

# 项目四 Linux下编程——命令行交互mySQL

# 项目的实验目标

## 初步熟悉Linux操作系统下的编程

### 命令行界面（终端界面）

- 通过命令行敲入命令交互，而不是图形界面实现交互
- 通过终端命令，深入理解操作系统底层原理
- 需要使用Linux，要学会使用Linux命令

本次实验推荐使用Ubuntu操作系统、vim编辑器和GDB调试。

- 熟悉gcc/g++编译和Makefile等概念以及使用

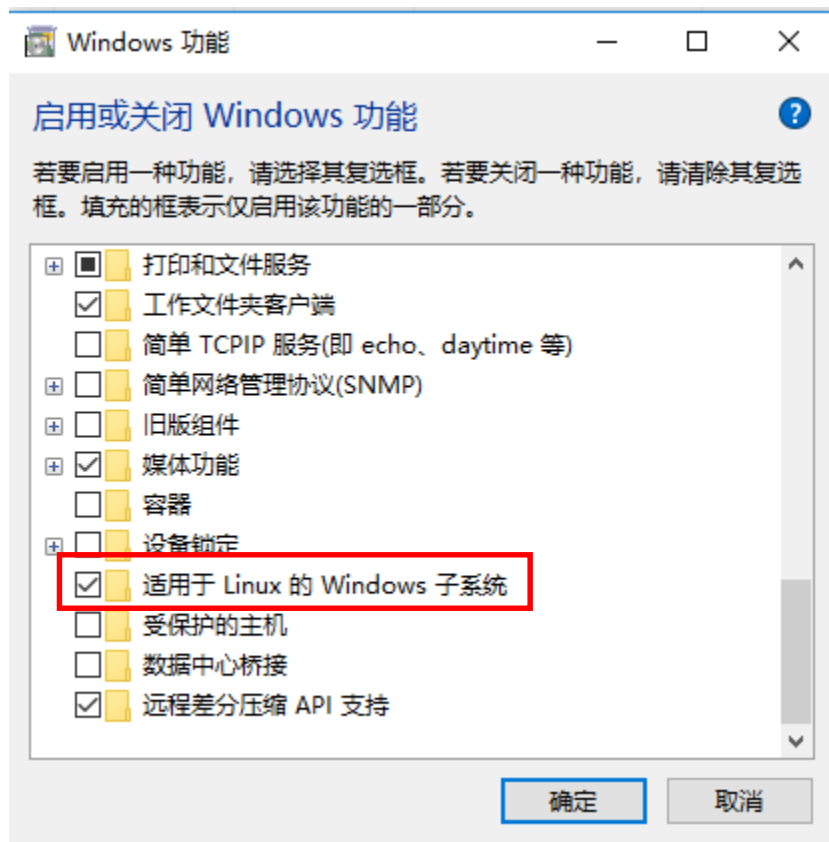
# Bash On Windows安装教程

- Windows用户使用Linux (难度递减)
  - 选项一：下载镜像，制作安装盘，安装Windows/Linux双系统，建议先装windows  
<https://www.jianshu.com/p/d79821e9fdb6> <https://blog.csdn.net/flyyufenfei/article/details/79187656>
  - 选项二：在虚拟机安装Linux系统，建议使用Vmware  
<https://zhuanlan.zhihu.com/p/38797088> <https://www.jianshu.com/p/3379892948da>
  - 选项三：Win10中的Bash On Windows，不需要安装新系统即可体验Linux，相当于在系统中安装一个app (**推荐使用**)
- Bash On Windows能让Windows用户能在系统中运行Linux子系统，也就是说你可以直接在Windows中获得原生Linux Bash级别的体验
- 安装Bash On Windows前提：
  - 64位windows
  - Windows10 版本号 > 14316 (window键+r键，输入winver)

# 打开Windows中“启用或关闭Windows功能”

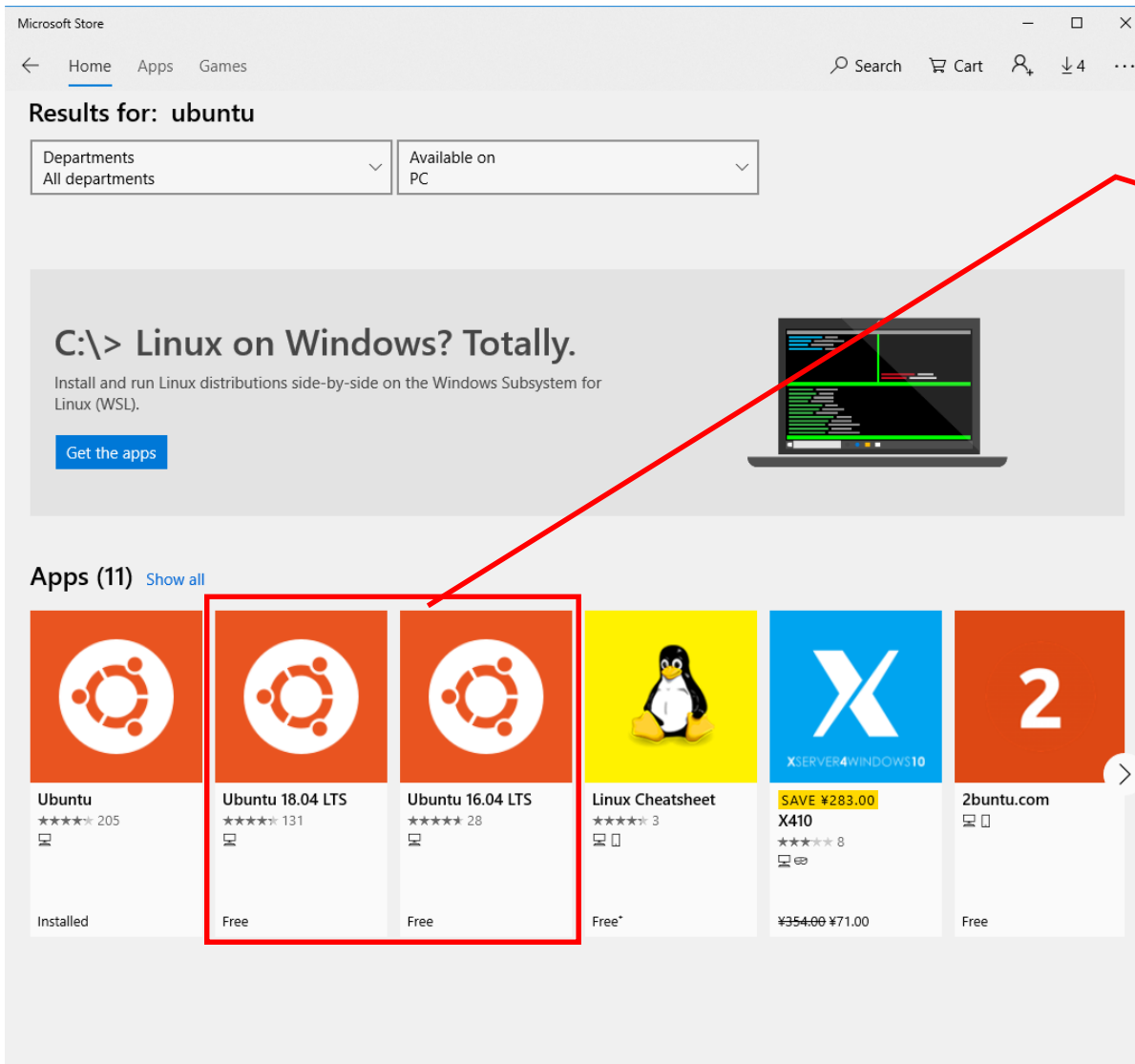


# 打开Windows中Linux子系统功能



确定并重启

# 在Microsoft Store中搜索Ubuntu



选择最新版本，下载安装，安装完成后，点击“启动”。等待安装完成，设定账号，密码。即可使用Ubuntu。  
以后使用，可以在最近添加菜单中选择运行。  
ubuntu

## 安装g++ , vim, gdb, make

- 输入命令: `sudo apt-get update`, 然后输入账号对应的密码后运行更新。
- 输入命令: `sudo apt-get install g++`
- 检测 g++ 是否安装成功, 输入命令: `g++ -v` 可以看到安装的g++的信息
- 输入命令: `sudo apt-get install vim gdb make`

# Linux常用命令（一）

- `cd` 文件夹名 进入已存在的文件夹。
- `mkdir` 文件夹xxx 新建一个名为“文件夹xxx”的空文件夹。
- `rm` 文件aaa 删除名为“文件aaa”的文件
- `rm -r` 文件夹bbb 删除整个文件夹bbb以及文件夹bbb内所有文件
- `ls` 列举出当前文件夹下所有的可见文件
- `cat` 文件aaa 仅查看阅读“文件aaa”里的内容
- `vim` 文件aaa

编辑名为“文件aaa”的文件，如果该文件未被创建，则创建并编辑，敲入命令回车后离开当前界面，进入vim编辑界面。



## Linux常用命令（二）

- `cp a.cpp b.cpp`

将a.cpp所有内容复制到b.cpp中，如果b.cpp非空会被a.cpp覆盖，b.cpp不存在会自动新建一个b.cpp

- `cp a.cpp /mnt/c/class`

将ubuntu当前目录下的a.cpp文件拷贝到windows下c盘的class目录下

- `pwd`

回车后返回当前绝对路径，即目前在哪个文件夹下

- `g++ hello.cpp -g -o hello.o`

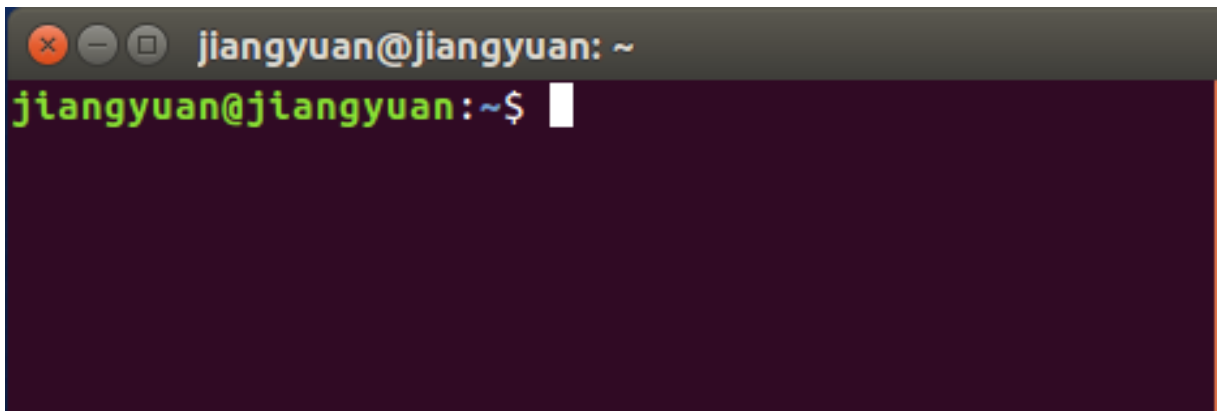
g++为C++编译命令，-g -o 为参数，上面的命令为编译名为hello.cpp的源码，编译完成后生成名为hello.o的可执行文件。

- `./hello.o`

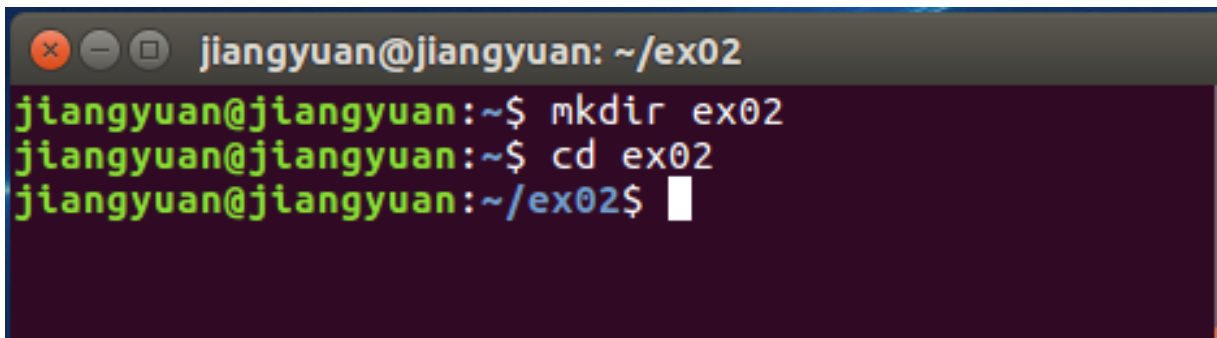
使用命令“./”执行刚刚编译完成的“可执行文件hello.o”

# Linux下编写C++程序示例

- 键盘按下Ctrl + Alt +T 快捷键打开终端，如下：

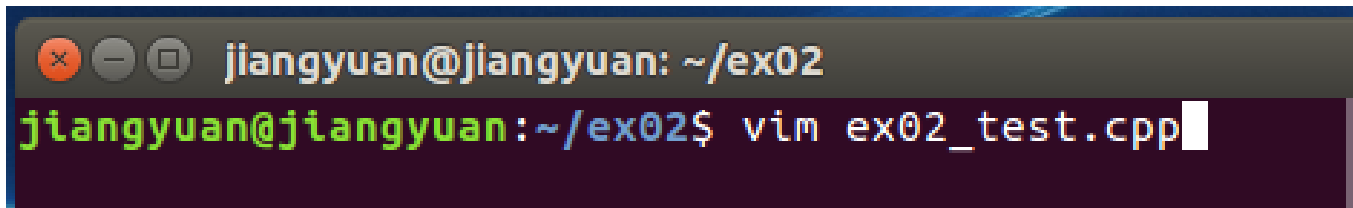
A terminal window with a dark purple background. The title bar shows the user 'jiangyuan' and the current directory '~'. The prompt 'jiangyuan@jiangyuan: ~\$' is displayed in green text, followed by a white cursor.

- 新建文件夹，存放项目代码文件，使用命令：mkdir 文件夹名
- 使用命令：cd 文件夹名，进入刚刚新建的文件夹内

A terminal window showing the execution of two commands. The title bar shows the user 'jiangyuan' and the current directory '~/ex02'. The prompt is 'jiangyuan@jiangyuan: ~/ex02\$'. The first command 'mkdir ex02' is entered and executed. The second command 'cd ex02' is entered and executed. The prompt now shows the current directory as '~/ex02\$'.

# Linux下编写C++程序示例

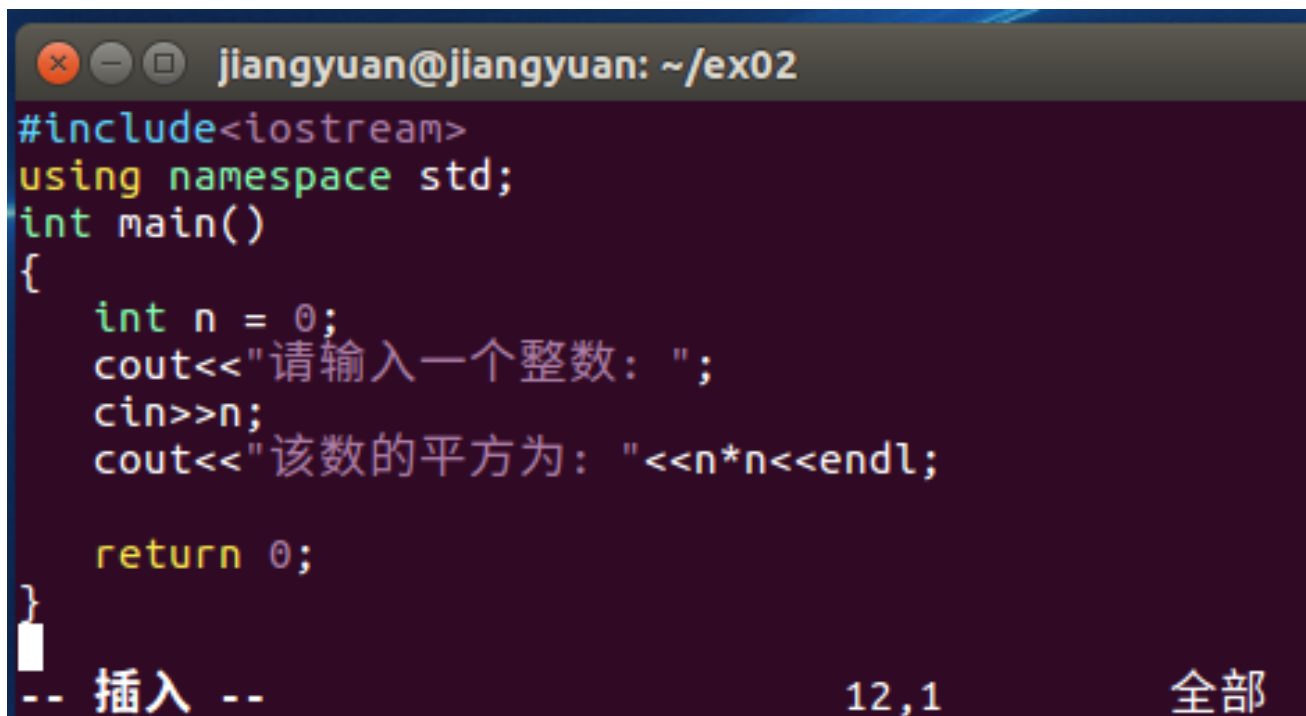
- 在新建的ex02文件夹下，使用vim命令创建并编辑代码文件，敲入 `vim ex02_test.cpp` 回车后会离开当前界面，进入vim编辑界面。如果提示“vim未安装”则使用命令“`sudo apt-get install vim`”回车后输入电脑密码进行安装即可。

A terminal window with a dark background. The title bar shows window control buttons and the text 'jiangyuan@jiangyuan: ~/ex02'. The prompt is 'jiangyuan@jiangyuan:~/ex02\$' and the command 'vim ex02\_test.cpp' is entered, followed by a white cursor.

```
jiangyuan@jiangyuan: ~/ex02
jiangyuan@jiangyuan:~/ex02$ vim ex02_test.cpp
```

# Linux下编写C++程序示例

- 进入编辑后，键盘按下“i”键进入插入模式，插入模式编辑框左下角会出现“--插入--”，这时候可以写代码了，我们输入如下图测试代码。



The screenshot shows a terminal window with a dark background. The title bar at the top reads "jiangyuan@jiangyuan: ~/ex02". The code is written in a light blue font. The code is as follows:

```
#include<iostream>
using namespace std;
int main()
{
    int n = 0;
    cout<<"请输入一个整数: ";
    cin>>n;
    cout<<"该数的平方为: "<<n*n<<endl;

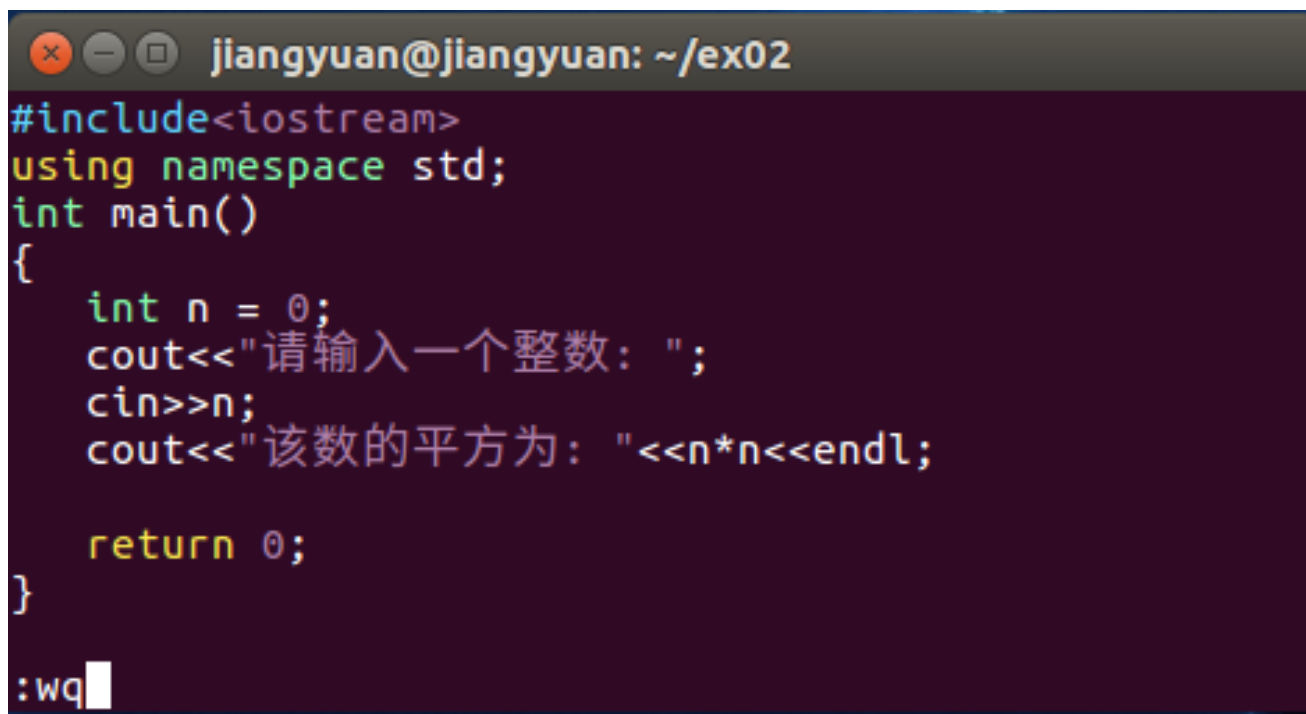
    return 0;
}
```

At the bottom left of the editor, the text "-- 插入 --" is displayed. At the bottom right, the text "12,1" and "全部" are visible.

- 编辑完成后，按“esc”键退出插入模式。

# Linux下编写C++程序示例

- 按“**esc**”键退出插入模式后，键盘输入冒号“**:**”，然后输入**wq**回车保存并退出，如图左下角，返回原来的界面。



```
jiangyuan@jiangyuan: ~/ex02
#include<iostream>
using namespace std;
int main()
{
    int n = 0;
    cout<<"请输入一个整数: ";
    cin>>n;
    cout<<"该数的平方为: "<<n*n<<endl;

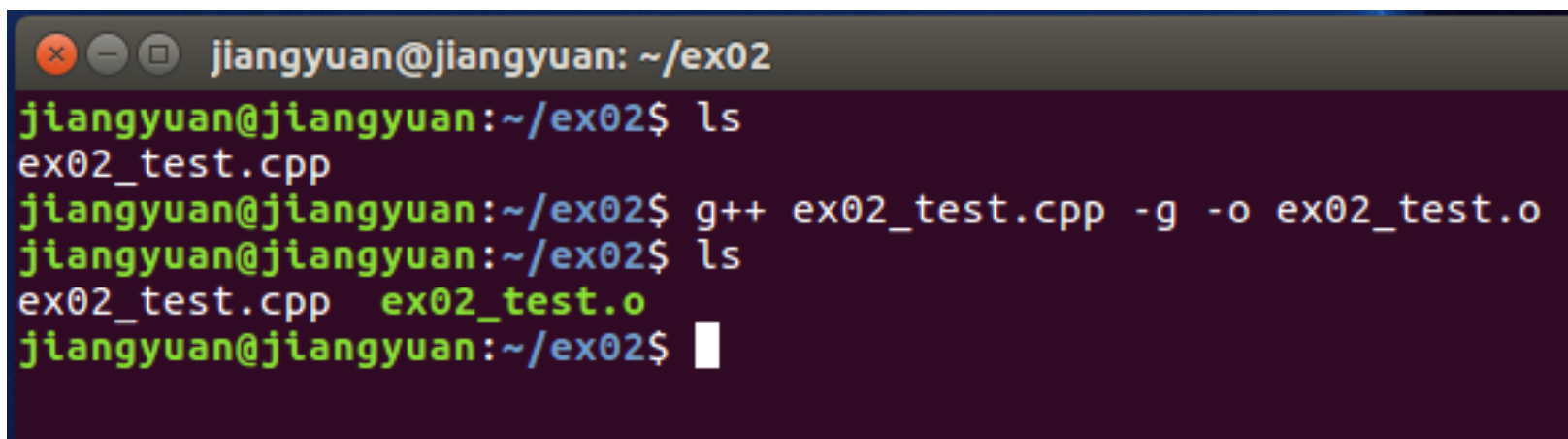
    return 0;
}

:wq
```

- **wq** 保存并退出
- **q!** 不保存仅退出

# Linux下编写C++程序示例

- `ls` 查看当前目录，发现刚刚编辑并保存的`ex02_test.cpp`文件已经存在
- 我们使用`g++`命令编译源代码，得到`ex02_test.o`可执行文件



```
jiangyuan@jiangyuan: ~/ex02
jiangyuan@jiangyuan:~/ex02$ ls
ex02_test.cpp
jiangyuan@jiangyuan:~/ex02$ g++ ex02_test.cpp -g -o ex02_test.o
jiangyuan@jiangyuan:~/ex02$ ls
ex02_test.cpp  ex02_test.o
jiangyuan@jiangyuan:~/ex02$
```

# Linux下编写C++程序示例

- 使用“.”命令执行可执行文件

```
jiangyuan@jiangyuan:~/ex02$ ls
ex02_test.cpp  ex02_test.o
jiangyuan@jiangyuan:~/ex02$
jiangyuan@jiangyuan:~/ex02$ ./ex02_test.o
请输入一个整数: █
```

- 输入一个整数，回车，即可执行代码内刚刚编写的程序。

```
jiangyuan@jiangyuan:~/ex02$ ./ex02_test.o
请输入一个整数: 4
该数的平方为: 16
jiangyuan@jiangyuan:~/ex02$ █
```

# GDB调试

- 使用“gdb 可执行文件名”命令回车进入调试模式

```
jiangyuan@jiangyuan: ~/ex02
jiangyuan@jiangyuan:~/ex02$ gdb ex02_test.o
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ex02_test.o...done.
(gdb) █
```



# GDB调试

- 在GDB模式下，使用“list n”命令从第n行列举源代码，一次显示10行，可以继续回车显示余下的源代码。

```
(gdb) list 1
1      #include<iostream>
2      using namespace std;
3      int main()
4      {
5          int n = 0;
6          cout<<"请输入一个整数: ";
7          cin>>n;
8          cout<<"该数的平方为: "<<n*n<<endl;
9
10         return 0;
(gdb)
11     }
12
(gdb)
```

# GDB调试

- 在GDB模式下，使用“break n”命令从第n行插入调试断点。

```
(gdb) break 8
Breakpoint 1 at 0x4009d5: file ex02_test.cpp, line 8.
(gdb) █
```

- 在GDB模式下，使用“run”命令执行代码，如果有提前插入断点，则从开头一直运行到断点的地方停止，如下所示，程序停止在8行。

```
(gdb) run
Starting program: /home/jiangyuan/ex02/ex02_test.o
请输入一个整数: 4

Breakpoint 1, main () at ex02_test.cpp:8
8          cout<<"该数的平方为: "<<n*n<<endl;
(gdb) █
```

# GDB调试

- 程序在14行断点处暂停，可以使用命令“s”进入单步调试，每次s回车后仅执行一步，达到调试的目的。

```
Breakpoint 1, main () at ex02_test.cpp:8
8          cout<<"该数的平方为: "<<n*n<<endl;
(gdb) s
该数的平方为: 16
10          return 0;
(gdb) s
11      }
```

- 单步调试结束后，可以直接使用“c”命令直接跑完接下来的所有代码。

```
(gdb) c
Continuing.
[Inferior 1 (process 4245) exited normally]
(gdb) □
```

# GDB调试

- 输入“quit”命令退出GDB调试模式，回到终端。

```
(gdb) quit  
jiangyuan@jiangyuan:~/ex02$
```

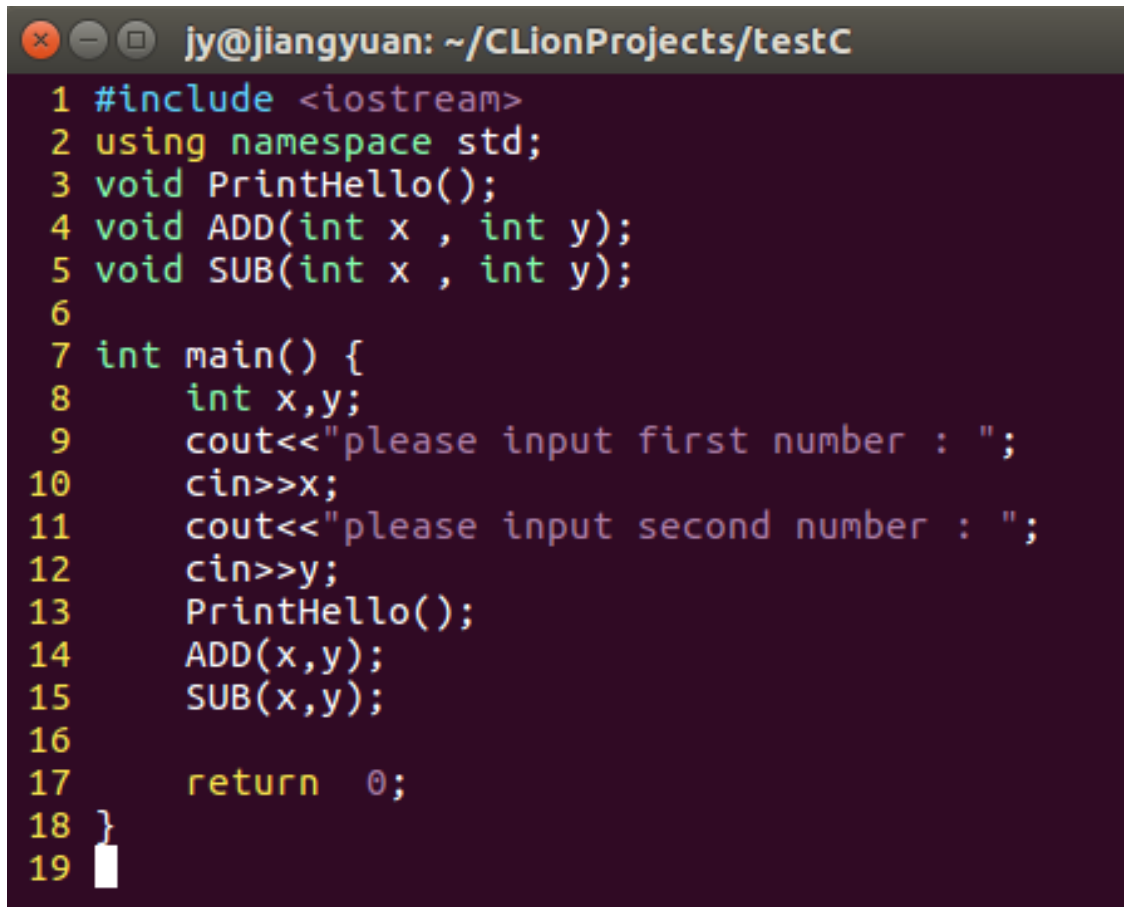
- 示例程序ex02\_test 编写、编译、调试和演示结束，更多的vim操作和GDB调试命令可以自行网上查阅。

# makefile的使用

- 前面的示例代码里，只有一个源代码文件ex02\_test.cpp 所以也只会生成一个可执行文件ex02\_test.o
- 但是当项目由多个源代码文件构成，再按前面的步骤一个个编译，将会很繁琐而且也会生成过多的可执行xxx.o文件。
- 这个时候要么把所有代码全部写进一个.cpp文件里，按照前面的介绍，正常使用g++编译；或者多个.cpp文件使用makefile进行统一编译，编译多个.c文件，但是只生成一个可执行文件。

# makefile的使用

- 假设有四个源代码文件：main.cpp、hello.cpp、add.cpp、sub.cpp
- main.cpp是主程序，如图所示；
- hello.cpp内定义了输出“Hello NJU”的函数PrintHello();
- add.cpp内定义了计算两个数和的函数ADD();
- sub.cpp内定义了计算两个数差的函数SUB();

A screenshot of a code editor window showing a C++ source file. The window title is 'jy@jiangyuan: ~/CLionProjects/testC'. The code is as follows:

```
1 #include <iostream>
2 using namespace std;
3 void PrintHello();
4 void ADD(int x , int y);
5 void SUB(int x , int y);
6
7 int main() {
8     int x,y;
9     cout<<"please input first number : ";
10    cin>>x;
11    cout<<"please input second number : ";
12    cin>>y;
13    PrintHello();
14    ADD(x,y);
15    SUB(x,y);
16
17    return 0;
18 }
19
```

# makefile的使用

- 我们可以执行命令 `vim makefile` 回车编辑makefile文件：
- ```
main: main.o hello.o add.o sub.o
    g++ -o main main.o hello.o add.o sub.o
.....
```
- 第二行g++前面的空格部分是<tab>键，不是“空格键”
- wq保存并退出后，回到终端。

# makefile的使用

```
main: main.o sub.o add.o hello.o
    g++ -o main main.o sub.o add.o hello.o
main.o: main.cpp
    g++ -c -o main.o main.cpp
add.o: add.cpp
    g++ -c -o add.o add.cpp
sub.o: sub.cpp
    g++ -c -o sub.o sub.cpp
hello.o: hello.cpp
    g++ -c -o hello.o hello.cpp
clean:
    rm main *.o
```

- 1、第一行冒号之前的main，称之为目标（target），被认为是这条语句所要处理的对象，即：要编译的这个程序main。
- 2、冒号后面的部分（main.o add.o sub.o hello.o），称之为依赖关系表，也就是编译main所需要的文件，这些文件只要有一个发生了变化，就会触发该语句的第三部分--命令部分，上述的.o文件又依赖于同名的.cpp文件。
- 3、只要将上面这几行语句写入一个名为Makefile或者makefile的文件，然后在终端中输入make命令，就会看到它按照makefile中的设定去编译程序。



# makefile的使用

- 上面的makefile解决了每次编译都要输入很长指令的问题，但是每次新增一个cpp文件都要打开makefile文件进行更改，如果想要自动获取所有的cpp，编译成一个可执行文件，该怎么做呢？
- 首先介绍一下变量，在makefile中，可以用 '=' 定义变量，例如  
    target = main  
表示定义了一个名叫target的变量，它的内容是main，\$(target)表示取它的值。
- 通配符 \*，\*.cpp表示所有以cpp结尾的文件
- wildcard关键字，通配符在变量中会失效，因此用wildcard关键字来使其有效，例如\$(wildcard \*.cpp) 表示当前目录下所有cpp文件

# makefile的使用

```
target = main
source = $(wildcard *.cpp)
OBJS = $(source:%.cpp=%.o)

$(target): $(OBJS)
    g++ -o $(target) $(OBJS)
%.o:%.cpp
    g++ -c -o $@ $<

clean:
    rm main *.o
```

"%.cpp=%.o"表示将source里所有的cpp文件改成对应的.o文件;

"%.o:%.cpp"表示每一个.o文件都是由它同名的.cpp文件生成;

"\$@"表示语句的目标, 这里就是"%.o";

"\$<"表示依赖关系表的第一项, 也就是%.cpp。

# makefile的使用

- 执行make main命令，自动编译四个源代码文件，仅生成一个可执行文件main，可以发现ls查阅之后，只有main文件为绿色，表示为可执行文件。再执行make clean 命令删除所有无用的 .o 文件

```
jiangyuan@jiangyuan: ~/ex02
jiangyuan@jiangyuan:~/ex02$ vim makefile
jiangyuan@jiangyuan:~/ex02$ make main
g++ -c -o main.o main.cpp
g++ -c -o sub.o sub.cpp
g++ -c -o add.o add.cpp
g++ -c -o hello.o hello.cpp
g++ -o main main.o hello.o add.o sub.o
jiangyuan@jiangyuan:~/ex02$ ls
add.cpp  hello.cpp  main      main.o    sub.cpp
add.o    hello.o    main.cpp  makefile  sub.o
jiangyuan@jiangyuan:~/ex02$ make clean
rm *.o
jiangyuan@jiangyuan:~/ex02$ ls
add.cpp  hello.cpp  main  main.cpp  makefile  sub.cpp
jiangyuan@jiangyuan:~/ex02$
```

# makefile的使用

- 执行main, 命令 ./main 回车

```
jiangyuan@jiangyuan:~/ex02$ ls
add.cpp  hello.cpp  main  main.cpp  makefile  sub.cpp
jiangyuan@jiangyuan:~/ex02$
jiangyuan@jiangyuan:~/ex02$ ./main
please input first number : 5
please input second number : 10
Hello NJU
5 + 10 = 15
5 - 10 = -5
jiangyuan@jiangyuan:~/ex02$
```

- 简单用法演示完毕, 其中自动生成的main.o、hello.o、sub.o、add.o可以 rm 命令删除, 不影响main (那个绿色文件) 的执行结果。
- makefile的其他用法请同学们自行网上查阅相关材料。

# SQL

- SQL是什么

- SQL，指结构化查询语言，全称是 Structured Query Language。
- SQL 可以让你访问和处理数据库。

- SQL能做什么

- SQL可以面向数据库查询
- SQL可以向数据库插入新的记录
- SQL可以从数据库中删除记录
- SQL可以修改数据库
- .....

# 项目四实验内容

- 在Linux下实现命令行交互的简易SQL——mySQL
- 要求实现：从命令行输入一条SQL语句（指令），解析该语句，执行相应的数据库操作，返回对应的结果。
- 要求实现的指令有：

CREATE TABLE

DROP TABLE

INSERT INTO

DELETE

UPDATE

SELECT

# 项目要求

- 本次实验在Linux下完成，并且答辩测试也是在Linux下进行。
- 数据库中所有TABLE均以TXT文件的形式保存在本地
- 所有对数据库的修改需要更新到数据库文件中
- 此外还需要维护一个文件表示数据库状态，文件内容为每行两个字符串，分别为TABLE名 对应的数据库文件名，表示这些TABLE在数据库中，每次启动mySQL需加载这些数据库

# 项目要求

- 运行你的程序，应该首先进入命令行界面，然后输入mysql进入到数据库程序，开始执行指令，直到输入quit指令，退出mysql程序。注意，这里的“~\$”以及“(mysql)==>”都是由你的程序输出，并非系统自带的软件或函数功能。

```
~$ mysql
(mysql)==>
(mysql)==>
(mysql)==> quit
~$
```



# 项目介绍

- CREATE TABLE 语句，格式为：

- CREATE TABLE name (column1,column2,...,columnT) TO file

- 创建一个TABLE，TABLE的名字是name，共有T列，其中columni为第i列的属性名
    - 所有属性用括号包含，不同的属性名以逗号隔开
    - file为TABLE在本地存储的文件名
    - 例如 CREATE TABLE Student (学号,姓名,专业) TO student.txt

- CREATE TABLE name FROM filename

- 从一个已经存在的数据库文件中读取数据创建TABLE
    - 例如 CREATE TABLE Student FROM student.txt

# 项目介绍

```
(mysql)==> CREATE TABLE Student (学号,姓名,专业) TO student.txt
| +-----+-----+-----+-----+
| | ID  | 学号  | 姓名  | 专业  |
| |-----+-----+-----+-----+
(mysql)==> █
```

```
xiayf@nlp-62:~/Project2$ cat student.txt
学号 姓名 专业
```

如上所示，创建一个名为Student的表格，生成student.txt文件。

ID列不是表格内容，展示表格时自动添加。

需要注意的是，一个数据库文件不能被多个TABLE共享，即以第二种方式创建TABLE的时候不能从已经在数据库中的文件创建。

# 项目介绍

- DROP TABLE 语句，格式为：
  - DROP TABLE name
    - 从数据库中删除名为name的TABLE
- TABLE LIST 语句，格式为：
  - TABLE LIST
    - 为了方便查看，使用TABLE LIST打印当前数据库中所有TABLE

# 项目介绍

```
(mysql)==> CREATE TABLE Student (学号,姓名,专业) TO student.txt
+-----+-----+-----+-----+
| ID   | 学号   | 姓名   | 专业   |
+-----+-----+-----+-----+
(mysql)==> CREATE TABLE Lecture (课程编号,课程名称,任课老师) TO lecture.txt
+-----+-----+-----+-----+
| ID   | 课程编号 | 课程名称 | 任课老师 |
+-----+-----+-----+-----+
(mysql)==> TABLE LIST
total:2
  Student: (3, 0) [学号,姓名,专业]
  Lecture: (3, 0) [课程编号,课程名称,任课老师]
(mysql)==>
```

TABLE LIST展示当前所有表格的名称，长度（列，行）以及属性列表

```
(mysql)==> DROP TABLE Lecture
(mysql)==> TABLE LIST
total:1
  Student: (3, 0) [学号,姓名,专业]
(mysql)==>
```

删除Lecture表，所以只剩下Student

# 项目介绍

- INSERT INTO 语句，格式为：
  - INSERT INTO name VALUES (value1,value2,...,valueT)
    - 向TABLE name里插入一行，共T个属性的值，T应该与name的列数一致
    - 所有属性值用括号包含，不同的属性值以英文逗号隔开
    - 例如 INSERT INTO Student VALUES (170000001,王二小,计算机科学与技术)
  - INSERT INTO name (column1,column2,...) VALUES (value1,value2,...)
    - 向TABLE name里插入一行，但是仅指定的列有值，value与column对应
    - 缺省的列应设置为默认值
    - 例如 INSERT INTO Student (学号,姓名) VALUES (170000001,王二小)

# 项目介绍

```
(mysql)==> INSERT INTO Student VALUES (170000001,王二小,计算机科学与技术)
```

| ID | 学号        | 姓名  | 专业       |
|----|-----------|-----|----------|
| 1  | 170000001 | 王二小 | 计算机科学与技术 |

```
(mysql)==> 
```

向Student插入一条记录，对应文件修改为：

```
xiayf@nlp-62:~/Project2$ cat student.txt
```

```
学号 姓名 专业
```

```
170000001 王二小 计算机科学与技术
```

```
(mysql)==> INSERT INTO Student (学号,姓名) VALUES (170000002,陈业秀)
```

| ID | 学号        | 姓名  | 专业       |
|----|-----------|-----|----------|
| 1  | 170000001 | 王二小 | 计算机科学与技术 |
| 2  | 170000002 | 陈业秀 |          |

```
(mysql)==> 
```

缺省的属性  
默认为空

# 项目介绍

- DELETE 语句，格式为：
  - DELETE FROM name WHERE column = value
    - 从TABLE name里删除若干行
    - 删除的行满足条件 column = value
    - 例如 DELETE FROM name WHERE 姓名 = 王二小
  - DELETE \* FROM name
    - 从TABLE name里删除所有行
    - 注意这里与DROP TABLE的区别是保留TABLE的结构，没有删除TABLE

# 项目介绍

```
(mysql)==> INSERT INTO Student (学号,姓名) VALUES (170000002,陈业秀)
```

| ID | 学号        | 姓名  | 专业       |
|----|-----------|-----|----------|
| 1  | 170000001 | 王二小 | 计算机科学与技术 |
| 2  | 170000002 | 陈业秀 |          |

```
(mysql)==> DELETE FROM Student WHERE 姓名 = 陈业秀
```

| ID | 学号        | 姓名  | 专业       |
|----|-----------|-----|----------|
| 1  | 170000001 | 王二小 | 计算机科学与技术 |

```
(mysql)==> 
```

删除一条记录



# 项目介绍

- UPDATE 语句，格式为：
  - UPDATE name SET column1 = value1, column2 = value2, ...
    - 更新TABLE name若干列的值
    - 将column i列的值设置为value i，修改多列时以逗号和一个空格 ‘,’ 隔开
    - 例如 UPDATE Student SET 学号 = 1700000000, 专业 = 计算机科学与技术
  - UPDATE name SET column1 = value1, column2 = value2, ... WHERE  
column = value
    - 更新TABLE name若干列的值，方法同上
    - 在更新的列中，只更新满足column = value条件的行
    - 例如 UPDATE Student SET 学号 = 1700000000, 专业 = 计算机科学与技术 WHERE  
姓名 = 王二小

# 项目介绍

```
(mysql)==> UPDATE Student SET 专业 = 软件工程
```

| ID | 学号        | 姓名  | 专业   |
|----|-----------|-----|------|
| 1  | 170000001 | 王二小 | 软件工程 |
| 2  | 170000002 | 陈业秀 | 软件工程 |

```
(mysql)==> UPDATE Student SET 专业 = 大数据占卜 WHERE 姓名 = 王二小
```

| ID | 学号        | 姓名  | 专业    |
|----|-----------|-----|-------|
| 1  | 170000001 | 王二小 | 大数据占卜 |
| 2  | 170000002 | 陈业秀 | 软件工程  |

```
(mysql)==> █
```

更新表，不指定特定行则修改全部

# 项目介绍

- SELECT 语句，格式为：
  - SELECT column1,column2,... FROM name
    - 从TABLE name里选择若干列展示
    - 不同列以逗号 ‘,’ 隔开
    - 例如 SELECT 学号,姓名 FROM Student
  - SELECT \* FROM name
    - 从TABLE name里选择所有列展示，即展示整个TABLE
    - 例如 SELECT \* FROM Student

# 项目介绍

```
(mysql)==> SELECT 学号,姓名 FROM Student
```

| ID | 学号        | 姓名  |
|----|-----------|-----|
| 1  | 170000001 | 王二小 |
| 2  | 170000002 | 陈业秀 |

```
(mysql)==> SELECT * FROM Student
```

| ID | 学号        | 姓名  | 专业    |
|----|-----------|-----|-------|
| 1  | 170000001 | 王二小 | 大数据占卜 |
| 2  | 170000002 | 陈业秀 | 软件工程  |

```
(mysql)==> █
```

查询表中记录的若干属性，\*代表查看全部属性

# 项目介绍

- SELECT 语句增加关键字，格式为：
  - SELECT DISTINCT column1,column2,... FROM name
    - 在TABLE name中，一列可能会有重复的值
    - DISTINCT关键字表示只展示不同的值
    - 例如 SELECT DISTINCT 专业 FROM Student
  - SELECT \* FROM name ORDER BY column1,column2,... ASC|DESC
    - 对返回的查询结果按某些列进行排序展示
    - 如果排序的条件有多列，以逗号 ‘,’ 隔开，ASC表示升序，DESC表示降序
    - 例如 SELECT \* FROM Student ORDER BY 学号 ASC

# 项目介绍

```
(mysql)==> UPDATE Student SET 专业 = 计算机科学与技术
```

| ID | 学号        | 姓名  | 专业       |
|----|-----------|-----|----------|
| 1  | 170000001 | 王二小 | 计算机科学与技术 |
| 2  | 170000002 | 陈业秀 | 计算机科学与技术 |

```
(mysql)==> SELECT DISTINCT 专业 FROM Student
```

| ID | 专业       |
|----|----------|
| 1  | 计算机科学与技术 |

```
(mysql)==> 
```

查看不同的专业

```
(mysql)==> SELECT * FROM Student ORDER BY 学号 DESC
```

| ID | 学号        | 姓名  | 专业       |
|----|-----------|-----|----------|
| 1  | 170000002 | 陈业秀 | 计算机科学与技术 |
| 2  | 170000001 | 王二小 | 计算机科学与技术 |

```
(mysql)==> 
```

按学号降序排列

# 项目介绍

- SELECT 语句增加关键字，格式为：
  - SELECT column1,column2,... FROM name WHERE column = value
    - 在TABLE name中，选择若干列进行展示
    - 在这些列中，只展示满足条件 column = value的行
    - 例如 SELECT 专业 FROM Student WHERE 姓名 = 王二小
  - SELECT column1,column2,... FROM name TO file
    - 将查询的结果写入文件file中
    - 写入文件需要保持TABLE的结构，即能以写入后的file生成新的TABLE (CREATE TABLE table\_name FROM file)
    - 例如 SELECT \* FROM Student WHERE 专业 = 计算机 TO 计算机系学生名单.txt

# 项目介绍

```
(mysql)==> SELECT * FROM Student WHERE 姓名 = 陈业秀
+-----+-----+-----+-----+
| ID   | 学号       | 姓名   | 专业               |
+-----+-----+-----+-----+
| 1    | 170000002 | 陈业秀 | 计算机科学与技术   |
+-----+-----+-----+-----+
(mysql)==> █
```

按姓名查询

```
(mysql)==> SELECT * FROM Student WHERE 姓名 = 王二小 TO class1.txt
+-----+-----+-----+-----+
| ID   | 学号       | 姓名   | 专业               |
+-----+-----+-----+-----+
| 1    | 170000001 | 王二小 | 计算机科学与技术   |
+-----+-----+-----+-----+
(mysql)==> █
```

将查询结果保存到指定文件

```
xiayf@nlp-62:~/Project2$ cat class1.txt
学号 姓名 专业
170000001 王二小 计算机科学与技术

xiayf@nlp-62:~/Project2$ █
```



# 题目要求

- 严格按照描述的指令格式，实现基本的指令功能，但是为了美观的虚线框可以省略或者自行设计修改，命令执行效果不得更改
  - 建议用正则表达式分析指令格式
- 命令行界面应简洁、美观，便于操作
- 充分考虑数据库操作中的非法操作，给出合理清晰的错误反馈
- 设计的程序应当分模块，各模块功能明确，有良好的代码风格。

## 额外创意，扩展功能

- 在完成基本功能的前提下，任意发挥，目标：更易用，更合理，更强大。
  1. 在前面你应该可以注意到，我们在多条语句中使用了WHERE关键字，它后面的内容表示一个条件判断，除了已经实现的 '=' 判断外，还有  
'>' ; '<' ; '!=' ; '>=' ; '<=' ; 'BETWEEN' ; 'LIKE' ; 'IN'  
等等，挑选你感兴趣的实现。

## 额外创意，扩展功能

2. 更复杂的功能，函数。在某些数据库文件中，一些属性和数值有关，你可以对这些数值类的属性进行一些操作，例如求平均数、计数、求最大值最小值等。

语句 `SELECT MAX(学号) FROM Student`

表示查询TABLE Student中学号最大的那一个。

更多的函数和功能自由发挥，函数返回格式和结果展示自行合理设计。

## 额外创意，扩展功能

3. 语句的复合。实现了函数之后，我们就可以将一些函数作为 WHERE 关键字的判断条件，例如

```
SELECT * FROM Student WHERE 学号 < (SELECT MAX(学号)
FROM STUDENT)
```

4. 将多条语句按顺序写进文件，称为一个事务文件，直接运行文件,执行文件中所有操作
5. 任何你想到的，好玩的功能，炫酷的界面，增强系统鲁棒性的方法，提高运行效率的算法等等

## 实验安排和评分（项目总分/10，计入总分）

- 12月14日**前**提交设计PPT，给出数据结构设计、功能分解、模块设计及核心函数声明。满分30分。
- 12月14日抽取部分学生上台讲解设计PPT。
- 12月21日**前**提交代码及用户手册。
- 12月21日答辩：根据课程网站上提交的代码，进行功能测试检查及提问。
- 完成题目要求功能的程序满分90分，实现扩展功能，实现1个10分，上限30分。
- 用户手册在设计PPT的基础上，增加程序操作说明（程序运行截图+文字说明）。满分30分。
- 答辩过程中，正确回答问题得20分。