# 2018.11.27（开始VIM）

### 关于Centos7无法打开vim测试题的问题解决：

如果遇到

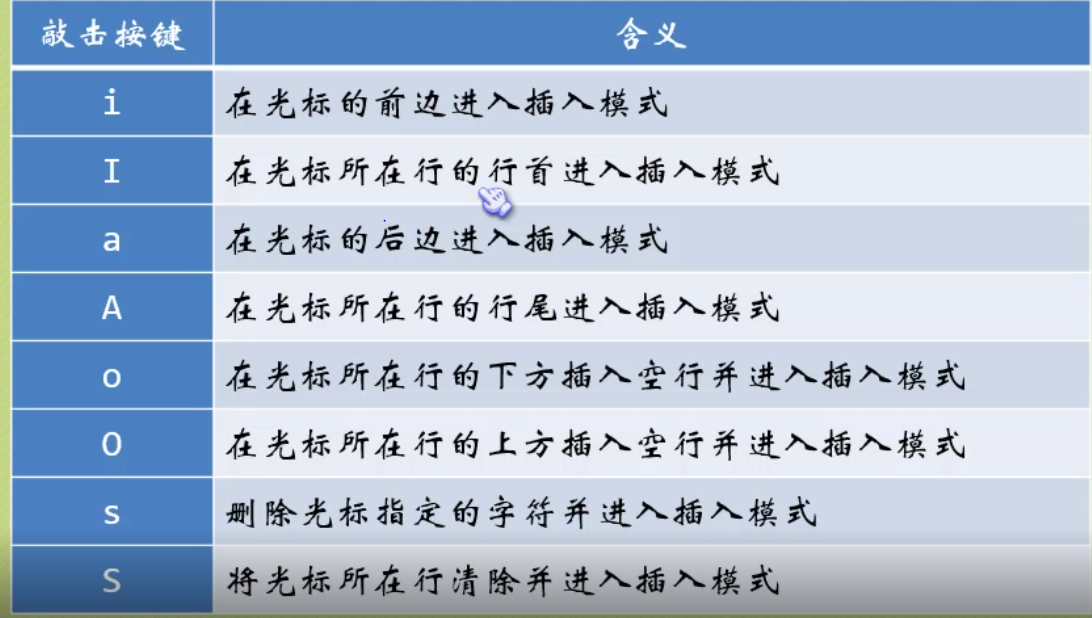


赋予文件执行权力：chmod 777 文件名

先安装32位库：yum install glibc.i686

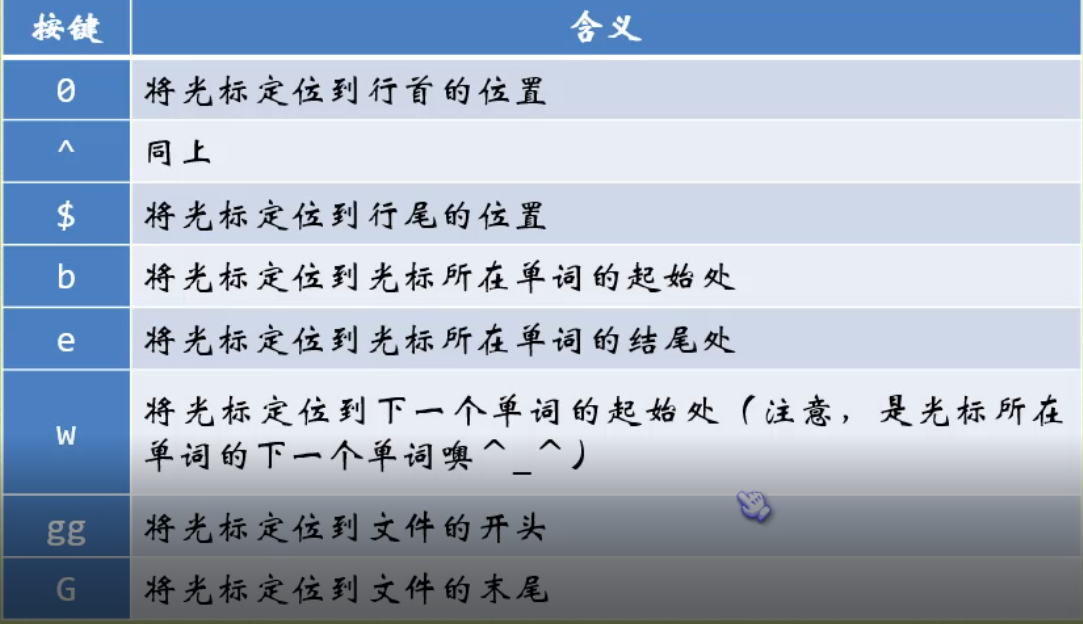
在安装依赖库：yum -y install libncursesw.so.5

#### 进入插入模式快捷键



#### 编辑文档快捷键

Motion按键如下图



换行常用的快捷键( 红色部分也属于motion按键 )：

**J – 下一行**

**K – 上一行**

**H – 左边**

**K – 右边**

0 - 到行首（不能自动换行连续）

$ - 到行尾（不能自动换行连续）

b / e – 到行头/到尾部 （能连续自动换行）

gg/G – 到开头/尾部

关于删除采用快捷键：

x-删除光标选中的文字

dd-删除本行

d可以与上方操作符拼车

比如db从光标位置到本行开头，其它类似

撤销采用快捷键：

u恢复是一个状态

U恢复本行状态

Ctrl+r – 撤回上一步操作

# 2018.11.30

#### 粘贴，拷贝，删除，替换，跳转行数

粘贴：

P – 光标前粘贴

p – 光标后粘贴

拷贝：

y +【数字】+ motion拷贝

删除：

d +【数字】+ motion删除( 其实d并不是真正的删除而是剪切 )

替换：

R进入替换模式,退格键可以还原替换的内容 , r替换光标选中的内容

c替换( 更加友好的替换方法 )

c +【数字】+ motion替换

跳转行数：

【数字】+ j/k 向下/上多少行

命令模式：【数字】，将直接跳转输入的行数

直接输入【数字】+ G直接跳转行数

#### 定位括号，快捷缩进

定位括号：

鼠标在括号上输入“%”会直接定位到下一个括号

快捷缩进：

【数字】 + 》/《

V进入可视模式，使用方向键可以选中内容，在使用《》进行缩进

#### 搜索

/ + 搜索内容( 从鼠标开始 – 向下搜索 )

? + 搜索内容( 从鼠标开始 – 向上搜索 )

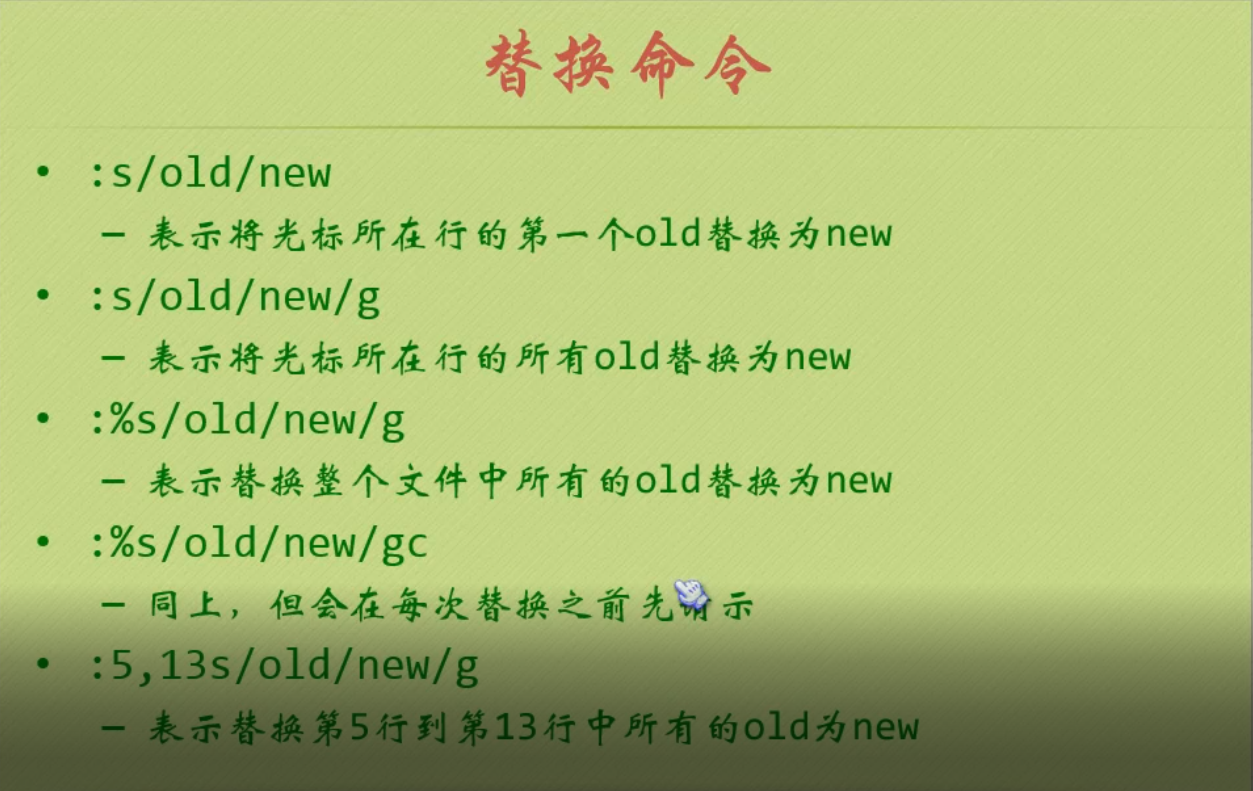
n – 下一个

N – 上一个

Nohl – 取消高亮

#### 替换

补充: “:1,10s/./--/g” 符号“.”为全部选择。



#### 查看文件信息

Ctrl + g 查看文件信息

#### 在vim中执行shell命令

:! + shell命令

例：: ! ls 直接显示当前列表

#### 文件另存为

: w 名字

# 2018.12.1（VIM完结）

#### 合并文件：

r + [合并的文件名称] – 在用w保存重名将覆盖原文件,不同名则创建新文件

#### 分屏显示打开多个文件

命令：

Vim -O 【文件名1】 【文件名2】小o横屏分屏显示，大O竖直分屏显示

切换光标所在文件：

CTRL+w w 到下一个文件

CTRL+w 方向键 到下一个文件

快捷退出：

：qa在打开多个文件时加入字母a表示全部退出

# 2018.12.7（VIM跨平台复制）

网上的方法大部分是废话，最后我在一个小角落里找到喽正确方法，原作者在chrome，c学习收藏夹里。

1. 安装“yum install vim-X11.x86\_64”( yum install vim”此时按下tab键可以查看安装列表” )
2. 进入一个文件 “vimx test.txt”
3. 随便复制一个东西，
4. 查看“：reg”查看复制的东西在那个寄存器里，有可能是+寄存器/\*寄存器
5. 假如是“\*寄存器，那么进入v浏览模式选择要复制的文本
6. 复制：依次输入“\*y ( 注意分号也要输入 )
7. 粘贴：在vim中shift+insert（外星人笔记本是：shift+FN+Delete因为键盘布局问题在fn的操作下delete相当于insert）
8. 在其它地方可以直接ctrl+v进行黏贴（因为\*寄存器为系统剪切板）

# 2018.12.10

#### 定义数据类型：

整数类型 - - - int a；

字符串类型 - - - char b；

浮点数类型 - - - float c；

双精度浮点数类型 - - - double d；

#### 格式化输出

#include <stdio.h>

int main(){

float a=3.15,b=2.2666,c=2.56;

printf("%d -- %d",a,b); //多个变量输入

printf("%.2f",a) // 保留2位数

printf(“%4.2f”,c) // 4相当于一共占4个位置,包括符号.以及小数。

printf("%010d",520000); // 空余部分用0补充

printf("%#x",getchar()); // “16进制显示0x开头”

return 0;

}

%-c --- 左对齐（默认为右对齐）

%c --- 字符串类型

%s --- 字符串类型

%d --- 十进制整数类型

%f --- 浮点数

%o --- 8进制（无前导0）

%x --- 16进制（无前导0x）

%p --- 指针值

%u --- 无符号类型变量值输出

%llu --- 输出unsigned long long int 类型数据

\b --- 相当于退格键

# 2018.12.11

#### 宏定义常量 （ 符号“#”为预处理符号 ）

#include <stdio.h>

#define URL "http://www.oo7.fun" // 宏定义，常量，常量的表示符都要为大写.

int main(){

printf("%s",URL);

return 0;

}

#### 关于字符串结束处理方法（底层原理）

一个字符串“abcd”，c语言自动给字符串末尾加入“\0”,系统检测到“\0”后结束字符串的读取。

# 2018.12.12

#### Sizeof测量变量大小以及，创建无符号变量

#include <stdio.h>

int main(){

int a;

short int b,c=-5; // signed默认有符号

unsigned short d=-5; // unsigned无符号 ，无符号就是没有正负号，无符号会使变量容量变大

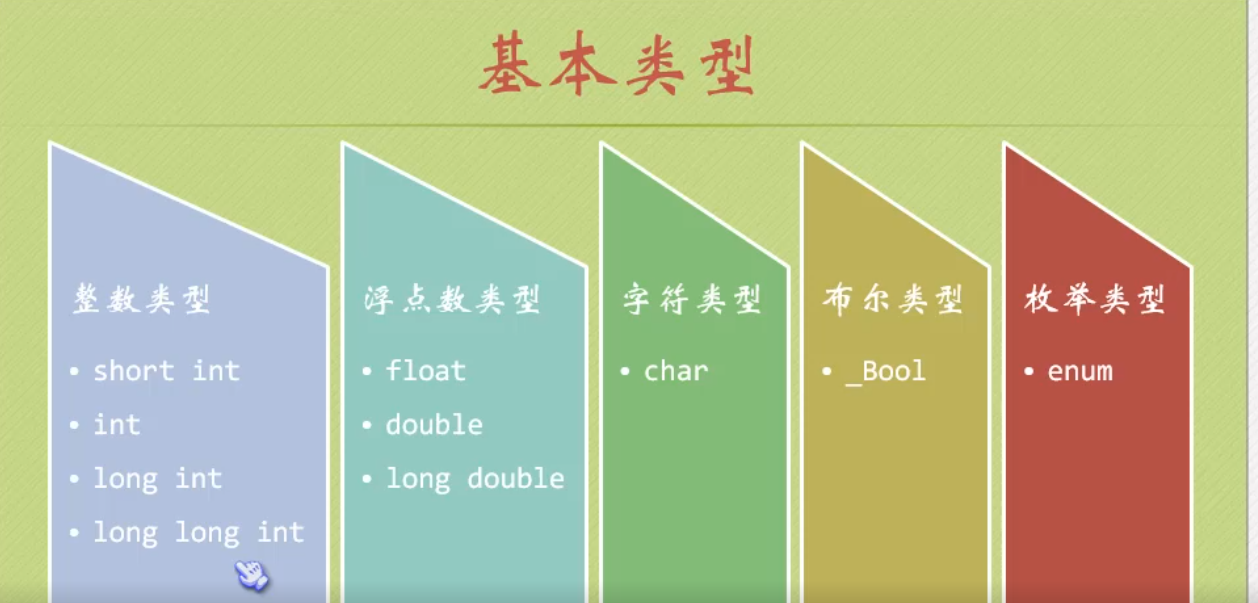
printf("%d,%d\n", sizeof(a),sizeof(b));

printf("%d,%d",c,d);

return 0;

}

#### 数据类型



# 2018.12.14

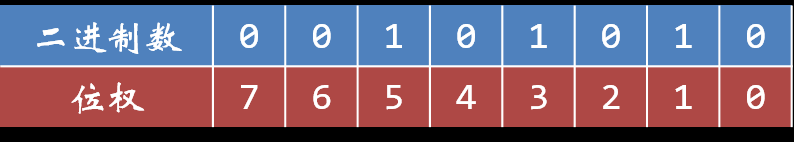
Cpu处理最小单位bit

内存处理最小单位Byte

8bit = 1Byte

### 进制转换

#### 二进制转十进制



位权法 - - - 二进制“无符号”或者为“正号”时

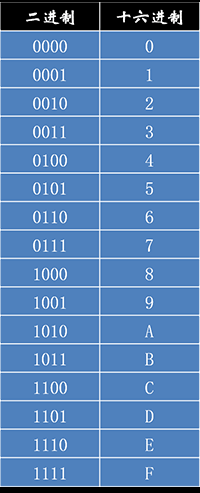
注意：如果为负号二进制则第七位乘-1

例如：有符号的二进制位为 1，表示这是一个负数，比如 1011 1100，那么符号位的权值就应该乘以 -1 得到：  
  
      -1 \* 2^7 + 0 \* 2^6 + 1 \* 2^5 + 1 \* 2^4 + 1 \* 2^3 + 1 \* 2^2 + 0 \* 2^1 + 0 \* 2^0  
  
== -128 + 32 + 16 + 8 + 4  
== -68

#### 二进制转十进制

#### 

### 16进制转2进制



#### 负数二进制表示



#### 有符号的二进制最大最小值



#### 各数据类型取值范围计算

无符号：如unsigned int 有4个字节 , 1Byte = 8bit , 则取值范围为0 ~2^32-1

有符号: 如int 有4字节,因为需要1bit表示正负号所以-2^31~+2^31-1



hh 表示将相应的参数按 signed char 或 unsigned char 类型输出

h 表示将相应的参数按 short 或 unsigned short 类型输出

l 表示将相应的参数按 long 或 unsigned long 类型输出

ll 表示将相应的参数按 long long 或 unsigned long long 类型输出

L 表示将相应的参数按 long double 类型输出

例如：Unsigned long long x; printf(“%llu”,x);

#### Scanf()输入函数

一句话,这scnaf（）函数和python的input（）相比，scanf（）函数就是个LG

#include <stdio.h>

int main(){

char c;

int i,j;

printf("输入: ");

scanf("%c%d",&c,&i );

printf("c = %c\n",c);

printf("i = %d\n",i);

return 0;

}

#### 特别强调！！！！scanf()细节

scanf("%d-%d-%d", &year1, &month1, &day1);

必须输入‘-’才可以如：2018-02-20

#### Scanf的跳过输入

Scnaf(“%d\*d%d”,a,b,c) // \*b不会被赋值

#### 字符串变量与输出与数据类型转换

#include <stdio.h>

int main(){

char test2[4] = {‘a’,’b’,’c’,’\n’}; // 字符串变量(方法2)

char test3[3] = {“abc”}; // 字符串变量(方法3)

char test[] = "asdasdasd"; // 字符串变量(方法1)

char \*name = “asdasd”; // 加个\*号一样可以使用（简洁的数组形式）

printf("%s%s\n",test,\*name); // 字符串输出用%s

printf("%d\n",1+(int)(1.6+23.123)); // 数据类型转换,注意转换格式

return 0;

}

# 2018.12.18

#### 输入字符串

#include <stdio.h>

int main(){

int a=666;

char i[5]; // 定义字符串定义变量( 不能超过5个元素 )

char \*ch; // 定义字符串( 方便，不需要设置接受的字符串数量，自动结尾加’\0’ )

scanf(“%s”,ch); // 直接接受字符串

printf("input: ");

scanf("%s%d",i,&a); // 字符串变量i,不需要加"&"

printf("%s: %d",i,a);

return 0;

}

#### Char字符串数据类型本质

C 语言中，最小的整数类型是？

答：char 类型

虽然 char 是字符类型，但它事实上是用于存放整数，因此属于整数类型。

下边摘录自 C99 原文：)JA89PI@d6.2.5 Types：

4. There are five standard signed integer types, designated as signed char, short int, int, long int, and long long int.

### Linux查看ascii表：

使用”man ascii”命令.

### C的严谨性（令人无语的计算,一定要注意） 例：9 / 5 \* 32 + 32 = 与 9 / 5.0 \* 32 + 32 完全是二种结果

因为9/5=1 ； 9/5.0 = 1.8

# 2018.12.23

8进制 – 以0开头

16进制 – 以0x开头

### 关于e

e前面必须有数字，后面必须是整数

### 关于%s - %d的一些细节

%5.3s 表示只输出字符串前3位,总长度为5

%3d 同上

### 字符串的一些函数

Putchar()单字符输出

Getchar()接受单个字符

### 快捷运算

Y=++x只能对整数类型计算

### 逻辑运算

短路现象：A && B 如果A为假直接输出0不会在测试B；相同的A || B如果A 为真则直接输出1不会在测试B

# 2018.12.24

### 优先级的小细节

1. (a=2)+3\*4; 相当于 a+3\*4

2 . 先后顺序(/\*% 优先级一样)+- 赋值=要比加减乘除都要低

3，逻辑！要比加减乘除优先级高

4． 赋值的计算是“自右向左”a=b=c=3 其实是c=3;b=c;a=b;

5. \*target++ != '\0' 相当于 \*( target++ ) != ‘\0’ ，因为++的优先级比\*取值符要高

# 2018.12.25

### 关于if

If其实有3种语句

If（条件）{代码区}else if(条件2){代码区}else{ 不满足条件/条件2时执行改语句 }

注意：if的“{}”大括号在只有单条语句时可以省略，但是不建议，养成良好的书写习惯。

#include <stdio.h> // if使用例子

int main(){

float fraction ;

printf("请输入分数: ");

scanf("%f",&fraction);

if (fraction >= 90 && fraction <= 100){

printf("A");

}

else if(fraction >= 80 && fraction < 90){ // 多条件判断

printf("B");

}

else if(fraction >= 60 && fraction <80){

printf("C");

}

else{

printf("D");

}

return 0;

}

## Steve Bourne大神编程方式

#include <stdio.h>

#define STRING char \*

#define IF if(

#define THEN ){

#define ELSE }else{

#define FI ;}

#define WHILE while(

#define DO ){

#define OD ;}

#define INT int

#define BEGIN {

#define END }

INT main() BEGIN

INT i;

printf("请输入你的年龄: ");

scanf("%d",&i);

IF i<18 THEN

printf("你未满18周岁,不得使用这个程序!\n");

ELSE

printf("你已满18周岁,欢迎使用本程序,嘻嘻嘻..\n");

END

return 0;

END

## 关于函数的使用方式

#include <stdio.h>

// 创建max函数接收2个值,返回大值

int max(int x , int y); // 声明函数,就算函数放在主函数后面也不会出错!( 注意接收参数的对应 )

void test();

void test(){ // void 类型函数无返回值,所以无return

printf("请输入2个值(用空格隔开!): ");

}

int max(int x , int y){

if(x > y){

return x;

}

else{

return y;

}

}

int main(){

int num\_1 , num\_2, result;

test();

scanf("%d%d",&num\_1,&num\_2);

result = max(num\_1,num\_2); // 接收max()函数返回值

printf("最大值为: %d",result);

return 0;

}

### 随机整数

#include <stdlib.h> // 含srand，rand函数

#include <time.h>

int getHeartRate(){

srand((unsigned int) (time(NULL))); // 以时间为随机种子的变换srand值,从而达到随机数字的目的

return rand()%11; // rand()函数生成随机数,%11代表0-10,( rand()函数受srand()函数影响 )

}

### 睡眠函数

千万要注意sleep()函数无法使用在循环中

#include <unistd.h> // 睡眠函数在这里

int main(){

printf("等待3秒...\n");

sleep(3); // 睡3秒( 暂停使用线程3秒,让给其它程序使用 )

return 0;

}

### 引入外部函数

1. #include "ssebtr1.h" // 读取当前目录文
2. 在文件ssebtr1.h 创建函数,但不能创建main函数,( 注意void类型函数无返回值 )
3. 在主文件直接输入相应函数名即可调用
4. 详细例子请看” 2018.12.25 ”SONY作品

# 2018.12.29

### 关于getchar()函数

Getchar()函数获取单个字符（注意是字符,不是字符串）

注意： getchar()函数接受的值格式只有字符格式

### switch

Case 目标值 : 执行语句; break; //如果没有break，将会执行后面的语句

Case后只能跟整数类型，且多个case标签值不能相同

switch( 接受值必须是整数类型 ){

case 10: case 20: printf("OK"); break; // case 后只能跟整数类型,多个条件可以多加case.

default: printf("END!"); break;

}

### 统计strlen()字符串/数组长度

小细节：注意strlen()返回的是unsigned int（ 无负值 ）

Strlen()函数类似于python的len()函数

#include <string.h> //保证strlen()函数能正常运行

Int main(){

Char \*ch=”asdasd”;

strlen(ch); // 检测长度,功能类似于python的len()函数

}

# 2018.12.30

### 关于getchar()无限循环单个字符处理

#include <stdio.h>

int main(){

char ch;

while( (ch = getchar()) != '\n'){ // getchar()无限循环单个字符处理

printf("%c",ch);

}

return 0;

}

### 关于sqrt（）函数

#include <math.h>

Int main(){

Sqrt(n) //相当于根号下的n，需要math.h头文件支持

Return0;

}

### 定义布尔类型变量

\_Bool ch;

### 关于int整数类型大小范围计算

其实小甲鱼在前几节课已经讲述，今天在回忆下

Int 大小为4个字节，有32位，因为需要一位来进行正负方便，所以范围为pow(2,(32-1));

### Atoi函数<stdlib.h>

默认功能是,将以数字开头的字符串转换为整数类型,遇到其它字符或者’\0’停止转换

#include <stdio.h>

#include <stdlib.h>

Int main(){

Char ch[]=”1212a32”;

Atoi(ch);

}

输出结果为1212

#### atoi()函数核心原理：

Int Value = 0;

根据字符串长度迭代循环

value = value \* 10 + 对应字符串 - '0'; //将数字字符转为对应的整形数

# 2018.12.31

### Const防止变量修改

#include <stdio.h>

int main(){

const char ch[100] = "aaaaaa"; // const防止修改变量,会将变量转为" 只读模式 ".

ch[3] = '1';　//　如果变量有const,会出现错误.

printf("%s",ch);

return 0;

}

### For循环语句细节

For循环可以和逗号表达式连用

For( a=0, b=10; a<b ; a ++, b-- ){}

在C中c99标准可以在for 循环创建局部变量( 出了for循环,定义的变量无用 )

如for(int a=0, b=10 ; a<b ; a++, b--){}

# 2018.1.5

#### 转换四舍五入的方法

result = (int)(result + 0.5) --- result == 5

### 关于++b于b++的区别

#include <stdio.h>

int main(){

int a=5,b=1;

a = ++b; // 先计算++b在给a进行赋值

printf("a=%d --- b=%d\n",a,b);

a=5;

b=1;

a = b++;　// 先赋值给a在进行b++计算

printf("a=%d --- b=%d",a,b);

return 0;

}

### 关于逗号计算符号

c = ( b = 2 , (c = b + 4) + 5 );

// 1. b = 2 逗号表达式优先执行,　左边语句

// 2. c = b+4

// 3. c + 5 得出结果

// 4. 输出c+5　因为逗号表达式“最终输出右边值”。

### 三目运算符

int a = 8;

a > 10 ? printf("OK") : printf("NO"); // 条件 ? 真 : 假;

### Goto语句,跳跃到标签执行

我感觉,比较实用,但是在真正的开发过程当中,尽量不要使用他,因为他容易造成逻辑混乱.

但是在黑客程序当中,可以使用,对一些反编译有好处

#include <stdio.h>

int main(){

int a = 8;

while(1){

goto A; // 跳越到标签A

}

Goto B;

A: printf("OK!");

printf("HELLOW");

return 0;

B: printf(“---”); //依然正常往下执行,不会有特殊的反应

}

### 生成“斐波那契数列数列”

for (i=0; i<10; i++){

printf("%lld, ",a);

// 斐波那契数列生成

n = a + b;

a = b;

b = n;

}

### 字符串转数字常量公式

Tir为位数

tir = tir\*10 + ch[pos] - '0'; // tir 决定位数

#### 提取整数类型任意位数

常量/10^tir%10 //tir决定位数

例：

123/(int)pow(10,0)%10 //提取个位数

# 2018.1.6

### 关于sprintf()函数转换字符串使用

Sprint()函数属于内置函数

#include <stdio.h>

Char a[];

Int b=1234;

sprintf(a,"%d",b); // 通过类似于printf格式化的形式转字符串

printf("%s",a); // 此时已经将,b的内容存入字符串变量,a中

### 生成随机数简洁版

注意此方法需要头文件<stdlib.h>与<time.h>头文件支持

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

Srand(time(0)); //生成随机数种子

Rand()%11 //生成0~11随机数字

Return 0;

### 简洁版颠倒整数（ 取位数算法 ）

#include <stdio.h>

int main(){

int target = 1234,invert=0;

while (target)

{

invert = invert \* 10 + target % 10; // 取位数公式

target = target / 10; // 消减数

}

printf("%d",invert);

return 0;

}

### 关于for与逗号连用的小细节

#include <stdio.h>

int main(){

int a,b;

for(a=0; a!=10,b!=10; a++){ //经验证: 是否循环条件由逗号右边决定

b=10;

printf("1");

}

}

#### 逗号运算

#include <stdio.h>

int main(){

int a,b,c;

b=(b=3,4); //结果b=4 逗号会将右边值赋给b

c = b++ + 5; //结果c=9 先赋值在进行+5

printf("c:%d b:%d",c,b);

}

### do….while语句

main(){

int a=0;

do{

a++;

printf("%d\n",a);

}while(a<10);

return 0;

}

### 数组

定义数组： 在C99标准中支持动态定义数组（int n; char a[ n ]; scanf(“%d”,&n);）

关于数组访问越界的问题：按照c语言的标准是不可以的，但是不同的编译器，可能会出现不同的结果，比如gcc编译器有访问保护，如果访问超越范围的数组，gcc编译器会根据内容值给超越的值合理的赋值，不过尽量不要干另类的事情。

#include <stdio.h>

int main(){

int a[5]={0},i=0; // 数组初始化为0

int b[5]={ 12,3,213,123 }; // 填充数字,没有填充位置为0

int c[5]={12}; // 同样,没有填充的位置为0

int d[]={123,12,3,123}; // 编译器会自动,填入数组大小

int e[5]={[0]=666,[4]=111}; // 可以指定位置进行赋值操作

for(i=0;i<5;i++){ // 迭代数组

printf("a %d\n",a[i]);

}

return 0;

}

### 判断闰年算法

// 判断闰年: 4年一闰,百年不闰,400年一闰

if( (year%4 == 0 && year%100 != 0) || (year%400 == 0) ){

printf("%d是闰年\n",year);

}

else{

printf("%d不是闰年\n");

}

# 2019.1.7

### 如何使用自定义函数返回一个数组

#include <stdio.h>

// 返回数组

int \*test(){ // 注意创建指针形式

static int a[2]={2,5}; // static改为静态存储变量（如果没有编译器会提示警告，static博大精深有待研究）

return a;

}

int main(){

int \*p;

p = test(); // 使用指针接收函数传来的数组

printf("%d",p[0]);

return 0;

}

### 特别强调！！！！SCANF()细节

scanf("%d-%d-%d", &year1, &month1, &day1);

必须输入‘-’才可以如：2018-02-20

Scanf() 如果在输入的过程中遇到“空格”会停止，然后进行下一行的代码执行

# 2019.1.8（C语言之路，理解指针c语言将不在神秘）

## 参考路线

1. 理解指针C语言将不在神秘

2. C语言在Unix系统上有更加广阔的空间，有丰富的源码库提供参考

3. 入门：由Kernighan和Ritchie编写的《The C Programming Language》中文名《C程序设计语言》

4. 辅助：《C缺陷与陷阱》

5. 辅助：一本《C语言参考手册》

6. 进阶：《C和指针》英文名《Pointers on C》

7. 精通：精通C语言的指标，能透彻理解掌握这本书《C专家编程》英文名称《Expert C Programming》

## 我的路线

1. 入门：小甲鱼《带你学C带你飞》视频+课后题（课后题必做）
2. 入门：小甲鱼《数据结构与算法》视频+课后题（课后题必做）
3. 进阶：彻底理解掌握指针《C和指针》英文名《Pointers on C》
4. 精通：透彻理解掌握《C专家编程》英文名称《Expert C Programming》

# 2019.1.10

### 字符串复制函数

#include <stdio.h>

#include <string.h>

int main(){

char str[]="hello";

char str2[]="how are you";

char str3[] = "welcometo";

char str4[] = "OK";

// 普通复制strcpy()函数（注意大小，不能溢出）

strcpy( str2,str ); // str内容复制到str2里( 注意str2长度>str才可以,否则会出现溢出错误 )

printf("%s\n",str2);

// 防溢出复制方法strncpy()函数（注意末尾加入‘\0’）

strncpy( str4,str3,strlen(str3) ); // strncpy( a,b,c )将b的内容复制到a,在由c决定复制几个字符.

str4[ strlen(str3) ] = '\0'; // 因为strncpy函数不主动加入'\0'所以要手动在末尾加入.

printf("%s\n",str4);

printf("%s",str3);

return 0;

}

### 关于strcat()拼接函数与strcmp()比较函数

#include <stdio.h>

#include <string.h>

int main(){

char a[]="hello ";

char b[]="word";

char c[]="233";

char d[]="233";

char e[]="233haha";

strcat(a,b); // strcat()把b拼接到a里

printf("%s\n",a);

strncat(a,c,1); // strncat()指定拼接

printf("%s\n",a);

printf("%d\n",strcmp( c,d )); // strcmp()如果二个相同返回0

printf("%d\n",strcmp( b,c )); // 如果不相等则返回非0值

printf("%d",strncmp( d,e,3 )); // strncmp()指定比较,如果相等返回0

return 0;

}

#### Strncat()的小细节

Char a[10];

Strncat(a,”abc”,10);

这样定义会出现乱码，因为a变量没有任何值,在使用strncat()进行指定拼接会出现乱码

Char a[10]=””;这样即可避免，目前只发现strncat()函数存在这种问题（我怀疑这是个BUG）

# 2019.1.11

### 关于sizeof()的小细节

sizeof 运算符是取得字符串的尺寸，即该字符串所处存储空间的大小。

Char a[10];

Printf(“%d”,sizeof(a)); // 大小会显示a定义的数大小

Char b[]=”abc”;

Printf(“%d”,sizeof(b)); // 此时sizeof(b)为4，abc’\0’,他会自动分析大小长度

#### Sizeof 在If语句中计算方法

char str1[] = "love";

char str2[] = "FishC";

int a;

if ((a = strlen(str1) - strlen(str2) ) < 0) // 必须个赋值才能比较

{

printf("字符串%s比字符串%s短！\n", str1, str2);

}

# 2019.1.12

### 二维数组

注意数组最忌讳的就是“越界访问”

二位数组在内存中以线性方式存储

#include <stdio.h>

int main(){

int a[3][4] = { 0,1,2,3,4,5,6,7,8,9,10,11}; // 第一种写法

int b[3][4] = { {0,1,2,3},{4,5,6,7},{8,9,10,11}}; // 第二种写法

int b[3][4] = { // 第三种写法( 推荐 )

{0,1,2,3},

{4,5,6,7},

{8,9,10,11}

};

// 数组初始化

a[3][4] = {0};

// 可以直接在括号内赋值

a[3][4] = { [0][0] = 1,[1][1]=2 };

// 注意括号内一个括号代表一行

a[3][4] = { {0},{1},{2} };

}

#### 二位数组输出字符快捷方法

char slogans[5][100]={

"1asdasdasdasdasdsa",

"2resadasdtsaedasdr",

"3sadasdasd",

"4asdasasddasd",

"5sads",

};

For(i=0; i<5; i++){

Printf(“%s”,slogans[i]); // 通过迭代循环x方向%s可以直接把一行字符输出

}

# 2019.1.13

## 指针

实战出真知：注意&取内存地址符与\*取数据值符，指针其实间接访问变量存储的数据（它的大小为4存储单元，因为它只保存的是一个16进制的内存地址，这个内存地址是指针指向的变量地址，在由变量存储的内存地址导向数据内存地址）

注意：指针只能存放地址

#include <stdio.h>

int main(){

char a='S';

int b=10;

// 控制指针指向变量内存地址( &符号 取地址运算符 )

char \*ch = &a;

int \*in = &b;

// 指针其实就是间接访问数据: 指针 - 指向变量内存地址(&符号) - 变量指向存储数据的内存地址 - 访问到数据(\*符号)

printf("内存地址导引(指针): ch(a) = %c,in(b) = %d\n",\*ch,\*in); // ( \*符号 在这里为取值运算符, \*符号 在定义类型中为创建指针 )

printf("原变量: a = %c,b = %d\n",a,b);

// 通过指针改变变量存储的值( 上放已经确定指向的变量 )

\*ch = 'A';

\*in = 666;

printf("a=%c,b=%d",a,b);

return 0;

}

#### 野指针( 禁止使用 )

不要使用也指针，虽然它语法上是正确的

#include <stdio.h>

int main(){

// 野指针( 语法上是正确的,但是这种写法是错误的,原因: 是指针在没有指方向时,会随机分配一个内存存储地址,这有可能覆盖到其它进程,导致莫名其妙的错误和BUG )

char \*ch;

\*ch = 'a';

return 0;

}

#### 查看指针保存的地址类型数据

我们找到指针是保存的一个内存地址，那么如何看到它呢？s

#include <stdio.h>

int main(){

char a='c';

char \*ch = &a;

// 查看指针保存的地址

printf("%p",ch); // %p 显示地址类型数据

return 0;

}

### 关于lvalue

**C 语言的术语 lvalue 指用于识别或定位一个存储位置的标识符。（注意：左值同时还必须是可改变的）**

至于rvalue：非lvalue即是rvalue

个人理解lvalue为变量，因为变量是可变值

# 2019.1.14

### 指针与数组

#include <stdio.h>

#include <string.h>

int main(){

// 数组与指针是一对好基友

// 核心本质

// 1. 数组是数据排序列表,各存储单元紧挨着( 数组名,其实是个地址信息,所以在使用scanf函数时不需要&地址符 )

// 2. 指针默认直接指向数组开头地址,借此来控制数组

int a[5]={1,2,3,4,5};

int \*p;

char \*str = "asdasd";

p = a;

printf("%p,%p\n",p,&a[0]); // 指针指向数组,默认为指向数组开头

printf("%d,%d\n",\*p,\*(p+1)); // 通过指针访问数组下一个元素

printf("%d,%d\n",p[0],p[1]); // 第二种访问方法

printf("%p,%p",a,&a[0]); // 经测试,数组名确实为内存地址,而且为数组开头内存地址

// 基友情

printf("%s,%d\n",str,strlen(str)); // 甚至不用去指定数组就可以直接使用( 这个属于野指针,尽量不要使用 )

printf("%d,%d",\*a,\*(a+1)); // a在没有定义指针的情况下,也能通过指针的方式索引数组值

return 0;

}

#### 关于scanf（）函数中指针为什么不用&符

因为指针本身存储的就是内存地址，所以不需要&地址符来引导

#include <stdio.h>

int main(){

int a;

int \*p = &a; // 取出a的内存地址,赋值给指针

scanf("%d",p); // 因为指针p本身保存的内存地址,所以不需要&地址符

printf("%d",a);

return 0;

}

# 2019.1.15

关于fgets()函数使用，只支持字符数组输入、

#include <stdio.h>

#define MAX 1024

int main(){

char a[MAX];

printf("input: ");

// fgets(数组,数量,输入流) 只支持字符输入

fgets(a,MAX,stdin); // stdin从键盘中获得输入流( 默认保存换行符 )

printf("%s,%d",a,strlen(a));

return 0;

}

### 关于NULL的功能

Int \*q;

\*q = NULL; // 指针被NULL赋值后，逻辑地址被赋值为00x000地址，但这个地址，并不存在，等同于指针q消失在内存中，所以一切关于于指针q赋值操作的，都会报错。

### 关于#define的小细节

#define 的“本质”是: 检测到定义的数据转换成要变成的数据，比如：

#define N 2

#define M N+1

#define NUM 2\*M+1 ( 相当于: 2\*N+1+1 = 2\*2+1+1 = 6 )

故：NUM 为 6

如果#define M (N+1)

那么#define NUM 2\*M+1 ( 相当于: 2( N+1 ) + 1 = 7 )

此时NUM 为 7

# 2019.1.16

## 指定中文传递算法，一定要记住（太烧脑啦！研究半天的结果）：2019.1.15/T2v2.c

### 关于int类型内存地址相加

p 存放的地址值是 0x11008888，那么请问 p + 1，p + 2，p + 4 和 p + 8 的地址分别是？

P + n == 0x11008888 + n\*4 == 结果内存地址

p + 1 == 0x11008888 + 4 == 0x1100888C

p + 2 == 0x11008888 + 8 == 0x11008890

p + 4 == 0x11008888 + 16 == 0x11008898

p + 8 == 0x11008888 + 32 == 0x110088A8

### 关于索引数组几种方法

注意:数组变量名编译器理解为数组开头地址

请问 str[20] 是否可以写成 20[str]？

答：可以。Powered by bbs.fishc.com

因为在访问数组的元素的时候，数组名被解释为数组第一个元素的地址。（ 相当于: 开头内存地址+数字\*n ( n为不同类型数组的存储单元)，所以加减不在乎先前先后 ）

所以 str[20] == \*(str + 20) == \*(20 + str) == 20[str]

#### 关于指针赋值问题

Int a[10]={0};

Int \*p;

P = a;

\*p = 1; 相当于 p[0] = 1;

# 2019.1.19

### 字符反转算法

// 反转字符串算法;n为开头位置,i为结尾位置

while( n<i ){

ch = b[n];

b[n] = b[i];

b[i] = ch;

n++;

i--;

}

### 取整数类型位数算法

Target = 1024;

// 这里我要取出2

Target = Target/10; // 通过除，确定取的位置

Target%10; // 取出最后一位

// 取出0

Target = Target/100;

Target%10

## 数组指针

#include <stdio.h>

int main(){

int i;

int temp[5]={1,2,3,4,5};

int (\*p2)[5] = &temp;// 先创建一个指针,在将一个数组的地址赋给指针,叫"数组指针"

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

printf("%d\n",\*( \*p2 + i )); // 索引复杂,这东西真的有用?

}

return 0;

}

## 指针数组

#include <stdio.h>

int main(){

char \*p[5]={ // 数组每一个元素为一个内存地址( []括号比\*优先级高 )

"1234",

"22222",

"adsad"

};

int i;

for(i=0;i<3;i++){

printf("%s\n",p[i]);

}

return 0;

}

### Main函数传递参数标准方法

#include <stdio.h>

// 传递参数要求

// C语言规定 main 函数的参数只能有两个，习惯上把这两个参数写为 argc 和 argv

// argc必须为整数类型变量, argv必须为指向字符的指针( 不是指针数组 )

// 功能：

// argc : 统计接受的参数个数，比实际情况大1

// \*argv[] : 字符串的形式保存接受的参数

int main( int argc,char \*argv[] ){ // main()函数只能接收这2个

int i;

for(i=0;i<argc;i++){

printf("%s\n",argv[i]);

}

return 0;

}

### 关于数组地址中&array与array区别

核心:

array开头地址,array+1为第二元素地址

&array为整个数组地址（ 但是它的值为开头地址,一定要注意它不等于array ）,

&array获取末尾位置可以：&array+1为数组最后位置（ 指针指向整体数组最后一个位置偏移1个单位 ）

(int \*)(&array + 1）强制转换整数类型数组，指针指的位置相当于数组的最后一位+1

例如：

int array[10] = { 0,1,2,3,4,5,6,7,8,9 }

// &array表示整个数组的地址,所以(&array+1)表示的整个数组地址最后位置,

// (int\*)强制转换后为array[10]的位置( 第11个元素 ), \*(p-6)相当于array[10-6] == array[4]

#### 关于&array具体官方解释

7. 你如何理解“指针的类型决定了指针的视野，指针的视野决定了指针的步长”这句话？

答：你必须清楚一个指针变量将告诉我们两个信息：某个数据结构的起始地址，以及该结构的跨度。比如 int p = &a; 说明该指针指向变量 a 的起始地址，以及它的跨度为 sizeof(int)。所以 p + 1 == p + sizeof(int)。

怕大家不够清晰，我接着给大家讲上一题（第 4 题）……同样的道理，int (\*p)[10] 虽然是定义一个整型指针，但不要忘了它后边还有一个数组，所以它的跨度应该是 sizeof(int) \* 10，而 array 作为数组名，它的含义是“指向数组第一个元素的地址”，所以 array 的跨度是 sizeof(array[0])，因此编译系统遇到 int (\*p)[10] = array; 这样的定义就会果断报错：“左右类型不一致”。

追问：那 &array 怎么可以？?W'krJI}\_]xCvtTuU<RYM"`4|a9b.i

追答：天啊，快点忘掉 &array 的值等于 array 这一回事儿吧（你会这么想完全是因为这个）……因为 &array 是整个数组的地址，所以它的跨度自然而然就是整个数组啦。

备注：此句名言来自 ryxcaixia 版主（一位潜伏在鱼 C，默默蹲守迷茫可爱的少女的热心版主……）

#### 实验：

#include <stdio.h>

int main(){

char \*array[2] = {"abc","wdw"};

char \*(\*p)[2] = &array;

printf("%d\n",sizeof(array[0])); // 加上2个'\0'确实为8

printf("%d\n",sizeof(\*p)); // 相当于8\*2 = 16

return 0;

}

### 关于如何理解\*(\*p)[5]

#include <stdio.h>

int main(){

char \*array[3]={"aaaaa","bbbbb","ccccc"};

char \*(\*p)[3] = &array; // 获取整个数组地址

printf("%s",(\*p)[1] );

return 0;

}

具上方我们已经知道指针角度问题

所以如果把 \*( \*p )[3] 看成 \*p[3]

调用：

输出一行字符串: p[1] , (\*p)[1]

输出一个字符： p[1][0], (\*p)[1][0]

所以” (\*p) ”是必须要写的

### 二维数组索引各种姿势

#include <stdio.h>

int main(){

char array[5][5];

int i,j,s=0;

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

for(j=0;j<5;j++){

array[i][j]=s++;

}

}

// 核心: \*( array + i ) == array[i]

printf("%p\n",\*(array+1));

printf("%p\n",array[1]);

// 核心2: \*(\*(array+i) + j) == array[i][j]

printf("%p\n",&\*(\*(array+1)+1));

printf("%p",&array[1][1]);

return 0;

}

#### 数组指针与二维数组

#include <stdio.h>

int main(){

int array[2][3] = { {1,2,3},{4,5,6} };

int (\*p)[3] = array;

// (\*p)表示开头, (\*p) + 1内存位置后移1位, (\*p + 1)内存位置前往末尾

printf("%d\n",\*\*(p+1));

printf("%d\n",array[1][0]);

printf("%d\n", \*((\*p +1 ) +2 ) );

printf("%d\n", array[1][2]);

return 0;

}

#### 关于二维数组的中介指针为什么这么写(\*p)[3]？

因为优先级问题，加入没有小括号，\*p[3] 与 (\*p)[3] 意思是完全不同的

\*p[3] 为指针数组（ 数组里全部为指针类型 ）

( \*p )[3] 为数组指针（ 指针默认变量名指向开头地址，其它为正常的数组类型 ）

记住一大核心：指针只能存储地址形式

## 关于数组使用指针法索引的二大核心

// 核心1 \*(array + 1) array[1];

// 核心2 \*(array)+1 array[0][1];

翻译:

这里假设array为一个三维数组

\*(\*(\*array+1)+2) --- array[0][1][2]

\*(\*(\*(array+1)+1)+2) --- array[1][1][2]

\*\*\*array --- array[0][0][0]

\*(\*\*array + 1) --- array[0][0][1]

### 关于如何使用一维数组强制转换二维数组正方形矩阵

#include <stdio.h>

int main()

{

int array[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

int (\*p)[3] = (int(\*)[3])&array; // 转换整数类型指针，分成3段（ 具体底层原理目前难以理解 ）

// 一定注意，前后值必须相等否则会报错

printf("%d\n", p[2][2] );

return 0;

}

# 2019.1.24

### 正方形矩阵长度算法

Len = sqrt( strlen( p ) ) // 直接根号长度就可以

### 关于void(无类型指针)类型指针

使用法则：( 简单的说，只能存储内存地址，不能被解引用 )

1. void 可以直接赋任何类型的数据指针，因为它只保存内存地址
2. void 为无类型,所以没有长度,即使赋完值，只保存内存地址
3. 规范输出空指针时,先强制转换指针类型
4. Void只保存内存地址，进行四则运算也是内存地址的计算
5. 可以把void理解为内存地址存储间接者

void \*pv;

(int \*)pv; // 即可将pv空指针转换为整数类型指针

1. 通过void的强制转换,我们可以间接的转换不同类型的指针类型内容( 无路可走在>使用,一定要注释 )

#### void可以接受的不同种类的指针类型

#include <stdio.h>

int main(){

int num = 1025;

int \*pn = &num;

char \*ps = "abcd";

void \*pv;

pv = pn // 接受整数类型指针;

printf("%p - %p\n",pv,pn);

pv = ps; // 接受字符类型指针

printf("%p - %p",pv,ps);

return 0;

}

### 关于NULL的使用

1. NULL的原理是: #define NULL (( void \*)0)
2. NULL可用于初始化指针( 使指针指向一个不使用的地址,防止成为野指针 )
3. 小甲鱼老师说很重要,将来写大型程序能节约大量的调试时间

#### 使用NULL防止出现野指针

#include <stdio.h>

int main(){

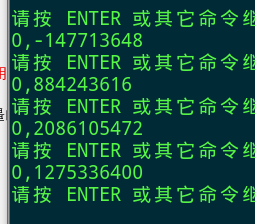
int \*p = NULL; // 非野指针（空闲内存区域）

int \*p2; // 野指针（随机内存地址）

printf("%d,%d",p,p2);

return 0;

}



# 2019.1.25

### 关于创建可变化数组小细节

Int x=10,y=10;

Int num[x][y]; // xy一定要有初始值,且初始值要“大于”改变值，否则会出现bug

// 这样在改变数组大小就不会出现bug（ 注意： 不要超过初始值 ）  
x = 4;

y = 5;

# 2019.1.26

### 利用sizeof( void \* )进行判断多少位平台

Sizeof( void \* ) == 2 - - - 16位平台

Sizeof( void \* ) == 4 - - - 32 位平台

Sizeof( void \* ) == 8 - - - 64 位平台

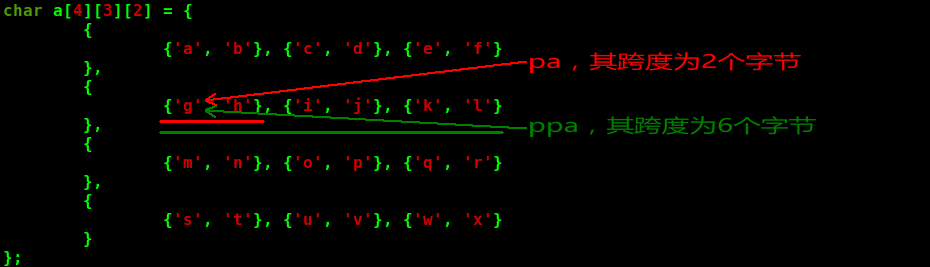
# 2019.2.14

## 非常重要的指针题目

1. char a[4][3][2] = {
2. {
3. {'a', 'b'}, {'c', 'd'}, {'e', 'f'}
4. },
5. {
6. {'g', 'h'}, {'i', 'j'}, {'k', 'l'}
7. },
8. {
9. {'m', 'n'}, {'o', 'p'}, {'q', 'r'}
10. },
11. {
12. {'s', 't'}, {'u', 'v'}, {'w', 'x'}
13. }
14. };
15. char (\*pa)[2] = &a[1][0];
16. char (\*ppa)[3][2] = &a[1];

问：想要通过指针 pa 和 ppa 访问数组 a 中的 'x' 元素，请问表达式应该怎么写？

答：\*(\*(pa+8)+1) 和 \*(\*(\*(ppa+2)+2)+1))。

原因：

# 2019.2.15

### 指针常量

#include <stdio.h>

int main(){

int num=233;

int num2 = 222;

const int \*p = &num; // 可修改指向,但不能修改指向的值( 如果没有const默认是可以通过解引用修改指向常量值 )

int \* const s = &num; // 不可以改变指向,但是可以通过解引用改变指向的值( 与上方相反,此方法叫"指针常量" )

const int \* const p = &num; // 二者都不可修改

printf("%d\n",\*p);

p = &num2;

printf("%d\n",\*p);

printf("%d\n",\*s);

\*s = 333;

printf("%d",\*s);

s = &num2;

return 0;

}

#### 关于const 小细节

为什么下方代码中，a变量值依然能被修改？

Const int a = 22; // 因为const只限制不能通过变量a来修改值，但是没有限制通过“指针解引用”来修改值

Int \*p = &a;

\*p = 33;

# 2019.2.16

### 判断大端小端

二者区别在与逻辑地址存储形式不同，具体参考：<https://fishc.com.cn/thread-75394-1-1.html>

#include <stdio.h>

int main(){

int a = 0x12345678;

char \*b = (char \*)&a; // 将索引的地址转换为char类型,方便char类型指针进行指向

if( \*b == 0x78 ){

printf("你的机器采用小端字节序.\n");

printf("%p 在内存中以此存放为: %p %p %p %p\n",a,\*b,\*(b+1),\*(b+2),\*(b+3)); // 注意显示逻辑地址

// 第二种索引方法 printf("0x12345678 在内存中依次存放为：0x%x 0x%x 0x%x 0x%x\n", p[0], p[1], p[2], p[3]);

}

else{

printf("你的机器采用大端");

}

return 0;

}

### 强制转换变量类型

Void \*p;

\*(int \*)p; // 将 p 先强制转换成 int \* 类型