1. calloc和realloc
   1. calloc和malloc一样都是在堆区分配内存
   2. 不同点在于 calloc 会将分配的内存初始化为0
   3. realloc重新在堆区分配内存
      1. 如果分配的内存比原来要大，这个时候有两种情况出现
      2. 原有空间后序有足够大的空闲空间，那么直接在原有空间后继续开辟内存，返回原有空间的首地址
      3. 原有空间后序没有足够大空闲空间，重新会分配一个足够大的空间，并且将原有空间的内容拷贝到新空间下，释放原有空间，将新空间的首地址返回
2. sscanf的使用
   1. 将已知的字符串通过格式化匹配出有效信息

| **格式** | **作用** |
| --- | --- |
| %\*s或%\*d | 跳过数据 |
| %[width]s | 读指定宽度的数据 |
| %[a-z] | 匹配a到z中任意字符(尽可能多的匹配) |
| %[aBc] | 匹配a、B、c中一员，贪婪性 |
| %[^a] | 匹配非a的任意字符，贪婪性 |
| %[^a-z] | 表示读取除a-z以外的所有字符 |

* 1. 案例
     1. 匹配char \* ip = “127.0.0.1”将中间数字匹配到 num1 ~ num4中
     2. 字符串 char \* str = “abcde#zhangtao@12345” 中间的zhangtao匹配出来

1. 查找子串
   1. 实现自己的查找子串功能，需求在字符串中查找对应的子串，如果有返回字符串第一个字母的位置，如果没有返回-1
2. 指针的易错点
   1. 指针容易越界操作
   2. 指针叠加会不断改变指针指向
   3. 返回局部变量地址
   4. 不可以释放野指针
3. const的使用场景
   1. 用来修饰函数中的形参，防止误操作
4. 二级指针做函数参数的输入输出特性
   1. 输入特性
      1. 在主调函数分配内存，被调函数使用
      2. 在堆区创建
      3. 在栈上创建
   2. 输出特性
      1. 在被调函数中分配内存，主调函数使用
5. 二级指针练习-文件读写
   1. 需求：从文件中读取数据，并且将数据存放到堆区的数组中
      1. char \*\* pArray = malloc(sizeof(char \*) \* len )
   2. 获取有效行数
      1. 将文件光标置为文件首 fseek（file ， 0 ，SEEK\_SET）
   3. 读取文件数据并且放入到pArray中
   4. 显示数组pArray
   5. 释放数组pArray
6. 位运算
   1. 按位取反 ~ 0变1 1变1
   2. 按位与 & 全1为1 一0为0
   3. 按位或 | 全0为0 一1为1
   4. 按位异或 ^ 相同为0 不同为1
7. 移位运算符
   1. 左移 <<
      1. << X 等价于 乘以 2的X次方 ，用0填充位
   2. 右移 >>
      1. >> X 等价与 除以 2的X次方 ，如果有符号，不同的机器可能有不同的结果