1. Linux系统简介

1. Linux操作系统的典型使用方式是通过终端软件（终端窗口）

---》终端窗口里只能使用键盘不能使用鼠标

2. 操作系统中专门用来管理文件的部分叫文件系统

---》根目录可以使用/表示

3. 绝对路径和相对路径

---》绝对路径的起点必须是根目录（以/作为开头，类似Windows中，c:/）

---》相对路径中用..表示父目录；用.表示当前目录

------------------------------------

4. 部分命令介绍

（1）clear、pwd、cd、mv、cp、ls、rm、mkdir、cat

（2）touch：创建普通文件（或修改已存在文件的最后修改时间）

（3）echo： 重定向命令，举例：echo 12345 > 文件路径

5. 终端窗口操作技巧：

---》Tab键妙用：按一次，补全未输入完整的函数名/命令/文件路径；按两次，显示更多可能补全输入内容的选项；

注意：Linux系统中，部分函数和命令同名，并且功能相同

=====================================

6. vi编辑器的使用

（1）vi编辑器的使用技巧

---》正常模式下：

拷贝：nyy命令 - 拷贝光标所在行开始的连续n行到剪贴板上

剪切：ndd命令 - 剪切将光标所在行开始的连续n行到剪贴板上

粘贴：np命令 - 把剪贴板的内容在光标所在行下面连续粘贴n次

替换：r/R - 替换光标所在处的字符（常用）/替换字符直到按Esc为止

Ctrl + b/f：向上/下移动一屏

---》插入模式下：

Ctrl + n：补全标识符名称（类似西红柿插件功能）

---》命令模式下

:行号： 跳到文件的第n行。

/关键字： 查找关键字，按n切换下一个

======================================================

二、C语言基础知识

1. C语言程序中不同函数的名字必须不同

2. C语言程序中所有不带小数点的数字默认是整数类型(int)，所有带小数点的数字默认double

3. sizeof关键字可以用来计算字节数，在sizeof小括号里对任何存储区内容的修改不会真正发生

4. ASCII码

5. 几个特别的转义字符

---》'\r' 回车字符 出现在这一行开始覆盖

---》'%%' 代表字符%

---》'\0' 与'0'的区别：ASCII不同，'\0' == 0，'0' == 48

1. 几个数据类型

---》1字节(B，byte) = 8位(bit)

---》程序中所有不带小数点的数字默认是整数类型(int)，所有带小数点的数字默认double

---》在不带小数点的数字后加u，表示这个数字的类型是无符号整数类型

---》在带小数点的数字后加f，表示这个数字的类型是单精度浮点类型

---》可以使用%lf或%lg做占位符把双精度浮点类型数据打印在屏幕上

%f/%lf - float/double，不会省略小数点后多余0；

%g/%lg - float/double，会省略小数点后多余0。

%e - 科学计数法

======================================================

三、二进制、八进制、十六进制

1.二进制相反数求法：各个位上数字取反+1

2.负数十进制转换成二进制 -37

(1)计算出相反数 37

(2)转换成二进制 00100101

(3)各个位上数字取反+1(求相反数) 11011011

注意：负数十进制与负数二进制不能直接互相转化

3.负数的二进制转换十进制 10110100 1000000

(1)各个位上数字取反+1(求相反数) 10110101 1000000

(2)转换成十进制 53 128

(3)再求其相反数 -53 -128

-----------------------------------

4. 可以在程序中用八进制方式表示数字，必须以0作为开头(0152)

---》采用%o做占位符可以把一个整数的八进制表示方式打印在屏幕上，打印结果中不包含0开头

---》如果是%#o，则会在打印的十六进制结果前面加上0

-----------------------------------

5. 可以在程序中使用十六进制方式表示数字，必须以0x作为开头(0xbd)

---》采用%x或者%X作为占位符可以把数字的十六进制表示方式打印在屏幕上，打印结果中不包含0x开头

---》%x做占位符的时候打印结果中的英文字母都是小写的

---》%X做占位符的时候打印结果中的英文字母都是大写的

---》如果是%#x，则会在打印的十六进制结果前面加上0x

==================================================

四、操作符及运算

1. 一条语句中使用多个赋值操作符时，从右向左算

num = num1 = 4; //从右向左算

2. 不要在一条语句中对同一个变量多次进行自增自减（++/--）计算，因为结果不确定

3. 位操作符可以把两个数字做二进制运算：~,&,|,^,>>,<< (多练习)

---》~是一个单目位操作符，它可以把一个数每个二进制数位取反

---》双目位操作符包括：&(按位与),|(按位或)以及^(按位异或)（异或：不同为1）

4. 移位操作符可以把数字中每个二进制数位统一向左或者向右移动n个位置

---》移位操作会得到一个新数字而不会修改原有数字(这样就会修改数据：ch >>= 1)

5. 布尔值 ? 公式一 : 公式二

6. 隐式类型转换

---》如果一个表达式里的数字类型不同就必须首先把这些数字转换成同一个类型然后再进行计算

---》这个转换过程叫隐式类型转换，由计算机自动完成

---》隐式类型转换过程中一定把占地小的类型转换成占地大的类型

---》如果不同数据类型占地大小一样，就把整数类型转换成浮点类型，把有符号类型转换成无符号类型

7. 强制类型转换

---》可以在程序中给数字指定一个类型，这叫做强制类型转换

---》语法如下：(char)num

---》强制类型转换可能会导致数据内容丢失

==================================================

五、选择和循环语句

switch(表达式)

{

case 常量表达式1： 语句1； break；

case 常量表达式2： 语句2； break；

case 常量表达式3： 语句3； break；

default： 语句； break；

}

continue; 直接进行下一次循环；

break; 跳出当前循环体；

do...while格式也可以用来实现循环，这种循环至少会执行一次

==================================================

补充：生成随机数的方法

<stdlib.h>

<time.h>

srand(time(0))； //初始化随机数种子

rand()%100+1 //[0,1)

==================================================

六、输入、输出缓冲区

1. 输入缓冲区

---》在不同速度的设备之间传递数据的时候需要使用缓冲区临时储存数据

---》scanf函数会把键盘输入的字符临时存储到一个输入缓冲区，程序实际上从这个输入缓冲区里获得数据

---》先进入输入缓冲区的数据必须优先处理

---》如果输入的内容和scanf函数要求的格式不一致，则会一直无法被处理，并导致后面输入的内容也无法处理

---》可以使用如下两条语句把输入缓冲区里可能存在的错误数据丢弃

scanf("%\*[^\n]"); //把输入缓冲区里第一个换行字符前的所有内容丢弃

scanf("%\*c"); //把输入缓冲区里第一个换行字符丢弃

2. 输出缓冲区

---》printf函数使用输出缓冲区临时存储要打印的内容

---》只有满足如下四个条件中的某一个，输出缓冲区里的内容才会显示在屏幕上

（1）'\n'换行字符前面的内容会被打印在屏幕上

（2）当输出缓冲区被充满的时候里面的内容会被打印在屏幕上

（3）当主函数结束后程序打印的内容出现在屏幕上

（4）可以使用fflush(stdout)语句把输出缓冲区里的内容强制显示在屏幕上

=============================================

1. 数组

1. 数组名称不可以被赋值

---》计算机把数组当指针处理的(相关地址的数据访问)

1. 数组名既可以表示数组中第一个存储区的地址，也可以表示数组整体

---》计算机处理方式：arr[3] <=> \*(arr + 3)

3. 变长数组

---》只有C99规范中才允许使用变长数组

---》变长数组不可以初始化，且赋值长度后，长度不可更改

---》变长数组在长度给定后，才可以声明

4. 用gdb调试打印01arr.c/04arr.c中内容，以明确多维数组各种写法的含义(arr, &arr, &arr[0]等)

=============================================

八、函数

1. 函数存储区：（存放临时变量，如：形参和返回值）

---》从被调函数只能传递一个数据到调用函数，这个数据叫做这个被调函数的返回值

---》函数传递参数、返回值的过程通常伴随着赋值操作，这些数据需要函数提供的存储区存放

2. 函数返回值

---》函数不可以返回非静态局部变量的地址（结果不确定）。

---》数组不可以作为函数返回值

1. 函数形参

---》C语言中允许函数的形式参数的个数不确定，这种参数叫做变长参数，如：printf，scanf，excl等

---》数组可以作为函数形式参数使用（真正形参是指针）

---》数组形参声明时，可以省略元素个数，但需要另外一个整型形参表示数组中包含的元素个数

---》void func（int arr[], int size）

4. return关键字

---》return关键字：存放数据到函数返回值存储区，并结束函数（函数结束）

---》exit标准函数：可以在任何地方结束程序的执行（程序结束）//exit(0);

注意：为了使用这个标准函数需要包含stdlib.h文件

5. 递归函数编写步骤：

（1）编写语句描述问题的分解方式（假设递归函数已经编写完成）

（2）在函数开头编写分支处理不可分解的情况，这个分支必须保证函数可以结束

1. 变量的作用域（空间范围）和存储区的生命周期（时间）

---》局部变量（又叫自动变量）前，都默认有auto关键字修饰，表示自动管理内存

---》全局变量和局部变量可以重名，这个时候语句会优先使用局部变量（就近原理）

---》静态局部变量离开作用域后，变量名无意义了，但可以通过指针访问

=============================================

九、指针

1. 指针变量的用途就是用来找另外一个变量的

2. 只有捆绑过的指针才可以使用

3. 所有没捆绑的指针必须记录空地址(NULL),这个地址的数值是0

4. 如果一个指针既没有捆绑过也没有记录空地址则称为野指针，程序中应该避免出现野指针

5. 指针初始化的时候，\*没有参与赋值过程，只是定义了类型

---》Int \*p=&a;

---》Int \*p1, p1=&b;

1. 可以把指针看作是存储区的某种身份（最大值，最小值等），指针的高级用法

7. 数组名称可以表示数组中第一个存储区的地址

---》计算机处理方式：arr[3] <=> \*(arr + 3)

8. 如果指针记录了数组中第一个存储区的地址则可以通过这个指针找到数组中每个存储区

---》int \*p\_num = arr;（指针捆绑数组名）

---》计算机处理：p\_num[3] <=> \*(p\_num + 3)

-----------------------------------------

9. 地址运算

（1）地址数据可以参与如下计算过程：

---》地址 + 整数

---》地址 - 整数

---》地址 - 地址

10. 指针和数组名称的区别

（1）数组名称不可以被赋值，指针变量可以被赋值

（2）它们的sizeof计算结果不同 sizeof(指针)=4；

（3）它们取地址操作的结果不同（&p，&arr）

--------------------------------------------

1. 指针作为函数形参

---》数组作为函数形式参数的时候真正的形式参数是一个指针变量

---》指针形式参数可以让被调用函数使用其他函数的存储区

1. 指针作为函数返回值

---》数可以把一个存储区的地址作为返回值传递给调用函数

---》不可以把非静态局部变量的地址作为返回值使用

--------------------------------------------

14. const关键字（const修饰写在前面的类型）

（1）const关键字修饰指针

---》const int \*p=&a; //常量指针

---》int const \*p=&a; //意义同上

---》int\*1const p=&a; //指针常量

（2）const关键字修饰变量（几乎等同于常量）（此项可能只报警告）

---》int const a = 100；

---》const修饰的变量必须初始化，否则后面无法修改

---》只有常量指针可以指向const修饰的变量（或其他字符串常量）

--------------------------------------------

15. 无类型指针void\*（泛型指针）

---》这种指针必须首先强制类型转换成有类型指针然后才能使用

---》可以指向任何类型数据（包括基本数据类型、复合类型、指针、二级指针等）

---》使用无类型指针作为形参，可以把任意类型的存储区从调用函数传递给被调函数

16. 二级指针

---》二级指针用来记录指针变量存储区的地址

---》二级指针变量名称前加\*代表捆绑的一级指针存储区

---》二级指针变量名称前加\*\*代表和它捆绑的一级指针存储区所捆绑的普通变量存储区

---》\*\*pp\_num = 10; //一个\*抵消一个p

17. 几种指针比较

---》指针可以表示一维数组

---》二级指针可以表示指针数组

---》二级指针不能表示二维数组

---》数组指针可以表示二维数组

---》二级指针作为函数形参，可以让被调函数使用调用函数的指针类型存储区

--------------------------------------------

18. 函数指针

（1）C语言中函数也有地址，函数名称可以表示函数的地址

（2）函数指针可以记录函数的地址

（3）函数指针声明可以根据函数声明变化得到

---》int (\*p\_add)(int, int) = NULL;

---》p\_add = add;

（4）不同类型的函数指针适合与不同格式的函数捆绑

（5）回调函数

---》函数指针可以作为形参使用

---》可以作为实际参数使用的函数叫回调函数（程序自动调用）

---》回调函数提供调用函数的各种工作模式

=============================================

十、字符串、清理语句和指针数组

1. 字符串的定义

（1）C语言中，所有字符串必须以特定的字符'\0'作为结尾（这个字符的ASCII码就是数字0）

（2）'\0'字符前的内容是有效文字内容

（3）'\0'（转义字符）表示只占一个字节的数字0；'0'表示字符0

2. 字符串的两种表示方式：

---》字符串字面值是一种字符串，用双引号中间的一组字符表示，例如"@#%sdg","sdg"

---》编译器会自动在最后加上'\0'字符，字符类型指针可以指向字符串首地址

（1）字符串字面值必须用字符类型指针表示(字符指针)

---》编译器会把字符串字面值替换成第一个字符的地址

---》字符串字面值里的所有字符在程序运行过程中不能修改

---》多个内容一样的字符串字面值其实是同一个字符串(地址相同)

---》多个并列写在一起的字符串字面值会被自动合并 ("abc""def"会自动合并成"abcdef")

（2）字符数组也可以用来表示字符串(字符数组)

---》程序中要保证字符数组中包含'\0'字符，否则字符数组不能作为字符串使用

---》可以使用字符串字面值对字符数组进行初始化，编译器会把字符串字面值里的'\0'字符也放入字符数组

---》用字符串字面值对字符数组初始化完成后它们的存储区完全无关

---》字符数组里的字符串可以修改

char str[10] = {'a','b','c'}; //正确

char str[ ] = {'a','b','c'}; //错误，没有分配'\0'

char str[ ] = "abc"; //正确

1. 无论是字符串字面值，还是字符数组做形参时，实际都是指针做形参。

4. 不同表示方式，在进程中的内存分布

char \*p = "Zimmer";

char str[ ] = "Zimmer";

---》"Zimmer"本身在代码段(只读常量区)

---》p指向代码段，本身在栈区；字符指针存储字符串的首地址，不能存内容

---》str指向栈区，本身也在栈区；字符数组存储字符串的内容，而不是地址

--------------------------------------------

5. 字符串操作函数

---》不可以使用操作符对字符串进行操作，而应该使用一组函数

---》Vi当中输入man 函数名，可以显示该函数的使用格式及作用

（1）以下函数需要包含string.h文件

strlen 用来统计字符串中有效字符的个数('\0'之前为有效)

strcat (修改原来字符串)合并两个字符串，有可能修改数组外存储区，这会导致严重错误

strncat 功能和strcat类似，可以避免出现问题

strcmp 比较两个字符串的大小(ASCLL码大小)，前者小于后者，结果-1；反之，1

strncmp 只比较两个字符串里前n个字符

strcpy 把一个字符串复制（覆盖）到另外一个字符数组里，有可能修改不属于数组的存储区，这会导致错误

strncpy 功能和strcpy类似，可以避免发生错误

memset 可以把字符数组中所有字符存储区填充同一个字符数据(内存拷贝)

---》char str[ ] = "Zimmer";

---》memset(str, ’h’, 3); //从第一个开始，连续3个字符都变成’h’，输出hhhxyz

---》memset(a, 0,sizeof(a)); //变长数组a全部变成0

strstr 在一个字符串中查找另外一个字符串所在的位置（第一个字符的位置）

---》char str[ ] = “abcdef”;

---》strstr(str, “cde”); //返回’c’所在地址

（2）以下函数不需要包含string.h文件

sprintf 按格式把数据打印在字符数组中，形成字符串

---》sprintf(str, “%d %c %g”, num, ch ,fnum);

sscanf 从字符串中得到数据并记录在变量里

---》sscanf(“p 2.4 62”, ”%c %g %d”, &ch, &fnum, &num);

（3）以下两个函数使用时需要包含stdlib.h文件

atoi 把字符串里的整数转换成整数类型（提取整数）

---》int num = atoi(“486”);

atof 把字符串里的浮点数转成浮点类型（提取浮点数）

--------------------------------------------

6. 清理语句

（1）输入、输出函数

---》在printf函数调用语句里用%s做占位符就可以把一个字符串打印在屏幕上

---》在scanf函数调用语句里用%s做占位符就可以从键盘得到一个字符串并把它记录在字符数组里

（2）scanf缺点：

---》如果输入内容中有空格则只能得到空格前的部分

---》如果输入内容超过数组容量则会出严重错误

1. fgets函数

---》该函数也可以用来从键盘得到字符串并记录到一个数组里，并且可以避免scanf函数问题

---》这个函数需要三个参数：数组名称、数组中存储区个数、用stdin表示键盘

---》fgets(str ,sizeof(str), stdin);

（4）fget函数特点

---》如果输入的字符个数不够则会把最后输入的回车当作'\n'字符也放到数组里

---》如果输入的字符过多则只会截取前面的一部分,下次读字符串的时候会从后面没有处理的内容里读取

---》应该在每次使用fgets函数获得字符串以后把可能存在的垃圾数据清理掉

---》清理语句应该放在一个if分支里，只有当确保有垃圾数据的时候才需要清理

char str[10] = {0};

fgets(str, 10 , stdin); //0, 1, 2, 3, 4, 5, 6, 7, 8, 9

if(strlen(str)==9 && str[8]!='\n'){//str[10] = {'a','s','f','d','f','e','v','c','x','\0'};

scanf("%\*[^\n]");

scan("%\*c");

}

--------------------------------------------

7. 指针数组

---》指针数组也是数组，它里面的每个元素都是指针类型的

---》字符指针数组中每个元素是一个字符类型指针，它可以用来代表一个字符串

（1）可以采用一个字符指针数组记录多个相关字符串

char \*strs[5] = {“China”,”Russia”,”America”,”France”,”England”};

printf(“%s\n”, strs[num]); //表示字符指针

（2）二维字符数组也可以用来记录多个相关字符串（占用内存多）

char strs[5][10] = {“China”,”Russia”,”America”,”France”,”England”};

printf(“%s\n”, strs[num]); //二维数组中某一个字符数组

（3）通常采用指针数组记录多个相关字符串（占用内存少）

（4）主函数的参数就使用字符指针数组记录了多个相关字符串

int main(int argc, char \*argv[ ]){

int num = 0;

for(num = 0;num <= argc - 1;num++){

printf(“%s\n”, argv[num]);

}

return 0;

}

=====================================================

十一、预处理指令、多文件编程

1. 宏定义

（1）宏只做替换，宏替换在预处理阶段完成

（2）宏可以替换数字、公式等

（3）替换数字时，可以不定义宏，改为在编译时添加选项：

---》如：gcc -DSIZE=9 -DMAX=50（表示定义了宏SIZE、MAX）

（4）宏没有存放返回值和参数的存储区，所以不能保证替换后的计算优先顺序，因此：

---》所有宏的参数加括号

---》整个公式加括号

（5）不要把自增或自减的计算结果作为宏的参数使用(一条语句会出现多次自增或自减)

---》例如：#define ADD ((++a)\*(++a))

---》((++5)\*(++5))=49或者42

1. 宏操作符（#、##）

---》#操作符： 把宏的一个参数转换成字符串字面值

---》##操作符： 把宏的一个代表标识符的参数和其他内容合并为一个新的标识符

#define STR(a) #a ==> "a"

#define ADD(b) m\_##b ==> m\_b

#define ADD(a, b) a##b ==> ab

--------------------------------------------

1. 条件编译可以决定什么语句应该编译什么语句不应该编译

（1）#define/#undef UNICODE

（2）#ifdef/#ifndef...#else...#endif（二选一）

（3）#if...#elif...#else...#endif（多选一）

（4）#if defined(YI) //判断是否定义过该宏

（5）#elif defined(ER)&&!defined(YI) //同理，判断布尔值

1. 多文件编程
2. 实现步骤
3. 头文件卫士
4. extern/static关键字

---》全局变量在别的文件中使用，需要在别的文件中用extern声明

---》static修饰的全局变量和函数只能在本文件中使用

---》static关键字经常被用于修饰不同文件定义的同名函数或变量，避免与别的文件反生冲突

=====================================================

十二、结构（体）和数据对齐补齐

1. 结构（体）

（1）结构体声明语句中需要使用struct关键字

（2）C语言中结构体不可以包含成员函数

（3）把struct关键字和结构体名称合起来作为类型名称就可以定义结构体变量了

（4）可以使用typedef关键字给数据类型起别名，代替原来的类型名称

---》可以把声明结构体的语句和起别名的语句合并成一条语句

---》这个时候可以省略结构体本身的名称，但是以后就只能用别名声明结构体变量了

（5）同类型结构体变量之间可以直接赋值

（6）结构体成员的访问方式

---》结构体变量名.成员变量名，如：prsn1.age

---》结构体指针->成员变量名，如：p\_prsn->age

--------------------------------------------

2. typedef的使用步骤(参考代码typedef.c)

第一步，定义变量： int count;

第二步，起别名： typedef int count;

第三步，使用： count i; //int i;

总结：新定义的别名就是脱去类型后的关键字，别名的数据类型就是除去typedef关键字后，定义的数据类型

（3）两个举例

int array\_t[3];

typedef int array[3];

array a; //int a[3];

-------------------------

struct node{

int data;

struct node\* next;

};

typedef struct node node\_t;

typedef node\_t\* node\_p;

node\_p\* q; <==> node\_t\*\* q;

--------------------------------------------

3. 数据对齐和补齐

（1）数据对齐

---》一个存储区的地址必须是它自身大小的整数倍，这个规则叫数据对齐

---》另外，double类型存储区的地址只需要是4的整数倍

---》结构体内部的成员变量通常也需要遵守数据对齐的规则

---》数据对齐可能导致结构体不同子存储区之间有空隙

（2）数据补齐

---》结构体存储区的大小必须是内部包含的最大基本类型子存储区大小的整数倍，这个规则叫数据补齐

---》另外，如果最大基本类型子存储区是double类型的，则结构体存储区的大小只需要是4的整数倍

---》数据补齐可能造成结构体存储区最后有浪费的字节

--------------------------------------------

4. 枚举（enum）

---》计算机用整数0代表枚举类型中第一个名称，后面的名称依次递增(0,1,2,3...)

---》类似使用宏定义了一串连续的自然数（0~n）

---》可以在声明枚举类型的时候指定某个名称用某个整数表示(这时后面的名称也会随之改变)

enum {CHUN, XIA = 5, QIU, DONG}; //CHUN = 0, QIU = 6

---》使用宏可以实现和枚举类似的效果

5. 联合（union）

---》形式类似于结构体，所有成员占用同一块内存空间

---》联合里同样包含多个不同的存储区，这些存储区的地址都一样，修改了任何一个其他的会受影响

---》联合的大小是其中最大部分存储区的大小

=====================================================

十三、动态分配内存、标准函数库

1. 相关函数

---》malloc/free/calloc(分配数组)/realloc(调整大小)

2. 释放动态内存

---》当不再使用动态内存的时候，必须释放，防止内存泄漏

---》某一次分配的所有内存必须一次释放掉，而且只能释放一次

---》free后的指针成为野指针，必须置NULL

---》free后的动态内存空间内容不确定(可能是原来数据，也可能被别的变量使用了)

--------------------------------------------

3. 标准函数库

#include<time.h>

ctime： 把time函数得到的结果转换成一个时间字符串

gmtime： 把time函数得到的结果按照格林尼治时间转换成一个结构体

localtime： 把time函数得到的结果按照当前时区转换成一个结构体

asctime： 把一个记录时间的结构体转换成字符串

=====================================================

十四、文件操作

1. 文本文件和二进制文件

（1）计算机里所有文件记录的都是二进制数据

（2）文本文件里的所有二进制数据都来自于字符数据

（3）文本文件以外的文件叫做二进制文件

（4）所有文件都可以采用二进制方式进行操作

2. 文件操作函数

fopen(3)、fclose(3)、fwrite(3)、fread(3)

3. 文本文件格式化输入、输出函数（fprintf、fscanf）

---》只能操作文本文件

---》fprintf(p\_file, "%c %g %d", ch, fnum, num);

---》fscanf(p\_file, "%c %g %d", &ch, &fnum, &num);

4. 文件指针相关函数

ftell函数： 获得位置指针的数值

rewind函数： 把位置指针调整到文件开头

fseek函数： 把位置指针调整到文件中任何位置

补充：用vi编写的文件，在文件尾有一个看不见的’\n’字符

=====================================================

学习C语言的三句箴言：

1.区分常量和变量

2.先找到变量的地址，然后根据类型访问变量的内容

3.运算符的优先级及其结合性。 例：int (\*p)[3];p是指针类型的变量。

补充：

a)什么是数据类型？

\* 是数据类型。

数据类型就是数据在内存中的访问方式。

指针不是地址，地址属于指针。

int char short float double void \*

int main(int argc,char \*argv[])

int \*p[3];指针类型的数组;数组p有三个元素，每个元素中存放的都是地址。

int (\*p)[3];p是指针类型的变量;指向有三个元素的数组

int (\*fun)(int,int);函数指针

int \*fun(int,int);函数声明

\*\*\*\*\*\*\*

所有跨函数使用存储区都是通过指针实现的

目前掌握的跨函数使用存储区的办法有两种:

1. 一种是被调用函数可以使用调用函数的存储区，（不可以使用变量）。反过来不可以。

小用大(用指针做形式参数传址调用,并注意加const关键字)

1. 另外一种是静态局部变量(static)的存储区可以被任意使用
2. 动态分配内存可以实现跨函数使用存储区

\*\*\*\*\*\*\*