./ root directory; **~** home directory **.**//present **..**//dad

Useful Command

mkdir dir1 dir2 “dir 3”

rmdir dir//only work when dir is empty

rm -r(recursive) -f(force, 不需要确认) lab or -rf

pwd//prints the name of the current directory

mv file “newfile”//rename file;

mv dir1 dir2//move dir1 to dir2

ls -l = **ll**//List dir in long content **e.g.:**

-rwxr-xr-x 2 user group 4096 May 12 12:40 filename

ls -a// show hidden file(name begin with “.”)

touch file//create file or update its timestamp

cat file//输出文件内容

cp file1 file2// copy

cp -r(递归) dir1 dir2//同上,但是包括文件夹内部所有文件

chmod [who(u,g,o,a)][operator(+-=)][permission] file

4=r, 2=w, 1=x

permission: user, group, other;read, write, execute

grep “abc” file//返回带有 abc 的行

grep -E //启用扩展**正则表达式**，以下为部分正则:

.//任意单字符; ^//行开始; $//行结束;

?//前一个字符 0or1 次; +//~>0 次; *//~>=0 次

()//group; |//or; [1-5]//匹配 1 到 5 中的数字字符

pattern {n}(匹配出现 n 次); {n,m}(n-m 次，可取边界)

cut -d ',' -f2 file//以”,”为分隔，提取每行第二块文本

diff file1 file2// 比对 file1file2 不同

‘> eee’:插入 eee ; ‘< ccc’:更改为 ccc

wc file//统计行数, (-w)words 和(-c)bytes,用-l 仅显示行数**, -m 字符数** e.g.

1(lines) 14(words) 514(bytes) file

sort file//按字母序排序文件

-n:数字排序;-r:倒序;-kn:对第 n fields 排序;-t:设置分隔符

uniq file//删除相邻的重复行

spell file//显示 incorrect 的词

su//superuser mode

find path -name “filename” -type(-f:只搜 file;-d:只搜 dir)

sed [选项] '命令' 文件, -n//仅显示处理后信息

a 新增 c/s 取代 d 删除 i 插入 p 打印 **e.g.**

sed “2,3p”打印 2-3 行; sed “s/1/2/g”全局替换 1th 匹配项

standard I/O

0-stdin; **1**-stdout; **2**-error out; **&0&1&2** 表示相同地址

><: redirection(覆盖模式) >><(追加模式)

|: pipe，传输数据到指令

bash scrip:

第一行: #!/bin/bash

变量赋值 a=”114”; 使用变量: **$a; **后加特殊字符

$[#a]**//变量长度; **${a:pos:len}//子字符串;

${a/old/new}替换子字符串

$0**//文件名; **$n(n>0)//第 n 个参数; **$#**//传入参数数

$*//=”$1 $2 ... $n”; **$@**//=$*的括号内部分; $$进程 ID 号

$!**//后台第一个进程 ID; **$-//返回当前 shell option flags

$?//最后命令的推出状态, 0 为无错误

“`cat file` 111”，单引号内

if [] then elif then else fi;

for ? in ? do done; **while** [] do done;

数学操作: **let** arg [arg ...]

read var//创建变量并从 stdin 中读取

Condition:

str: [“$a”] a 不为空=[-n “$a”]=-[-z “$a”]; 比较符:==!=

\>\<//按字典序比较字符串; 在[[ ]]中时无需”\”转义

file: -e//exist; -f//is file; -d//为文件夹; -s//size>0;

-r//readable; -w-x 同理; **logic**:-a=&&, -o=||, !:Not

条件为**指令**，则当成功执行时为真

整数: [$a operation $b]; -eq==, ne!=, gt>, lt<, ge>=, le...

数学运算前加 **let**: **let** "b = $a + 9"

MakeFile:

$^**//所有依赖项(去重) **$<//第一个依赖项 **$@**//当前项

$+**//所有依赖项(保留重复); **$?所有比目标新的依赖项

$*//模式匹配的内容（用于模式规则）

```
a.o: a.cpp a.h
	g++ -c $^
main: a.o, b.o...
	g++ $^ -o main
clean:
```

```
rm -f *.o main
```

.PHONY: clean(以上为 dddsx 所写)

C++ File I/O

fstream

```cpp
#include <fstream>

int main() {
    std::fstream file; // 创建fstream对象
    file.open("filename", mode); // 打开文件
    // 进行文件操作
    file.close(); // 关闭文件
    return 0;
}
```

filename//文件名称; mode//打开文件的模式,

- `std::ios::**in**`：以输入模式打开文件。

- `std::ios::**out**`：以输出模式打开文件。

- `std::ios::**app**`：追加模式打开文件。

- `std::ios::**ate**`：打开文件并定位到文件末尾。

- `std::ios::**trunc**`：打开文件并截断文件，即清空文件内容

sstream

- `**istringstream**`：用于从字符串中读取数据。

- `**ostringstream**`：用于将数据写入字符串。

- `**stringstream**`：是上述两种组合，可以同时读取和写入

```cpp
std::string data = "10 20.5";
std::istringstream iss(data);

int i;
double d;

iss >> i >> d;

std::cout << "Integer: " << i << std::endl;
std::cout << "Double: " << d << std::endl;

return 0;
```

```cpp
std::ostringstream oss;
int i = 100;
double d = 200.5;

oss << i << " " << d;

std::string result = oss.str();
std::cout << "Resulting string: " << result << std::endl;

return 0;
```

```cpp
std::string data = "30 40.5";
std::stringstream ss(data);

int i;
double d;

// 从stringstream读取数据
ss >> i >> d;

std::cout << "Read Integer: " << i << ", Double: " << d << std::endl;

// 向stringstream写入数据
ss.str(""); // 清空stringstream
ss << "New data: " << 50 << " " << 60.7;

std::string newData = ss.str();
std::cout << "New data string: " << newData << std::endl;

return 0;
```

iomanip 库中的函数通常与 << 和 >> 操作符一起使用，以实现对输出流的控制。

函数/操纵符	功能	示例代码	输出结果
`std::setw(int n)`	设置字段宽度，为下一次输出指定宽度	`std::cout << std::setw(5) << 42;`	`42`
`std::fixed`	设置定点格式输出浮点数	`std::cout << std::fixed << std::setprecision(2) << 3.14159;`	`3.14`
`std::setfill(char)`	设置填充字符（默认是空格）	`std::cout << std::setfill('*') << std::setw(5) << 42;`	`***42`
`std::left`	设置左对齐	`std::cout << std::left << std::setw(5) << 42;`	`42`
`std::right`	设置右对齐	`std::cout << std::right << std::setw(5) << 42;`	`42`
`std::internal`	符号靠左，其余靠右	`std::cout << std::internal << std::setw(5) << -42;`	`- 42`
`std::setprecision(int)`	设置浮点数的有效位数	`std::cout << std::setprecision(3) << 3.14159;`	`3.14`
`std::scientific`	设置科学计数法格式输出浮点数	`std::cout << std::scientific << 3.14159;`	`3.141590e+00`
`std::hex`	设置整数以 16 进制显示	`std::cout << std::hex << 42;`	`2a`
`std::oct`	设置整数以 8 进制显示	`std::cout << std::oct << 42;`	`52`
`std::dec`	设置整数以 10 进制显示（默认）	`std::cout << std::dec << 42;`	`42`

Vector

Operation	Parameter	Effect	Time Complexity
`v.push_back(T t)`	An item `t` of type `T`	Inserts `t` at the end, increases size by 1.	Amortized O(1)
`v[i]`	Integer `i`	Accesses the object at position `i`.	O(1)O(1)
`v.size()`	None	Returns the number of items in `v`.	O(1)
`v.pop_back()`	None	Deletes the item at the end, decreases size by 1.	O(1)
`v1 = v2`	Two `vector` objects `v1` and `v2`	Sets `v1.size()` = `v2.size()`, and performs `v1[i]` = `v2[i]` for each `i = 0, 1, .., v2.size() - 1`.	O(n)O(n), where nn is the size of `v2`

List

Operation	Parameter	Effect	Time Complexity
`v.push_back(T t)`	An item `t` of type `T`	Inserts `t` at the end, increases size by 1.	Amortized O(1)
`v[i]`	Integer `i`	Accesses the object at position `i`.	O(1
`v.size()`	None	Returns the number of items in `v`.	O(1)
`v.pop_back()`	None	Deletes the item at the end, decreases size by 1.	O(1)
`v1 = v2`	Two `vector` objects `v1` and `v2`	Sets `v1.size()` = `v2.size()`, and performs `v1[i] = v2[i]` for each `i = 0, 1, .., v2.size() - 1`.	O(n), where nn is the size of `v2`

Map

Operation	Parameter	Effect	Time Complexity
`v.push_back(T t)`	An item `t` of type `T`	Inserts `t` at the end, increases size by 1.	Amortized O(1)
`v[i]`	Integer `i`	Accesses the object at position `i`.	O(1)
`v.size()`	None	Returns the number of items in `v`.	O(1)
`v.pop_back()`	None	Deletes the item at the end, decreases size by 1.	O(1)
`v1 = v2`	Two vector objects `v1` and `v2`	Sets `v1.size()` = `v2.size()`, and performs `v1[i] = v2[i]` for each `i = 0, 1, .., v2.size() - 1`.	O(n), where n is the size of `v2`

进阶用法

检查键是否存在:

```cpp
if (myMap.find(key) != myMap.end()) {
    // 键存在
}
```

清空 map:

```cpp
myMap.clear();
```

删除元素:

```cpp
myMap.erase(key);
```

获取 map 的大小:

```cpp
size_t size = myMap.size();
```