任务列表

-e：编辑指定用户的计划任务列表

-u：指定用户名，如果不指定，

-r：删除指定用户的计划任务列表

编辑计划任务：

语法：一行一计划。分 时 日 月 周 需要执行的命令。

四个符号：\*表示所有，-表示区间，/表示每多少个，“，”表示多 个取值

权限：管理员可以通过配置来设置某些用户不允许使用设计计划任务。（打开 etc/cron.deny，添加用户，注意一行一个）

## 7、权限

权限概述

权限介绍：在多用户计算机系统中，权限是特定用户具有特定的系统资源 使用权利。分别有读、写、执行权限

读权限：对于文件夹来说，影响用户是否能够列出目录结构，对于文件来 说，影响用户是否可以查看文件内容。

写权限：对于文件夹来说，影响用户是否可以在文件夹下创建、删除、复 制、移动文件，对于文件来说，影响用户是否可以编辑文件内容。

执行权限：一般都是对于文件来说，特别是脚本文件

身份介绍：

Owner身份

Group身份：（与文件所有者同组的用户）

Others身份：（其他人）

root用户

权限的介绍：

drwxrwx--x ： 第1位文档类型（d文件夹，-文件，|软链接），

第2~4文档所有者权限（r：读，w：写，x：可执 行）

第5~7文档所有者在同一个组的用户的权限（r： 读，w：写，x：可执行），

第8~10文档其他用户的权限（r：读，w：写，x： 可执行）

权限设置

语法：chmod 选项 权限模式 文档

常用选项:-R:表示递归设置权限

权限模式：改文档需要设置的权限信息。

字母形式：

u（用户）、g（所属群体）、o（其他人）、a（所有人）

+（增加权限）、-（减少权限）、=（确定权限）

r（可读）、w（可写）、x（执行）

chmod u+x，g+w，o+rw 文档

chmod u=rwx，g=w，o=rw 文档

数字形式：

0（---），1（--x）、2（-w-）、3（-wx）、4（r--）、5（r-x）、6（rw-）、 7（rwx）

chmod 777 文件

属主和属组设计

属主：所属的用户（文件的主人）

属组：所属的用户组

chown：更改文档的所属用户

“chown -R（递归加） username 文档路径”

chgrp：更改文档的所属用户组

“chgrp -R groupname 文档路径”

chown -R username:groupname 文档路径

扩展

普通用户使用不了超级用户的reboot shutdown init等命令，所以可以使 用sudo（switch user do）来进行权限设置。

默认sudo中是没有除root之外的用户的规则，想使用先配置sudo

sudo配置文件：/etc/sudoers

配置文件使用“visudo” ->配置普通用户的权限（root表示用户名 ALL 表示允许登陆的主机 （ALL）表示谁的身份执行里面表示root身份 ALL 表示当前用户可以执行的命令多个命令使用","分割）

## 8、网络

网络相关命令

ping：检测当前主机与目标主机的连通性。

netstat：查看网络的连接信息。

traceroute：查找 当前主机与目标主机之间的所以的网关

上线 流程

服务器购买：

域名购买：

域名备案 ：

## 9、shell基础

shell基础

是一种c语言编程程序，属于内置脚本。

编写规范：

代码规范：#!/bin/bash(指定告知系统当前的这个脚背要使用的shell解释 器 )

文件命名规范：sh

使用流程：创建sh文件（touch text.sh）->编写shell代码(vim text.sh)->执行 shell脚本(./text.sh)。

进阶

echo输出指令

readonly：只读变量。

接受用户输入：read -p 提示信息 变量名

删除变量：unset 变量名

条件判断：if condition

then

command

elif command1

fi

运算符：

算术 运算符：在相加时候中间需要空格隔开。加expr

关系运算符：

-eq判断两个数是否相等，相等返回为true；

-neq判断两个数是否相等，不相等返回为true；

-qt判断两个数大小，左边大为true

-lt判断两个数大小，右边大为true

-qe判断两个数大小，左边大等于为true

-le判断两个数大小，右边大等于为true

逻辑运算符：！非运算；-o或运算；-a与运算；

字符串运算：= 相等；！= 不相等；-z检验 字符串 长度是否为 0返回true；-n 检验字符串长度不为0返回true；str 检验是否为空 返回true；

文件测试运算符：

-b file 检验文件是否是 块设备文件，是返回为true；[-b $file]

-c file 检验文件是否是字符设备文件，是返回为true；

-d file 检验文件是否是目录，是返回true；

shell脚本附带选项

自定义可执行指令：

if [ $1 = '' ]

then

useradd $2

else

fi

## 10、数据库

Linux下的软件安装方式

源码包

二进制包

yum等傻瓜安装

yum list 列出当前已经装的和可以装的 软件

yum search 名 搜索指定的关键词

yum [-y] install 包名 安装指定的 包（-y表示不确认）

yum [-y] update 包名 更新指定的 包（-y表示不确认）

yum [-y] remove 包名 卸载指定的 包（-y表示不确认）

安装MySQL

安装：yum install mysql-server

初始化：service mysqld start；mysql\_secure\_installation(netstat-tnlp： 查看网络的连接端口号)

启动控制：service mysql start/stop/reastart；

进入：mysql -u 用户名 -p

默认目录和文件位置

数据库储存目录：/var/lib/mysql

配置文件：/etc/my.cnf

mysql基本操作

库操作：SHOW DATANBASRES 显示当前全部数据库

CREATE DATABASE 库名 创建数据库

DROP database 库名 删除数据库

use 库名 切换数据库

表操作：show tables; #显示所有的数据表

create table 表名称 #创建数据表

（

列名称1 数据类型 [not null#不为空] auto\_increment#自增

列名称2 数据类型

列名称3 数据类型

‘’‘’

primary key（主键）

）;

desc 表名称 #描述一个数据表

drop table [if exists] 表名 #删除一个数据表

字段操作

增加记录： insert into 表名称 values（值1，值2....）

insert into 表名称 (列1，列2....) values（值1，值2....）

更新记录： update 表名称 set 列名称1=新值，列名称2=新 值....where 列名称=某值

查询记录： select 列名称1，列名称2... from 表名称 where 条 件；

select\*from 表名称 where 条件 #查询所有的

删除记录： delete from 表名称 where 列名称=值

备份和还原

备份（导出）：（需要退出sql）

全量备份（数据+结构）：#mysqldump -uroot -p123456 -A>备份文 件路径

指定库备份（数据+结构）：#mysqldump -uroot -p123456 库名>备 份文件路径

多个库备份（数据+结构）：#mysqldump -uroot -p123456 -detabases db1 db2>备份文件路径

案例：每一分钟自动备份一次数据库

建立一个 sh文件

编辑输入filename="text\_" `date +'%Y%m%d%H%M%S'`".sql";mysqldump -uroot -pxbb151726 test >/root/$filename

建立任务：crontab -e ;\*\*\*\*\* /root/text07.sh

还原（导入）：

mysql >source 备份文件路径和文件名字

#mysql -uroot -pxbb151726 <备份文件路径

扩展

设置mysql连接字符集：mysql> set names utf-8

mysql的远程管理工具：

B/S：b是指浏览器，s指服务器，百度搜索

C/S：c是指客户端，s指服务器，例如qq等应用程序

解决允许远程登陆问题：

进入mysql数据库->执行“select host，user fromuser”

## 11、项目上线

编译安装雨卸载nginx：（流行的web服务器软件）

下载文件：wget http://nginx.org/download/nginx-1.19.2.tar.gz

解压：tar -zxvf 文件 tar

跳转文件：cd 文件

配置文件：./configure --prefix=/usr/local/nginx --with-pcre with- zlib=/usr/loacl/src/zlib[前提再内部下载一个zlib源码包:](安装前装yum install pcre-devel；yum install zlib-devel）

开始安装：make；make install

运行：usr/local/nginx/sbin/nginx

重载配置文件：usr/local/nginx/sbin/nginx -s reload

关于lamp：

linux +apach+mysql+php：

lamp环境部署：

php和Apache安装：

安装php：yum install php;(包含httpd)

启动apache：service httpd start(存在问题：vim /etc/httpd/conf/httpd.conf;去掉servername前面的#)

mysql的安装和初始化：

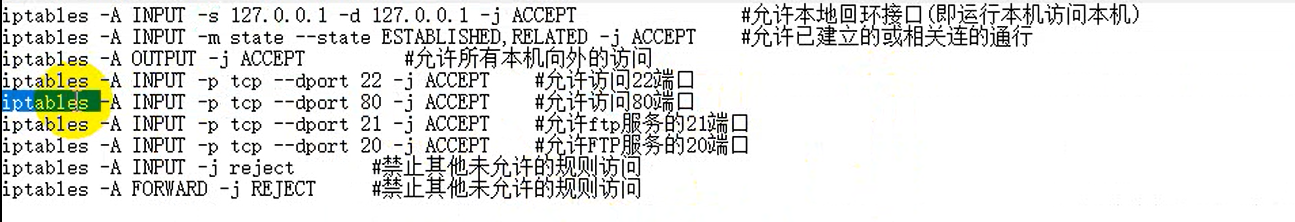
安装：yum install mysql-server

service mysqld start

初始化：mysql\_secure\_installation(密码y,移除y，n，y)

项目上线：

熟练使用系统调用；



熟练Unix/Linux内存管理，进程，线程调度；

熟悉网络服务器开发方式，熟练编写TCP，UCP网络服务程序；

掌握同步/异步IO模型在网络编程中的使用方式。

## 12，进阶1

### 12.1、库

静态库的创作和使用：

动态库的创作和使用：

制作：将原文件生成.o(gcc a.c b.c -c -fpic(fPIC))->打包(gcc -shared a.o b.o -o libxxx.so)

使用：gcc main.c -I ./include -L ./lib -lxxx -o app.

(动态库无法加载：

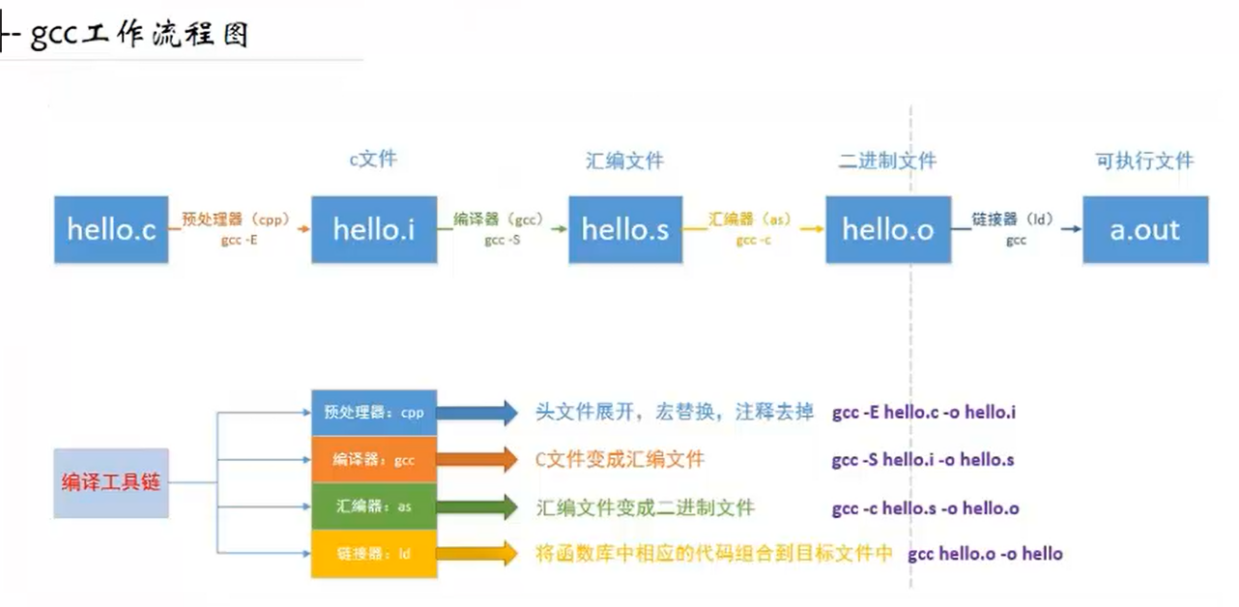
临时：export LD\_LIBRARY\_PATH=动态库的路径

永久：vim ~/.bashrc->添加#export LD\_LIBRARY\_PATH=动态库的路径用户级别

vim /etc/profile ………….

)

### 12.2、gcc



### 12.3、makefile

规则：app: a.c b.c c.c

gcc a.c b.c c.c -o app

编写一：app: a.c b.c c.c

gcc a.c b.c c.c -o app（效率低，修改一个文件，所有得文件都会被修改）

编写二：app: a.c b.c c.c

gcc a.c b.c c.c -o app

a.o:a.c

gcc a.c -c

b.o:b.c

gcc b.c -c

c.o:c.c

gcc c.c -c

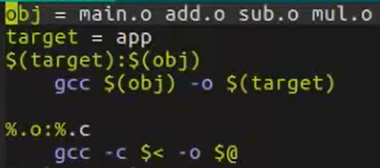
编写三：

自动变量：obj=10

变量取值：aa=$(obj)

Makefile的自带变量：大写：CPPFLAGS ；CC

自动变量：$@规则中目标；$< 规则中第一个依赖;$^;规则中所有依赖



http://c.biancheng.net/view/7161.html（makefile知识点）

### 12.4、jdb调试

启动gdb：gdb app（可执行程序名字）;传参 set args xxx xxx

查看代码：l 行号/函数名。Set listsize n 显示行数

断点操作：

设置断点：b 行号；b函数名

查看断点：info(i) b

删除断点：d num（编号）；d num1 num2

设置断点无效：dis num

断点生效：ena num

设置断点条件：b 行号 if 变量==var

调试相关命令：

打印变量的值：p 变量名字

打印变量的类型：ptype 变量名

向下单步调试： n (next不会进入函数体)

s（step -会进入到函数体内部）

继续运行gdb，停在下一个断点位置：c（continue）

退出gdb：q

变量的自动显示：display 变量名/undisplay 取消变量名

从循环体跳出：until （不能有断点）

直接设置变量等于一个值：set var 变量名=value；

### 12.5、文件io

open:

函数原型：int open(const char \*pathname,int flasgs)

int open(const char \*pathname,int flasgs,mode\_t mode)

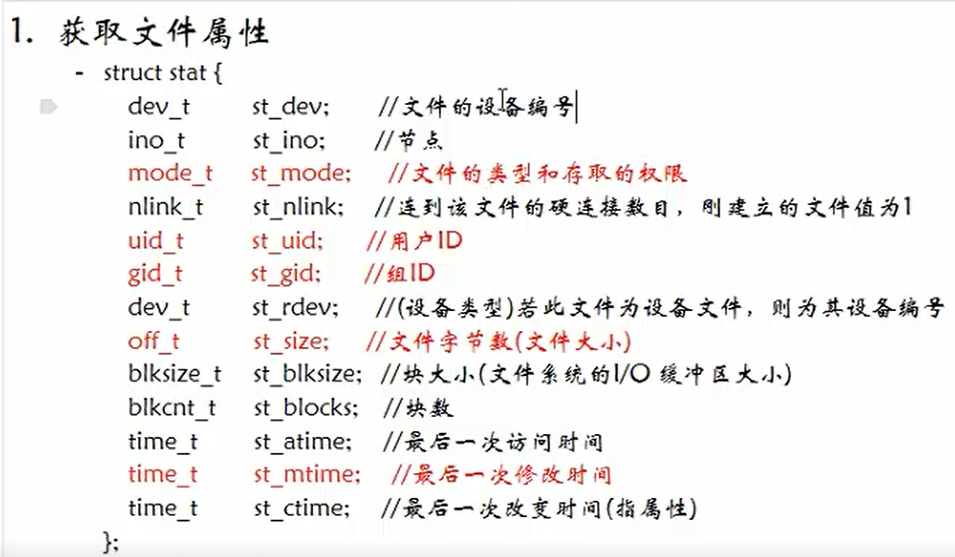
参数：

Flags:必选项互斥的（O\_RDONLY,O\_WRONLY,O\_RDWR）

可选项（创建文件：O\_CREAT,追加文件：O\_APPEND,文件截断： O\_TRUNC,设置非阻塞：O\_NONBLOCK）

mode:

### 12.6、stat/lstat函数



### 12.7、文件属性函数

### 12.8、目录操作相关函数

### 12.9、目录遍历相关函数

### 12.10、dup

# 五、Mysql

掌握SQL语言的实用技巧。Oracle,MySQL数据库的使用方式。

# 六、Qt

掌握QT类库构架，图形界面开发模型；

掌握QT开发技巧，消息机制，图形处理；

掌握QT网络编程，UDP，TCP使用方式；

掌握QT文件处理方式，序列化；

掌握QT在windows，linux，ios，android不同平台下的移植技术。

## 1、qt概述

是一种跨平台的C++图形用户界面应用程序框架，面向对象的

## 2，按钮控件API

//创建按钮类

QPushButton \*button=new QPushButton;

button->setParent(this);

button->setText("第一个按钮");

//第二种方法

QPushButton \*button2=new QPushButton("按钮",this);

button2->move(100,100);

resize(400,400);//重置窗口大小

setFixedSize(400,400);//固定窗口大小

setWindowTitle("第一个窗口");//设置窗口标题

## 3,QT的坐标体系

左上角为0，0

## 4，信号和槽

连接方式：connect(1，2，3，4）参数1：信号的发送着。参数2：发送信号；参数3：信号的接受着。参数3：处理的槽函数

· 断开连接：disconnect（）

拓展： 信号可以连接信号，

一个信号可以连接多个槽函数

多个信号可以连接同一个槽函数

信号和槽函数参数必须一一对应

信号和槽函数，信号的参数可以多余槽函数

信号槽可以断开连接disconnect()

lambda表达式：可修改标识符mutable,可以修改传递的数据，改变不了本体。

int ret=[]()->int{return 100;}

## 5，QMainWindow

菜单栏创建(只能有一个)：

QMenuBar \*bar=menuBar()//创建菜单栏

setMenuBar(bar);//菜单栏放入窗口

QMenu \*fileMenu= bar->addMenu("文件");//创建菜单

QMenu \*editMenu= bar->addMenu("编辑");

fileMenu->addAction("新建");//创建菜单项

fileMenu->addSEparator();//添加分割线

fileMenu->addAction("打开");

工具栏(可以有多个)：

QToolBar \*toolBar=new QToolBar(this);

addToolBar(Qt::LeftToolBarArea, toolBar);//默认停靠左边

toolBar->setAllowedAreas(Qt::LeftToolBarArea);//设置只允许左右停靠

toolBar->setFloatable(false);//设置浮动

toolBar->setMovable(false); //设置是否移动

toolBar->addAction(newAction);//添加内容

toolBar->addAction(openAction);

QPushButton \* button1=new QPushButton("aa",this);//添加控件

toolBar->addWidget(button1);

状态栏(只能有一个)：

QStatusBar \* stbar=statusBar();

setStatusBar(stu);

QLabel \*label=new QLabel("信息提示",this);//放入标签控件

stu->addWidget(label);

QLabel \*label1=new QLabel("右侧信息提示",this);//放入标签控件

stu->addPermanentWidget(label1);

铆接部件(浮动窗口)可以有多个:

QDockWidget \* dockWidget=new QDockWidget("浮动",this);

addDockWidget(Qt::BottomDockWidgetArea,dockWidget);

dockWidget->setAllowedAreas(Qt::TopDockWidgetArea|Qt::BottomDockWidgetArea);//设置只允许上下浮动

中心部件(能设计一个):

QTextEdit \*edit=new QTextEdit(this);

setCentralWidget(edit);

资源文件：

拷贝到当前目录

右键项目添加新文件:Qt：Qt recourse File->给资源起名

生成res.qrc

open in editor编辑资源

添加后缀和文件

## 7，对话框(QDialog)

分类：

//模态创建（不可以对其他窗口进行操作）

QDialog dialog(this);

dialog.resize(200,100);

dialog.exec();

qDebug()<<"模态对话框弹出了";

//非模态创建（可以对其他窗口进行操作）

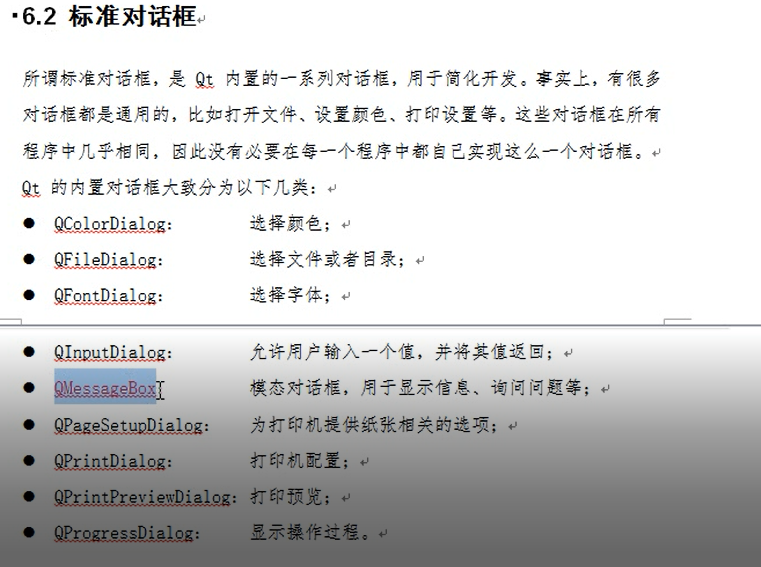
QDialog \*dialog2=new QDialog(this);

dialog2->resize(200,100);

dialog2->setAttribute(Qt::WA\_DeleteOnClose);//释放

dialog2->show();

标准对话框：



其他：

颜色对话框

QColor color= QColorDialog::getColor(QColor(255,0,0));

qDebug()<<"r="<<color.red()<<"g="<<color.green()<<"b="<<color.blue();

文件对话框（参数1：父亲，参数2：标题，参数3：默认打开路径，参数4：过滤的文件格式。返回值是选取的路径）

QString str= QFileDialog::getOpenFileName(this,"打开文件","E:\\桌面","(\*txt)");

字体对话框

bool flg;

QFont font= QFontDialog::getFont(&flg,QFont("宋体",12));

qDebug()<<"字体："<<font.family().toUtf8().data()<<"字号"<<font.pointSize()<<"是否加粗："<<font.bold()<<"是否倾斜："<<font.italic();

消息对话框；（参数1：父类，参数2：标题，参数3：内容，参数4:按键类型）

QMessageBox::critical(this,"提示","错误");//错误对话框

QMessageBox::information(this,"info","信息提示");//信息提示框

if(QMessageBox::Save==QMessageBox::question(this,"question","提问",QMessageBox::Save|QMessageBox::Cancel))//提问对话框

{

qDebug()<<"保存对话框";

}

else

{

qDebug()<<"取消保存";

}

QMessageBox::warning(this,"warning","警告");//警示对话框

标准文件对话框：

## 8，常用控件

按钮组：

QPushButton,常用按钮

QToolButton，工具按钮，显示图片，如果想显示文字，修改风格（toolButtonStyle）,凸起风格（autoBaise）

radioButton,单选按钮，设置默认Ui->rtnMan->aetChecked(true)

checkbox,多选按钮，监听状态。

QlistWidget:

QlistWidgetItem \*item=new ()一行类容

ui->listwidget->additem(item);

item->settextallignment(Qt::AlignCenter);//设置居中

QTreeWidget树控件

ui->treeWidget->setHeaderLabels(QStringList()<<"英雄"<<"英雄介绍");//设置水平头

//父亲节点

QTreeWidgetItem\* liItem=new QTreeWidgetItem(QStringList()<<"力量");

ui->treeWidget->addTopLevelItem(liItem);

//子节点

QTreeWidgetItem\* li=new QTreeWidgetItem(QStringList()<<"刚备注"<<"前排坦克，能够吸收伤害等，伤害力报表");

liItem->addChild(li);

QTableWidget

ui->tableWidget->setColumnCount(3);//创建行和列

ui->tableWidget->setRowCount(5);

ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"姓名"<<"性别"<<"年龄");//添加表头

QStringList nameList;

nameList<<"亚瑟"<<"赵云"<<"张飞"<<"关羽"<<"花木兰 ";

QList<QString> sexList;

sexList<<"男"<<"男"<<"男"<<"男"<<"女";

for(int i=0;i<5;i++)

{

int cont=0;

ui->tableWidget->setItem(i,cont++,new QTableWidgetItem(nameList[i]));

ui->tableWidget->setItem(i,cont++,new QTableWidgetItem(sexList[i]));

ui->tableWidget->setItem(i,cont++,new QTableWidgetItem(QString::number(i+15)));

}

其他控件：

stackWIdget控件：ui->stackedWidget->setCurrentLndex(1);

下拉框：ui->comboBox->addItem("奔驰);

Qlable(显示图片)：ui->lbl\_image->setPixmap(QPixmap(":/iamge/png"));

(显示动态)：ui->lbl\_image->setmovie(movie);

movie->start();

自定义控件

添加控件类（QT 设计师界面类），设计两个控件

添加widget，右键添加提升新加控件类。

在新类中提供两个函数。

在widget中实现功能，信号槽监听。

## 9，Qt中消息机制和事件

鼠标事件：

enterEvent(QMouseEvent \*event)//鼠标进入事件

leavelEvent(QMouseEvent \*event)//鼠标离开事件

mousePressEvent(QMouseEvent \*event)//鼠标按压事件

if(event->button()==Qt::LeftButton)//如果鼠标左键按下 实现

QString str=QString("鼠标按下 x=%1 y=%2 ").arg(event->x()).arg(event->y());

qDebug()<<str;

mouseReleaseEvent(QMouseEvent \*event)//鼠标松下事件

mouseMoveEvent(QMouseEvent \*event)//鼠标移动事件

定时器：

利用事件：timerEvent(QTimerEvent \*ev)

启动定时器：startTimer(1000);//单位是毫秒

定时器名称：调用ev判断启动定时器的名称

第二种：//定时器2

QTimer \* time=new QTimer(this);

time->start(500);

connect(time,&QTimer::timeout,[=](){

static int num=1;

ui->label\_2 ->setText(QString::number(num++));

});

event事件：负责事件分发，也可以做拦截（不建议）

返回值bool event(QEvent \*e)，如果true不向下分发，return Qlable：：event（返回父类执行）

e->type()==鼠标按下。

事件过滤器：（在事件分发之前在做一次拦截）

安装：ui->label-installEventFilter（this）;

重写：bool eventFilter(Qobject \*,QEvent \*);//参数1 判断label，参数2：判断鼠标按键事件

## 10，描述和绘图设备

Qpainter绘图

//实例化画家

QPainter painter(this);

//实例化画笔

QPen pen(QColor(255,0,0));

pen.setWidth(3);//设置画笔的宽度

pen.setStyle(Qt::DotLine);//设置画笔风格

painter.setPen(pen);//画家使用

//实例化画刷

QBrush brush(Qt::red);

brush.setStyle(Qt::Dense6Pattern);

painter.setBrush(brush);

painter.drawLine(QPoint(0,0),QPoint(100,100));//画直线

painter.drawRect(QRect(25,250,25,250));//画矩形

高级设置：

QPainter painter(this);

painter.setRenderHint(QPainter::HighQualityAntialiasing);//抗锯齿能力，效率低

painter.translate(100,0);//移动画家

painter.save();//保存画家状态

painter.restore();//还原画家状态

painter.drawPixmap(posX,10,300,300,QPixmap(":/image/2.jpg"));//画家画图片

绘图设备：

QPixmap：pixmap绘图设备，专门为平台做了显示优化

QPixmap pix(300,300);//定义pix画布

pix.fill(Qt::white);//填充颜色

QPainter painter(&pix);

painter.setPen(QPen(Qt::green));

painter.drawEllipse(QPoint(150,150),100,100);

pix.save("E:\\pix.png");//保存到磁盘的图片

QBitmap：//前者的子类，黑白的

QImage：//像素级访问做优化

QImage image(300,300,QImage::Format\_RGB32);//定义pix画布

image.fill(Qt::white);//填充颜色

QPainter painter(&image);

painter.setPen(QPen(Qt::green));

painter.drawEllipse(QPoint(150,150),100,100);

image.save("E:\\image.png");//保存到磁盘的图片

//在paintEvent重载写入

QPainter painter(this)

QImage img;

img.load("照片地址");

for（i）

for（j）

QRgb value=qrge(255,0,0)

img.setPixel(i,j,value);

QPicture：//记录和重现QPainter

QPicture pic;

QPainter painter;

painter.begin(&pic);//开始画

painter.setPen(QPen(Qt::green));

painter.drawEllipse(QPoint(150,150),100,100);

painter.end();//结束画

pic.save("E:\\pic.zt");

//重现

QPainter painter(this)

QPicture pic;

pic.load("E:\\pic.zt");

painter.drawPicture(0,0,pic);

## 11,QFile对文件进行读和写

connect(ui->pushButton,&QPushButton::clicked,[=](){

QString path=QFileDialog::getOpenFileName(this,"打开文件","E:\\QT\\myProject");//选取文件，返回值为文件路径

ui->lineEdit->setText(path);//显示路径

//读取文件

QFile file(path);//参数读取的路径

file.open(QIODevice::ReadOnly);//设置打开方式

QByteArray arry=file.readAll();

ui->textEdit->setText(arry);//读取放入textEdit中

file.close();

//写文件

file.open(QIODevice::Append);//设置打开方式

file.write("aaaa");

file.close();

//读取文件信息

QFileInfo info(path);

qDebug()<<"大小"<<info.size()<<"后缀名："<<info.suffix()<<"文件名称："<<info.fileName()<<"文件路径"<<info.filePath();

qDebug()<<"创建日期："<<info.created().toString("yy/MM/dd hh:mm:ss");

qDebug()<<"最后修改日期："<<info.lastModified().toString("yy/MM/dd hh:mm:ss");

# 七、MFC

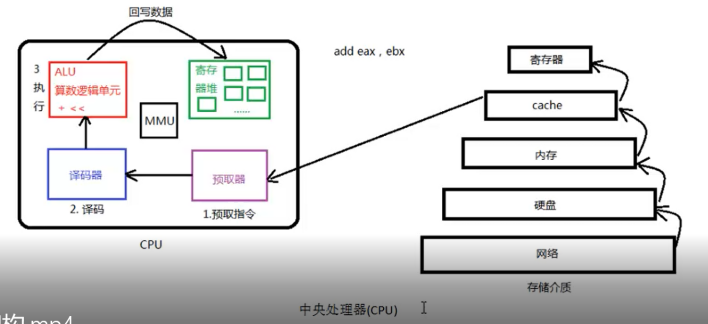
# 八、linux服务器开发

## 1，linux系统编程

### 第一天

进程相关概念

Cpu：中央处理器

:

Pcb进程控制块

环境变量：指定操作系统运行环境变量的一些参数，

常用的环境变量：PATH、SHELL、TERM、LANG、HOME

Getenv函数：获取环境变量值

Setenv函数

Unsetenv函数：

环境进程 ：

进程控制

Fock函数 ：创建一个子进程。

Exec函数族：

### 第二天

### 第三天

## 2、linux网络编程