

Shell 其实就是一种脚本语言，也是一个可以用来连接内核和用户的软件，我们编写完源码后不用编译，直接运行源码即可。

bash, a general purpose scripting language.

command [arguments]

interactive and batch / 交互式和批处理

Gnuplot (在Linux下基于Qt的一套绘图工具)

https://blog.csdn.net/weixin\_42014622/article/details/82962634

LaTeX（LTEX，音译“拉泰赫”）是一种基于ΤΕΧ的排版系统

https://blog.csdn.net/pipisorry/article/details/54571521

关于shell 和 bash 之间的关系

<https://blog.csdn.net/weixin_42432281/article/details/88392219>

|  |  |  |
| --- | --- | --- |
| Platform | Linux | Windows |
| 字符界面模式 | shell | Command Prompt |
| 命令行 | bash | command |
| 批处理模式 | batch mode | 批处理 bat |

commands: aliases, functions, built-ins, keywords, and executables.

Text, letter

Description automatically generated

**Here document** 的使用总结

https://blog.csdn.net/liumiaocn/article/details/86715953

Sudo apt install / upgrade/update \*\*\*\*

Sudo apt-get install \*\*\*\*\*

确定型随机数是一个伪随机数，因为给定一相相同的种子seed，它就会生成相同的随机数，所以这种随机是假的，称为伪随机数。

如果想得到不同的随机序列，需要确保每次使用不同的种子，生成不同的序列。

Mersenne Twister算法译为马特赛特旋转演算法，是伪随机数发生器之一，其主要作用是生成伪随机数。

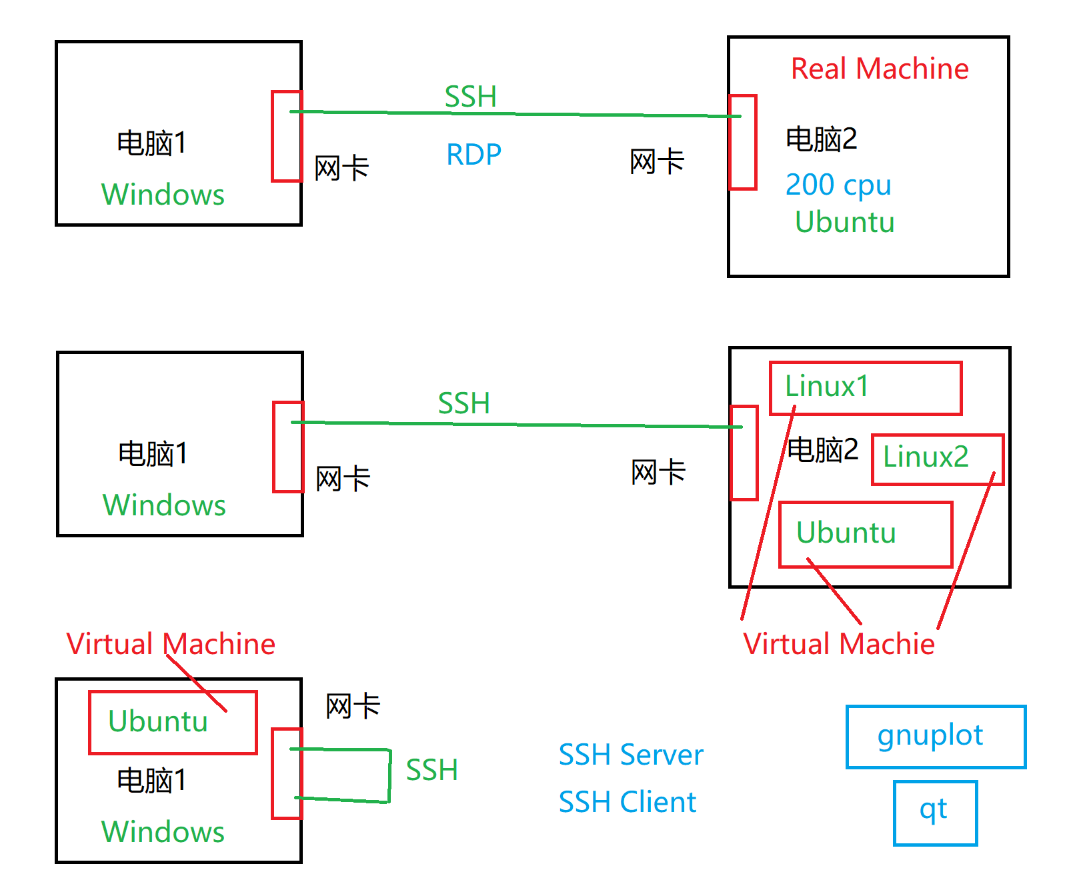
由于bash是不支持浮点型的计算，bash利用expr只能计算整数的基本运算。目前如果想要在bash中计算浮点型，只能通过bc或者awk来帮忙。

Echo $(expr $n1 + $n2)

Echo $(( $n1 + $n2 ))

Touch 创建文件

Vim/vi 文件，也可以创建文件



关于makefile和make:

<https://blog.csdn.net/ZBraveHeart/article/details/123187908>

Makefile文件中的命令有一定规范，一旦该文件编写好以后，在Linux命令行中执行一条make命令即可自动编译整个工程。

Text

Description automatically generated

它表示，为了达成目标，必须1，先有依赖，2，顺序执行下面的命令

Makefile并不会关心命令是如何执行的，仅仅只是会去执行所有定义的命令，和我们平时直接输入命令行是一样的效果。

1、目标即要生成的文件。如果目标文件的更新时间晚于依赖文件更新时间，则说明依赖文件没有改动，目标文件不需要重新编译。否则会进行重新编译并更新目标文件。

2、默认情况下Makefile的第一个目标为终极目标。

3、依赖：即目标文件由哪些文件生成。

4、命令：即通过执行命令由依赖文件生成目标文件。注意每条命令之前必须有一个tab保持缩进，这是语法要求（会有一些编辑工具默认tab为4个空格，会造成Makefile语法错误）。

5、all：Makefile文件默认只生成第一个目标文件即完成编译，但是我们可以通过all 指定所需要生成的目标文件。

**makefile中wildcard的理解**

<https://blog.csdn.net/dianqicyuyan/article/details/123509400>

wildcard 用来明确表示通配符。因为在 Makefile 里，变量实质上就是 C/C++ 中的宏，也就是说，如果一个表达式如 objs = \*.o ，则 objs 的值就是 \*.o ，而不是表示所有的 .o 文件。若果要使用通配符，那么就要使用 wildcard 来声明 \* 这个符号，使 \* 符号具有通配符的功能。

gnu make工具的[Makefile](https://so.csdn.net/so/search?q=Makefile&spm=1001.2101.3001.7020" \t "_blank)中$(SRC:%.c=%.o)的含义：**将SRC变量中所有以.c结尾的文件名替换成对应的以.o结尾的文件名，然后赋回给SRC。**

[makefile](https://so.csdn.net/so/search?q=makefile&spm=1001.2101.3001.7020)**中.PHONY的作用是什么？**

<https://blog.csdn.net/weixin_41969690/article/details/106727385>

.PHONY all clean起到“欺骗”的作用，欺骗make别管别的，让你干嘛你就干嘛的意思。

缺省情况下，makefile中的target是一个文件，上面的定义就是告诉make命令，后面的target不是文件，而只是一个label而已

Makefile有三个非常有用的变量。分别是$@，$^，$<代表的意义分别是：

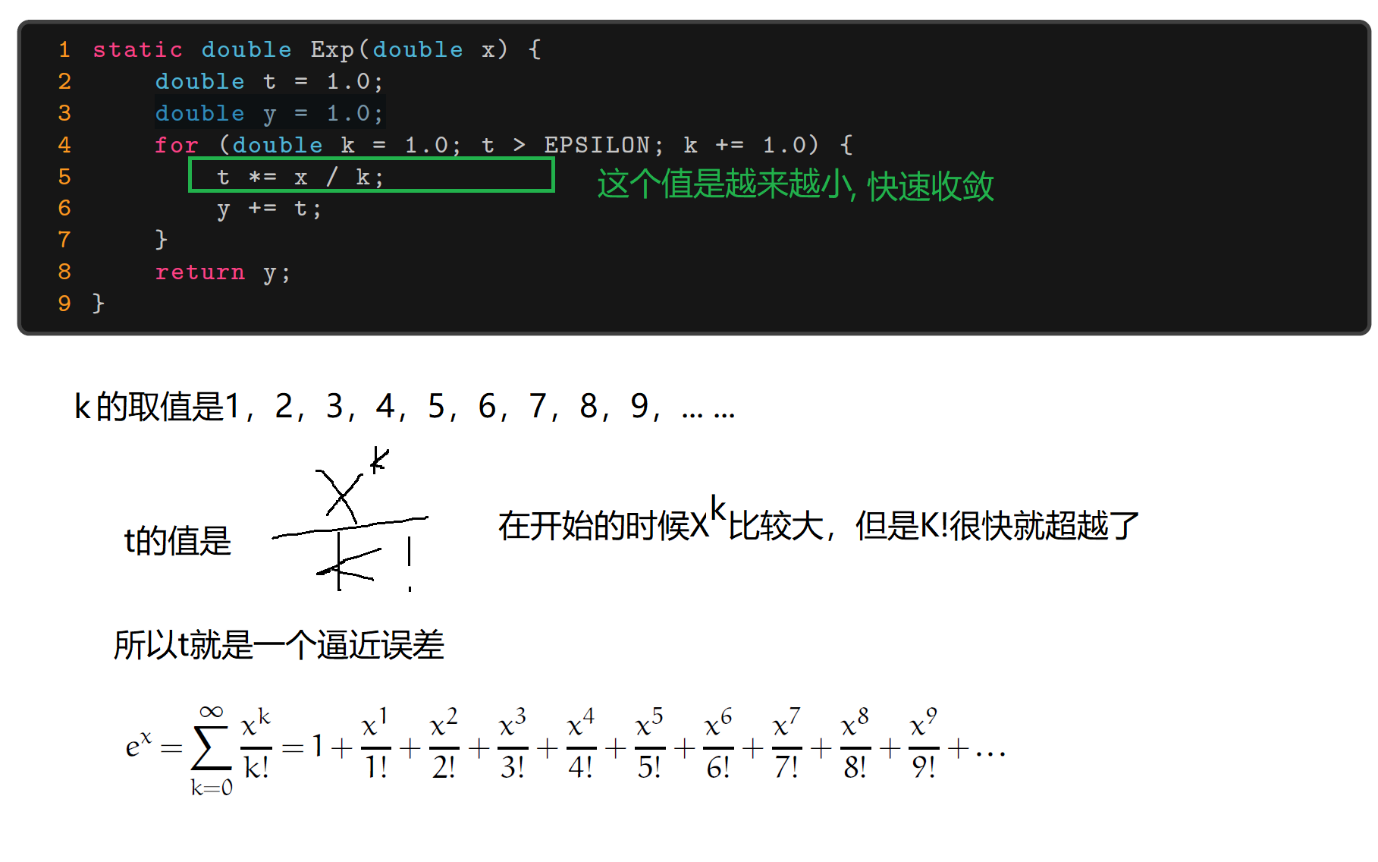
$@--目标文件，$^--所有的依赖文件，$<--第一个依赖文件。

mathlib.o : mathlib.c mathlib-test.c …

$< = mathlib.c

$@ = mathlib.o

$^ = mathlib.c mathlib-test.c …



关于LateX 排版系统

<https://blog.csdn.net/weixin_35805593/article/details/113454369>

documentclass{class} 为配置环境，可大致理解为C++中的 using namespace std;

usepackage{}为宏包调用，可大致理解为 #include<>等库函数的调用

正文区，以begin{document}打头，且有且只能有一个。这一点又与C++中的main函数如出一辙

latex运用%来做注释

主要用latex来输出数学公式

数学公式的输出有两种形式，一种为行内输出一种为行间输出。

分别用$ $ 与 $$ $$来表示。

latex的一个重大特点是“所见不是所得”，因此我们在命令行中另起一行，生成的pdf文件并不会换行，latex的换行指令为空一行另起。

{} 表示作为一个整体来统一处理

\[ \] 表示另起一行的公式，相当于 $$ $$

c\_m 2^m 表示下标和上标

软件人员必须了解的几个硬件知识

中央处理器CPU，内存Memory，硬盘Hard Disk

C语言必须了解的几个底层知识

内存空间，地址，指针，数组，数组长度/大小

位数：32位，可访问地址空间为2^32 = 4G，64位可访问地址空间为2^64，

18,446,744,073,709,551,616，理论上很大的一个值，但实际没有用到，足够程序运行

数据结构相关的几个知识

数组Array（可变数组），链表LinkedList（单向链表/双向链表），队列Queue，栈Stack，堆Heap，二叉树 Binary Tree，内存

对堆栈的误解

默认情况下，在 32 位上运行的线程的最大堆栈大小为 256 KB，64 位上的最大堆栈大小为 512 KB。可以通过系统函数进行修改。

关于程序设计的几点提示：（以asgn2为例子）

1. 读懂需求 （主要是老师提供的asgn2.pdf，特别是Your task部分，一定要读懂每一句的意思）
2. 列出文件，就是在Deliverables节中的内容

Mathlib.h 函数的申明，不能修改

Mathlib.c 在mathlib.h中的函数的实现

Mathlib-test.c main函数必须在这个文件，解析命令行，定义OPTIIONS

Makefile

README.md

DESIGN.pdf

WRITEUP.pdf

1. 根据文件的功能，先实现框架，保证能运行，可测试

比较明确的细节可以先用注释标出

4，再细化每一个文件的功能

细化的原则也是先简单，后细节

5，对复杂的问题进行拆解，分成多个函数，每个函数也是先框架，再细节

6，每细化一个函数，测试一个函数

测试时可以通过输出，断点查看

7，完成步骤2中的所有文件

有些工具必须先准备好，“工欲善其事，必先利其器”，“事半功倍”

比如Visual Studio，Windows和Ubuntu上文件的共享，Ubuntu上的可视化编辑器，等等

VS 2022 Professional key

共享文件夹，修改Ubuntu屏幕大小

<https://blog.csdn.net/Inochigohan/article/details/119791518>

作为CS专业，要善于利用网络资源，比如一些教学视频，在线的小工具

概率论和数理统计

<https://www.bilibili.com/video/BV1Pb411771f/?p=9&spm_id_from=pageDriver&vd_source=342115c1d2b02981c05a987047b37614>

求补码计算器

<http://www.99cankao.com/numbers/twos-complement.php>

二进制计算器

<https://miniwebtool.com/zh-cn/binary-calculator/?number1=0100&operate=1&number2=1010>

理论上的内容，最后用小程序进行验证等

比如Quiz上的题目

根据上面的提示应用到asgn3

1，读懂需求 （主要是老师提供的asgn3.pdf，特别是Your task部分，一定要读懂每一句的意思）

2，列出文件，就是在Deliverables节中的内容

dreidel.h 函数的申明

dreidel.c 在dreidel.h中的函数的实现

play-dreidel.c main函数必须在这个文件，解析命令行，定义OPTIIONS

mtrand.h 生成随机数的函数申明 不能修改

mtrand.c 生成随机数的函数实现 不能修改

Makefile

README.md

DESIGN.pdf

WRITEUP.pdf

3，根据文件的功能，先实现框架，保证能运行，可测试

比较明确的细节可以先用注释标出

4，再细化每一个文件的功能

细化的原则也是先简单，后细节

5，对复杂的问题进行拆解，分成多个函数，每个函数也是先框架，再细节

6，每细化一个函数，测试一个函数

测试时可以通过输出，断点查看

7，完成步骤2中的所有文件

Unix 文件系统，文件描述符（一个小的非负整数）

特别的，文件描述符0，1，2分别赋给stdin, stdout, stderr

Unix使用< 和 > 重定向和管道 | 改变输入/输出的IO

open 返回一个文件描述符

fopen 返回一个文件指针