- **单字符输入输出**
  int putchar(int c);
  向标准输出写一个字符
  返回写了几个字符，EOF(-1)表示写失败，EOF是c语言里定义的一个宏
  int getchar(void);
  从标准输入读入一个字符
  返回类型是int是为了返回EOF(-1)
  windows crtl-z ； unix crtl-d

```
int main(int argc,char const *argv[]){
    int ch;
    while((ch=getchar())!=EOF){
        putchar(ch);
    }
    printf("EOF\n");
    return 0;
}
```

输入crtl+c只是强制程序结束，并没有正确让其知道输入结束
当输入crtl+d时，我们读到了EOF状态
此外，我们输入很多字符，可是只有当按下回车，才会给输出，原因是
程序和用户键盘显示器之间有一个shell，负责运行程序，键盘输入的先给shell
你在键盘上键入的所有东西，shell会形成一个行编辑，也就是说你在键盘上输入的所有东西，在按下回车前，都没有被送到程序那里去
在shell给的缓冲区里

```
123回车234回车456回车\0
```

程序并没有结束，直到我们按了crtl+d,shell在看到crtl+d后会填入一个-1或者其他标志到末尾表明结束。
如果输入crtl+c，shell不会把其放入任何地方，该任何东西，而是直接将程序关闭

- **字符串数组**
  char **a
  表示a是一个指针，指向另外一个指针，而那个指针指向一个字符（串）
  char a[][]

```
char a[][] = {
    "hello",
};
```

编译器会报没有完备的类型，因为它希望第二维给出确切的大小，但是在给出具体大小，如a[][10],如果其中某个单元超过10了，编译器会报错

```
char *a[] = {
    "hello",
    "sjdjd",
};
```

这种情况下a[0]相当于char*

- **程序参数**
  int main(int argc,char const *argv[])
  argv[0]是命令本身，当使用unix的符号链接时，反应符号链接的名字，
  如./a.out,在unix可以改变这个指令，因此是有意义的
- **函数STRLEN**
  size_t strlen(const char *s)
  如果希望函数不修改传进去的数组就加上const

```
char line[] = "hello";
printf("strlen=%lu\n",strlen(line));
printf("sizeof=%lu\n",sizeof(line));
```

前者返回5，后者返回6，多的一个是结尾的\0

```
//写自己的strlen
size_t mylen(const char*s){
    int idx=0;
    while(a[idx]!='\0'){
        idx++;
    }

    return idx;
}
```

- **strcmp**
  int strcmp(const char* s1,const char* s2)
  比较两个字符串，返回0:s1 ==s2;1:s1>s2;-1:s1<s2
  比较两个数组不能用 ==直接比较，因为两个数组的地址永远不相等
  cmp给出的是差值

```
//自己写
int mycmp(const char* s1,const char* s2){
    while(*s1==*s2&&*s1!='\0'){
        s1++;
        s2++;
    }

    return *s1-*s2;
}
```

- **strcpy**
  char* strcpy(char *restrict dst,const char *restrict src); 把src的字符串拷贝到dst，restrict表明dst和src不重叠（c99），返回dst，为了能够链起代码来
  不重叠是为了适应现代计算机的多核

```
//复制一个字符串
char *dst = (char*)malloc(strlen(src)+1);
strcpy(dst,src);
```

```
//自己写strcpy
char *mycoy(char *dst,char *const src){
    char *ret = dst;
    while(*src!=0){
        *dst++ = *src++;
    }
    *dst = "\0";

    return ret;
}

改进
char *mycoy(char *dst,char *const src){
    char *ret = dst;
    while( *dst++ = *src++)
        ;
```

```
    *dst = "\0";

    return ret;
}
```

编译器会自动将代码变成最简洁的样子，因此不需要自己写成这样

- **字符串搜索函数**
  char *strchr(const char *c,int c);
  char *strrchr(const char *c,int c);
  返回NULL表示没有找到，
  从左边找，和从右边找
  如何寻找第二个相同字符

```
char s[] = "hello";
char *p = strchr(s,'l');
p = strchr(p+1,'l');
printf('%s\n',p);
```

复制某字符后面的部分

```
char s[] = "hello";
char *p = strchr(s,'l');
char *t = (char*)malloc(strlen(p)+1);
strcpy(t,p);
printf("%s\n",t);
free(t);
```

复制某字符前面的部分

```
char s[] = "hello";
char *p = strchr(s,'l');
char *c = p;
*p = 0;
char *t = (char*)malloc(strlen(s)+1);
strcpy(t,s);
printf("%s\n",t);
free(t);
```

char *strstr(const char *s1,const char *s2); char *strcasestr(const char *s1,const

char *s2); 忽略大小写
字符串中找字符串