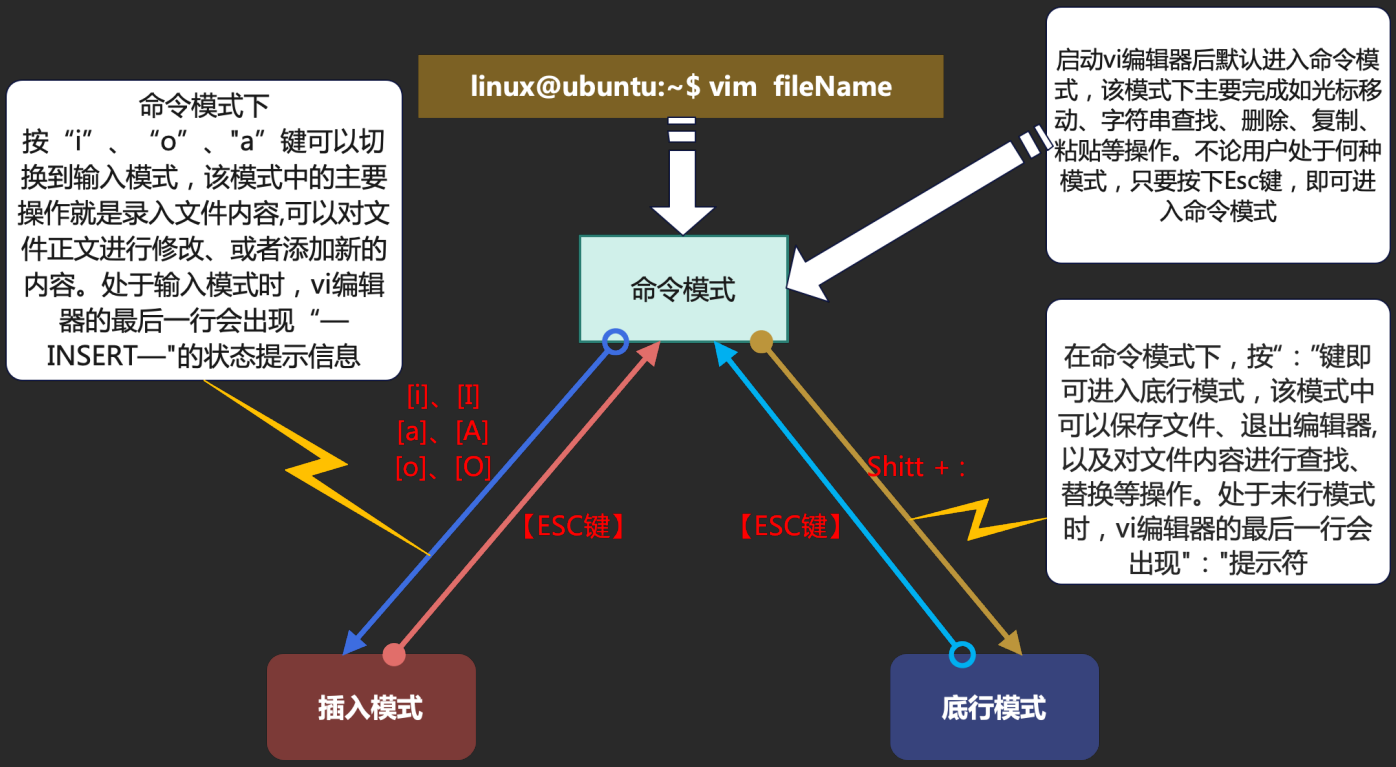
1、vim编辑器的使用

```
1 vim是ubuntu系统中的一个文本编辑器，现阶段使用vim编辑器。
2 后续课程会给大家配置vscode工具，是windows中的一个软件。
3
4 vim文本编辑器有三种工作模式： 命令行模式/插入模式/底行模式
```



1.1 打开vim编辑器

```
1 vi/vim    路径/文件名        ---> 打开文件,
2                              ---> 如果存在则直接打开
3                              ---> 如果不存在则先创建再打开
4 vi/vim    路径/文件名1    路径/文件名2   -O   --> 左右分屏打开文件名1和文件名2
5
```

1.2 vim编辑器的三种工作模式

1.2.1 命令行模式

```
1  vim打开文件时，默认就处于命令行的模式。
2
3  通过上下左右箭头移动光标的位置。
4
5  复制
6       yy    ---> 复制光标所在的行
7       nyy   ---> 复制光标所在行下的n行，n是一个
   整数
8  粘贴
9       p     ---> 粘贴到光标的下一行
10      np    ---> 在光标下一行粘贴n次，n是一个整
   数
11      shift + p ---> 在光标上一行进行粘贴
12
13 剪切
14      dd    ---> 剪切光标所在的行
15      ndd   ---> 剪切光标所在行的下的n行，n是一
   个整数
16
17 回撤
18      u     ---> 撤回上一次的操作
19
20 搜索
21      /查找字符串    ---> 按下/键，输入查找的字
   符串，回车
22
23      n ---> 下一个
24      shift + n ---> 上一个
```

gg ----> 光标回到文件第一行
shift + g ---> 光标回到文件的末尾行

```
1       gg=G   ---> 代码对齐及缩进
2       或者
3       选中要对齐的代码，然后按下"="键，将选中的代码进行对齐缩进处理
```

1.2.2 插入模式

```
1 思考，按下i I a A o O，查看光标位置的变化
2 i : 进入插入模式，在光标前开始插入数据(使用频率最高)
3 I(shift + i) : 进入插入模式，将光标定位到行首开始插入数据
4 a : 进入插入模式，在光标后开始插入数据
5 A(shift + a) :  进入插入模式，将光标定位到行位开始插入数据
6 o : 进入插入模式，  在光标的下边插入新的行，开始插入数据
7 O(shift + a) :  进入插入模式，  在光标的上边插入新的行，开始插入数据
```

1.2.3 底行模式/末行模式

```
1 在命令行模式下按下键盘的shift + : 进行到底行模式。
2       :w  ----> 保存文件不退出
```

```
    :q  ----> 不保存，直接退出，要求文件不能被修改
    :q! ----> 不保存，强制退出，不管文件是否被修改
    :wq ----> 保存并退出，注不可以写成qw
    :wq!    ---> 强制保存并退出
    :x   ---> 强制保存并退出
    :wqa ---> 全部保存退出，左右分屏打开多个文件时

设置行号的显示或者不显示:
    :set nonu   ---> 取消行号的显示
    :set nu      ---> 显示行号

取消高亮的显示，当使用搜索功能时，搜索到的字符串默认会高亮，
    :nohl

剪切:
    :n,md          --> n到m行进行剪切，粘贴使用P

复制:
    :n,my        ---> n到m行进行复制，粘贴使用P

替换:
    %s/要被替换的字符串/替换的字符串/g    ---> 全部进行替换
    %s/要被替换的字符串/替换的字符串/     ---> 只替换每行中首次出现的字符串
```

```
26
27      n,ms/要被替换的字符串/替换的字符串/g    --> 替换n到m行出现的所有的字符串
28      n,ms/要被替换的字符串/替换的字符串/   ---> 只替换n到m行中，每行中首次出现的字符串
29
30
```

2、第一个C代码

2.1 编写hello world的应用程序

01hello.c文件：

```c
// #include ：用来包含系统定义的或者自定义的头文件
// <头文件的名字> ：  默认从系统指定的头文件路径下查找对应的头文件
//          使用系统提供的头文件时一般使用<>进行包含。

// "头文件的名字" ：  先从当前目录下查找对应的头文件，如果没有找到，
//                                  在从系统指定的路径下查找对应的头文件
// 双引号包含头文件的方式一般用于自定义的头文件的包含
```

```c

// stdio.h : 标准输入输出的头文件
// std : standard   i : input   o : output
// 使用的printf和scanf函数就在stdio.h头文件中进行声明
#include <stdio.h>       // 也可以使用 #include "stdio.h"

// 输入main字符串，按下tab键，自动将主函数补全

// 每个应用程序有且只能有一个main函数，应用程序只能有一个入口
// int ：main函数前边的int，表示main函数的返回值，程序执行结束之后
// 返回一个int类型的值。
//int argc, const char *argv[] : main函数的参数，后续讲解
int main(int argc, const char *argv[])
// {} : main函数的函数体，要想自己编写的代码被执行，需要写到{}中
{
    // printf : 打印的函数，将结果打印到屏幕
    // 将""以内的字符串在屏幕上打印输出
    // \n : 换行符  将光标定位到下一行的行头

    // \r : 回车   将光标定义到当前行的行头
    // \t : tab 制表符
    printf("hello world!\n");

```

```
    // main函数的返回值，程序执行结果返回0，表示程序正常退出
    return 0;
}

```

```
自己编写的每一个应用程序，都需要包含以下内容：

#include <stdio.h>

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

    /* user code 用户代码 */

    return 0;
}

```

2.2 gcc编译器

```
C语言属于编译型的语言，要想C语言的程序可以运行必须使用编译器将其
编译生成对应的可执行的文件。

GCC属于GNU工具集中的一个二进制工具。GNU是一个开源的组织。

编译型的语言：
```

```
7       特点：需要使用编译器将其编译生成对应的可执行程序，C/C++都属于编译型的语言
8       优点：编译型的语言在执行时已经被编译，执行的效率高，安全性高。
9       缺点：依赖编译器，使用编译器将程序编译生成对应平台的可执行程序
10  解释型的语言：
11      特点：不需要使用编译器进行编译，但是需要使用解析器进行解析，
12              Python语言，shell脚本语言。
13      优点：可以移植性强，跨平台较好。
14      缺点：执行效率低，需要对每条指令都要进行解析，不安全。
```

2.3 编译c的程序生成对应的可执行文件，并执行

```
1   方式1：生成默认名字的可执行程序
2       gcc ****.c      ----> 默认生成一个名字为a.out的可执行程序
3                   注：可执行程序的后缀不是.out，是elf格式的可执行文件。
4
5       执行程序：  ./a.out
6
7   方式2：生成指定名字的可执行程序
8       gcc  ****.c -o ****(自定义的可执行程序名)
```

```
9       gcc   -o ****(自定义的可执行程序名) ****.c
10
11      gcc   -o ****.c   ****(自定义的可执行程序名)    // 错误的写法
12
13      -o 参数后边跟的是输出文件的名字。
14
15      执行程序:    ./****(自定义的可执行程序名)
16
17  方式3：分步实现编译生成可执行程序(重点要理解，面试题，C高级时Makefile会使用)
18      从.c文件到可执行程序需要4步，预处理->编译->汇编->链接
19      1> 预处理：头文件的展开，宏定义的替换，注释的删除,不检查语法的错误
20      gcc -E ****.c -o ****.i
21      2> 编译：语法分析，词法分析，检查语法错误，如果有错输出对应的错误，
22              如果没有错误生成对应汇编文件。
23      gcc -S ****.i -o ****.s
24
25      3> 汇编:  将.s文件生成对应目标文件(二进制文件，不可以执行，原因没有进行链接)
26      gcc -c  ****.s -o ****.o
27
28      4> 链接 :  将.o文件链接生成可执行文件(可以执行)
29      gcc ***.o -o ****(自定义可执行程序的名字)
30      或者
```

```
31      gcc ***.o          ----> 生成a.out可执行程序
32
```

3、计算机数据存储

```
1 计算机中数据的存储分为两类：数值型数据和非数值型数据。
2
3 数值型数据   : 3.14   520   10000   0x500   0b0101   0897
4 非数值型数据 : 'A'   '8'   '\n'   '\r'   "hello world"
5
```

3.1 数值型数据的表示方式以及不同数值类型数据的相互转换

3.3.1 十进制数

```
方便人类识别和使用的数字。
对应的数字为0-9；
特点：逢十进一
前导符：无
比如：520   1314   410   123456789

```

3.3.2 二进制数

```
计算机可以识别的数字，计算机只能识别0和1。
计算机识别和存储的不是0和1，而是电信号，为了方便人们去表示电信号，
0表示低电平信号，1表示高电平信号。

特点：逢二进一
前导符：0b
比如：  0b01    0b110   0b00001
```

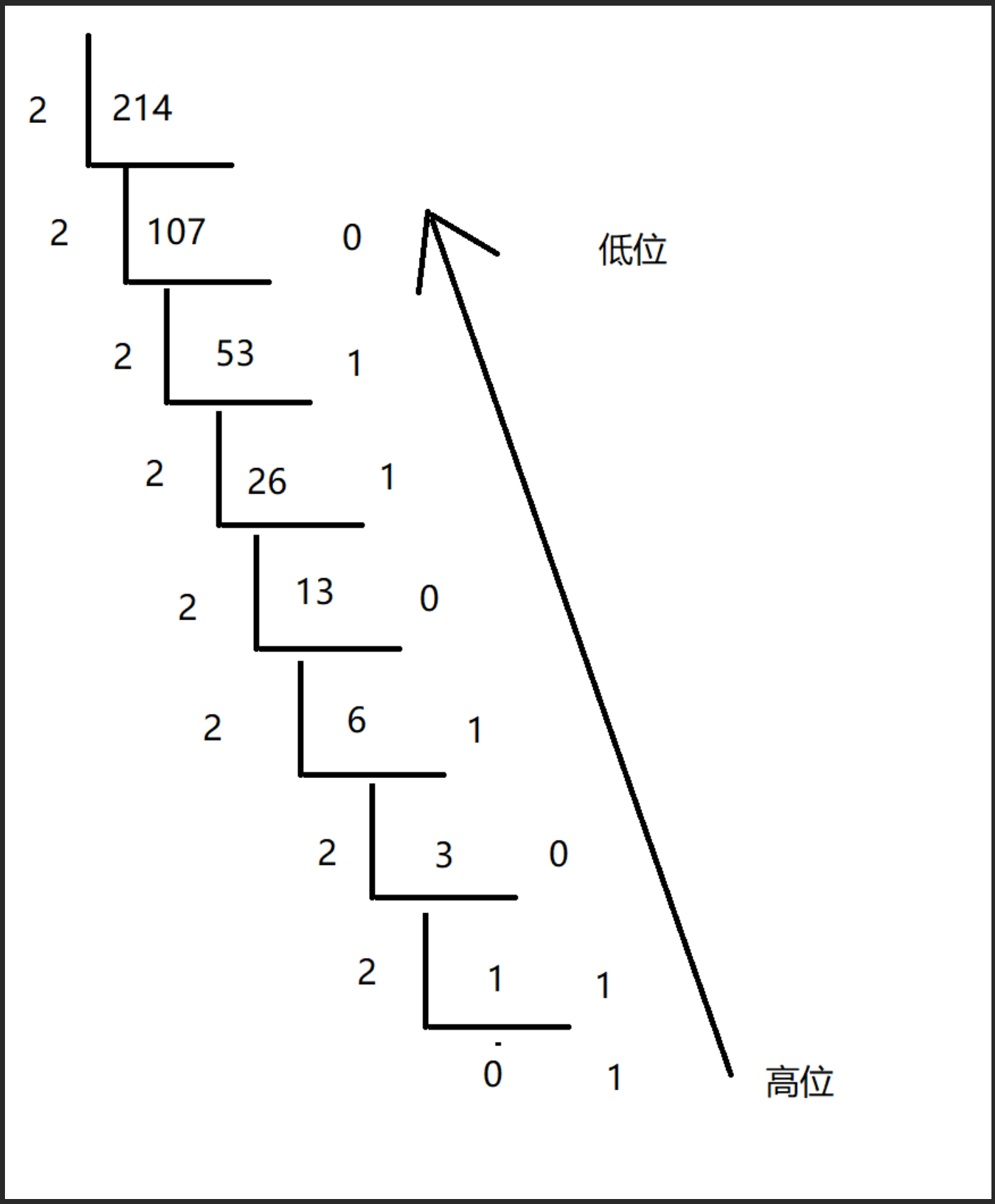
```
二进制转十进制数：

0b1101 0110             ----> 二进制数据的最低为在右边，记为第0位

=> 1*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 0*2^0 =
     128  + 64    + 0     + 16    + 0     + 4     + 2     + 0     =
     214

```

十进制转换为二进制：
辗转相除法（相除取余）：
　　让十进制的数除以2，得到商和余数，
　　然后再让商除以2，得到商和余数，
　　依次类推，直到商为0即可。
　　将所有的余数倒叙取出，最结尾的数是二进制数的高位


2
214
2
107
0
低位
2
53
1
2
26
1
2
13
0
2
6
1
2
3
0
2
1
1
0
1
高位

```
1 二进制和十进制数据之间的相互转换的工具或者使用程序员计算器:
2     https://c.runoob.com/unit-conversion/7988/
```

3.3.3 八进制数

```
1     特点: 逢8进1,  每一位对应的数的范围为0-7
2     前导符: 0
3     比如 :  065   0234   0576
```

```
 1 八进制数转十进制数:
 2 01234    ---> 右边的数据是最低位
 3     = 1*8^3 + 2*8^2 + 3*8^1 + 4*8^0
 4     = 668
 5
 6 十进制数转八进制数:
 7     辗转相除法: 除8取余数。
 8
 9 八进制数转二进制数:
10 二进制数转八进制数: 三位二进制数可以表示1位的八进制数。
11
12     ***     ---> 3位的二进制数, *表示任意, 可以是0也可以是1
13     |||--> 此位为1, 表示1;  此位为0, 表示0。
14     ||---> 此位为1, 表示2;  此位为0, 表示0。
15     |---> 此位为1, 表示4;  此位为0, 表示0。
```

```
16
17      有0，1，2，4任意一个或者多个数进行组合相加可以到到0-7之间的任意一个数。
18
19  举例：
20      07651234   =转换二进制结果如下=>
21      111 110 101 001 010 011 100
22
23      011 101 110 000 001 110 00    ==> 从右到左，依次组合3位的二进制数
24                      |----> 转换，从低位开始，每3位进行组合，高位不足三位补0
25      001 110 111 000 000 111 000
26                      |----> 在转换为8进制数
27              01670070
```

3.3.4 十六进制数

```
1       特点：逢16进1，  每一位对应的数的范围为0-9，A B C D E F
2       前导符：0x
3       eg ：  0x65   0x234   0x576   0xABC
```

```
1  十六进制数转十进制数：
2       0x123   =
3       1*16^2 + 2*16^1 + 3*16^0 = 291
4
```

十六进制数转二进制数：(4位二进制数对应1位十六进制数)

    ****   ---> 4位的二进制数，*表示任意，可以是0也可以是1

    ||||--> 此位为1，表示1； 此位为0，表示0。

    |||---> 此位为1，表示2； 此位为0，表示0。

    ||---> 此位为1，表示4； 此位为0，表示0。

    |----> 此位为1，表示8； 此位为0，表示0。

    有0，1，2，4，8任意一个或者多个数进行组合相加可以得到0-F之间的任意一个数。

    案例：

    0xFC87 =转换成二进制的结果=>

    1111 1100 1000 0111

十六进制数转八进制数：先转换为二进制数，在转换为八进制数。

    案例：

    0xFC87 =转换成二进制的结果=>

    1111 1100 1000 0111 =转换为3位二进制=>

    001 111 110 010 000 111 =转换为八进制数的结果=>

    0176207

十进制数转换为十六进制数：

    辗转相除，除以F，取余数(一般不用)。

二进制数转换为十进制数:(4位二进制数对应1位十六进制数)

八进制数转换为十进制数:(先将八进制数转换为二进制数，在将二进制数转换为十六进制数)

总结:

十进制转二进制;
十进制转八进制;
十进制转十六进制;
二进制转十进制;
八进制转十进制;
十六进制转十进制; ---> 以上转换借助程序员计算器

二进制转八进制;
二进制转十六进制;
八进制转二进制;
十六进制转二进制; ---> 必须可以口算完成

4位二进制数对应1位十六进制数; ------> (8421)
3位二进制数对应1位八进制数; ----> (421)

```
练习题：
十六进制转二进制：
    0x12345678 ===> 0b0001 0010 0011 0100 0101 0110 0111 1000
    0xA754EF   ===> 0b1010 0111 0101 0100 1110 1111

八进制数转二进制数：
    0520         ===> 0b101 010 000
    035647       ===> 0b011 101 110 100 111
```

3.2 非数值型数据的存储

```
    计算机只能识别0和1，但是在实际的使用中有很多的数据都不属于
数值型的数据，比如：性别，人名，网址，公司名等。
    非数值型的数据在内存中最终也会被转换为0和1进行存储，专家编写了
一套ASCII码，专门用来表示非数值型的数据，每个字符在ASCII码表中都
对应着一个整数。
    查看ASCII码表的方式，man ascii，或者谭浩强C语言，附录中有ASCII码表。

    ASCII码表对应的图，在有道云笔记中，自行查看，或者使用以下链接进行查看。
```

```
    https://hqyj-note.oss-cn-beijing.aliyuncs.com/note-picture20230403144497
6.png

    非数值类型的数据的存储最终存储的是ascii码表中每个字符对应的ascii码值。

常见的ASCII码:
    'A' ~ 'Z'  ----> 65 ~ 90
    'a' ~ 'z'  ----> 97 ~ 122
    '0' ~ '9'  ----> 48 ~ 57
    '\n'   ----> 10
    '\0'   ----> 0

转义字符:
    所有的ASCII都可以使用"\八进制的数"进行表示。
        比如:  \101   ---> 表示字符'A',
        "\150\145\154\154\157"  ---> 表示"hello"字符串
        "\150\14511\157"    ---> 此种用法常用在笔试题中, 计算字符串的长度

    有一些字符, 打印时无法显示, 或者此字符已经失去本来的含义:
        比如:  '\n' '\r'  '\t'  '\0'
```

02data.c文件:

```c
#include <stdio.h>

```

```c
int main(int argc, const char *argv[])
{
    // 字符'A'替换%c对应的位置，将""中的字符串打印到屏幕上
    // %c :  如果打印的是一个字符时，使用%c格式化字符
    printf("打印字符 = %c\n", 'A');

    // %d : 如果打印的是一个整数时，使用%d格式化字符
    printf("打印字符对应的ASCII码值 = %d\n", 'A');  // %d : 十进制数
    printf("打印字符对应的ASCII码值 = 0%o\n", 'A');  // %o : 八进制数
    printf("打印字符对应的ASCII码值 = 0x%x\n", 'A');  // %x : 十六进制数

    // %s : 打印字符串对应的格式化字符
    printf("%s\n", "\150\145\154\154\157");

    return 0;
}

```

```c
#include <stdio.h>
int main(int argc, const char *argv[])
{
    printf("helloworld\n");    // /n 换行，光标定位到下一行的行头
    printf("helloworld");    // 没有换行符
    printf("hello\tworld\t!");   // 制表符，tab键
    printf("\n");

printf("sasfdjksadfjiowfejioqwfjoadfiofjoisajfofjowqiuehfwqoierf\r");  // /r : 回车将光标定位到行头
    printf("\n");
    return 0;
}
```

4、C语言中的关键字和标识符

4.1 C语言的关键字

```
关键字：编译器以及指定好的具有特定用途的单词，直接使用编译器就可以识别。
C语言中共计有32个关键字。
数据类型相关的关键字(12个)：
    char short int long float double void unsigned signed enum struct union
```

```
5
6  存储类型相关的关键字(6个):
7      auto const static extern register
   volatile
8
9  自定义类型的关键字(1个):
10     typedef
11
12 计算数据类型大小的关键字(1个):
13     sizeof
14
15 控制语句相关的关键字(12个):
16     if else switch case default for
   while do goto break continue return
17
18 注: C语言中是严格区分大小写, 关键字都是小写。
```

4.2 标识符

```
1  C语言中定义的变量名, 函数名, 结构体变量名, 枚举
   变量名,  联合体变量名,  宏名。
2
3  命名的规范:
4      1> 由数字, 字母和下滑线组成
5      2> 不可以使用c语言中的关键字
6      3> c语言中严格区分大小写
7      4> 不可以使用数字开头, 也不可以有特殊的字
   符, 比如#,$,&...
8      5> 定义变量名时尽量做到"见名知意"。
9
```

```
命名的法则：驼峰命名法或下滑线命名法
    1> 驼峰命名法：
        大驼峰命名法：所有单词的首字母大写。
            比如：LastName  FirstName
        小驼峰命名法：第一个单词的首字母小写，后边单词的首字母都大写。
            比如：lastName  firstName
    2> 下滑线命名法：多个单词之间使用“_”隔开。
        比如：set_value   get_value   _hal_init
```

5、数据类型

5.1 C语言的本质

```
    C语言的本质就是对内存的操作。
    内存的特点：内存中存储的是正在运行的程序或者数据，
        内存中的数据掉电丢失，内存的大小一般为：2G,4G,8G,16G,32G.

    硬盘的特点：没有运行的程序或者数据放到硬盘中，
        硬盘中的数据掉电不丢失，硬盘的大小一般为：512G,1T,2T,5T

```

5.2 内存分配的最小单位

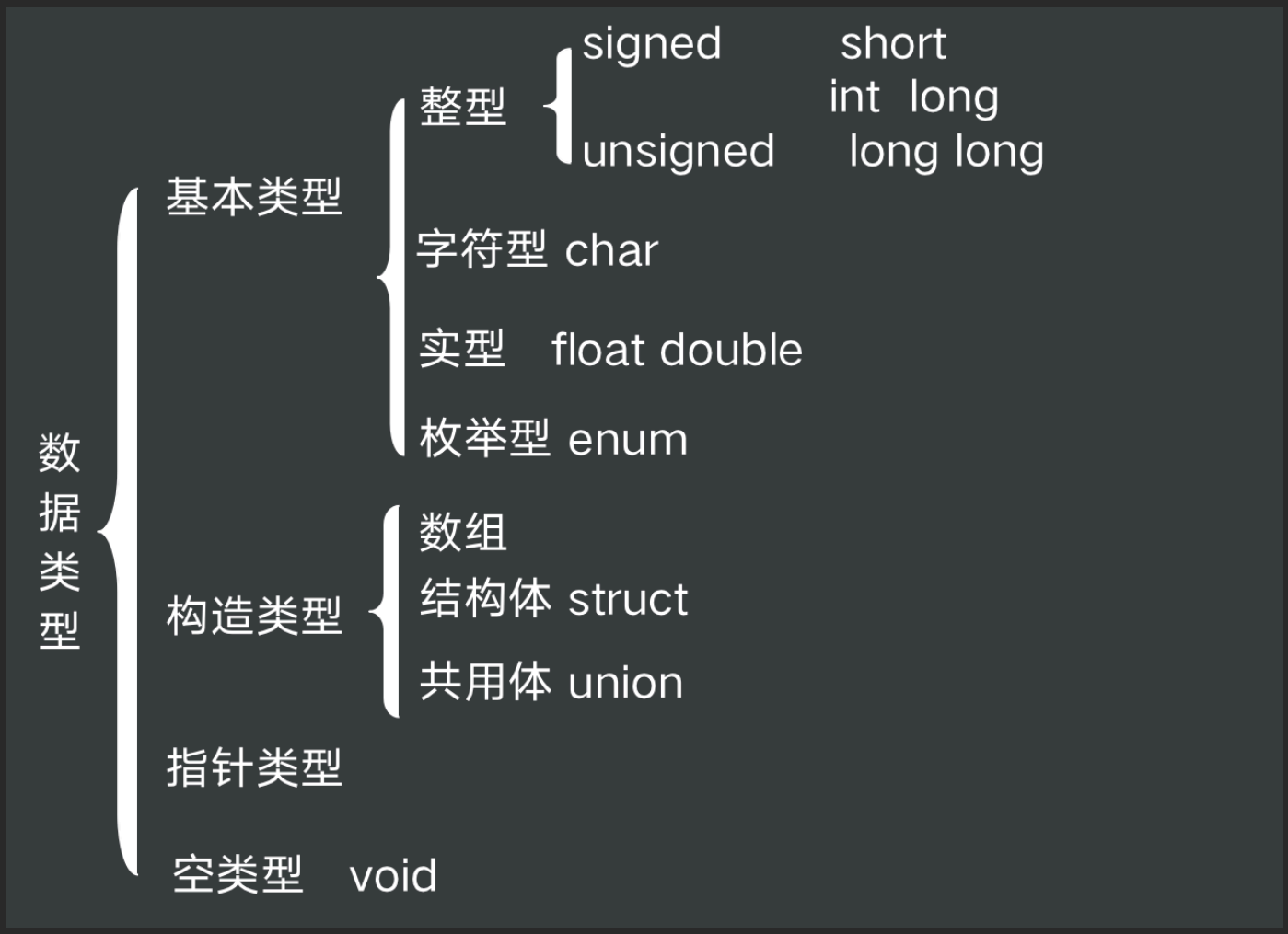
内存分配的最小单位为字节，1个字节占用8个bits的空间，最低位为第0位。
8bits的空间也可以这样表示[7:0]bits

5.3 数据类型的作用

　　可以使用不同的数据类型定义不同类型的变量，从内存中申请不同大小的内存空间。

　　通过不同的数据类型可以描述不同的事物，比如：
　　身高：使用short
　　体重：使用short
　　性别：使用char
　　名字：使用字符数组

5.4 数据类型的分类


数据类型
基本类型
整型
signed
unsigned
short
int
long
long long
字符型 char
实型 float double
枚举型 enum
构造类型
数组
结构体 struct
共用体 union
指针类型
空类型 void

5.5 整数类型

```
1       整数类型：short / int / long / long long
2       有符号的整型：signed short / signed int / signed long / signed long long
3               定义有符号的整型变量时，可以省略signed.
4
5       无符号的整型：unsigned short / unsigned int /
6                       unsigned long / unsigned long long
7               定义无符号的整形变量时，不可以省略unsigned.
8
9       数据在内存中存储时，通过最高位的值区分有符号数为正数或者负数。
10      有符号数的最高位也称为符号位，最高位为0表示正数，最高位为1表示负数。
```

5.5.1 short类型（短整型）

```
1       短整型在内存中占用2字节的空间，共计16bits，  [15:0]bits.
2       有符号数(short/signed short)：-32768 ~ 32767
3       无符号数(unsigned short)：0 ~ 65535
```

5.5.2 int类型(整型)

```
    整型在内存中占用4字节的空间，共计32bits，[31:0]bits.
    有符号数(int/signed int)：-2^31 ~ 2^31-1
    无符号数(unsigned int)：0 ~ 2^32-1
```

5.5.3 long类型(长整型)

```
32位操作系统：
    长整型在内存中占用4字节的空间，共计32bits，[31:0]bits.
    有符号数(long/signed long)：-2^31 ~ 2^31-1
    无符号数(unsigned long)：0 ~ 2^32-1

64位操作系统：
    长整型在内存中占用8字节的空间，共计64bits，[63:0]bits.
    有符号数(long/signed long)：-2^63 ~ 2^63-1
    无符号数(unsigned long)：0 ~ 2^64-1
```

5.5.4 long long类型(长长整型)

```
32位操作系统:
    长长整型在内存中占用8字节的空间，共计64bits，[63:0]bits.
    有符号数(long/signed long)：-2^63 ~ 2^63-1
    无符号数(unsigned long)：0 ~ 2^64-1

64位操作系统:
    长长整型在内存中占用8字节的空间，共计64bits，[63:0]bits.
    有符号数(long/signed long)：-2^63 ~ 2^63-1
    无符号数(unsigned long)：0 ~ 2^64-1
```

5.6 char(字符类型)

```
    字符类型在内存中占用1字节的空间，共计8bits，[7:0]bits.
    有符号字符类型(char/signed char) : -128 ~ 127
    无符号字符类型(unsigned char) : 0 ~ 255

    也可以使用ascii码表中的字符对字符类型的变量赋值。
```

6、明天的授课内容：

```
1．源码，反码，补码之间的转换
2．常量
3．变量
4．C语言的运算符
```

7、作业

```
吸收今天的内存，完成04练习题。
```