printf 函数实现机制

软1101班　马语
201192466

1. printf函数简介

- printf 函数(格式化的输出函数〕
- 作用是向终端〔或系统隐含指定的输出设备〕，输出假设干个指定类型的数据。
- printf 函数的一般格式
- 一般格式：printf (格式控制，输出表列);

格式说明　　普通字符

- 格式说明：由“%”和格式字符组成，如%d，%f等
- 普通字符：原样输出的字符
- 输出列表：需要输出的数据，可以是常量、变量或表达式

1. printf函数简介

- 例如：
  - printf (“%d, %c”, i, c);
  - printf (“Hello World!”);
- 问题：

```c
#include<stdio.h>
int main()
{
    int i=3, j=4;
    printf("%d\n", i, j);
    return 0;
}
```

```
3
Press any key to continue
```

```c
#include<stdio.h>
int main()
{
    printf("%d %d %d %d\n");
    return 0;
}
```

```
0 0 2147348480 -858993460
Press any key to continue
```

1. printf函数简介

- 由于printf 是函数，因此“格式控制”字符串和“输出列表”实际上都是函数的参数。其一般的函数表示形式为：
  - **printf (参数1，参数2， 参数3，…，参数n)**

可变参数列表

- 需要解决的问题
  - 1.怎样让printf 函数知道传递了多少参数？
  - 2.printf 函数怎样访问这些参数？

2. 可变参函数

- 什么是可变参函数
  - 所谓含有变长参数的函数是指该函数可以接受可变数目的形参。
- 如何实现可变参函数
  - 三个宏和一个栈
    - **va_start**
    - **va_arg**
    - **va_end**

2. 可变参函数

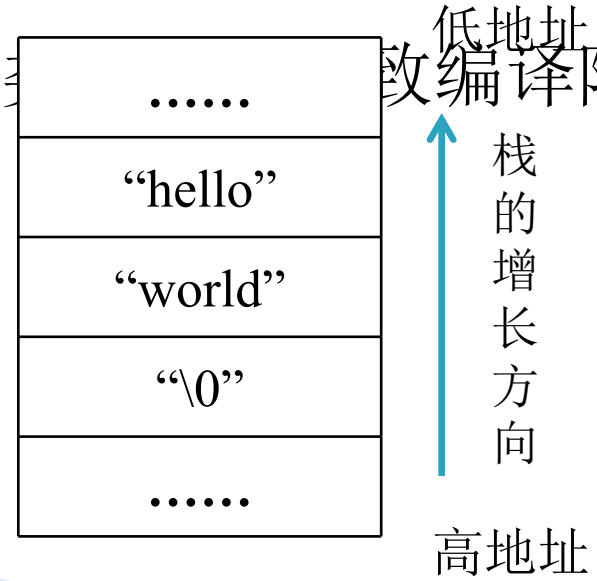
```
#include<stdio.h>
#include<stdarg.h>
#include<string.h>

void demo(char *msg,...)
{
    va_list argp;
    int arg_number=0;
    char *para = msg;
    va_start(argp,msg);
    while(1)
    {
        if ( strcmp( para, "\0") != 0 )
        {
            arg_number++;
            printf("parameter %d is: %s\n",arg_number,para);
        }
        else
            break;
        para = va_arg(argp,char *);
    }
    va_end(argp);
}

int main()
{
    demo("Hello","World","\0");
    return 0;
}
```

```
typedef char *  va_list;
```

定义于“stdarg.h”中，用来保存函数参数

2. 可变参函数

- 函数调用栈
- 对于c语言，它的调用规那么遵循_cdecl调用规那么：
- 1. 参数从右到左依次入栈
- 2. 调用者负责清理堆栈
- 3. 参数的数量和类型不匹配导致编译阶段的错误



2. 可变参函数

- 第一个宏：**va_start**

```
#define va_start(ap,v)  ( ap = (va_list)&v + _INTSIZEOF(v) )
```

  - 参数： ap 为va_list类型指针， v是最后一个确定的参数，最后一个确定的参数的含义是指它以后的参数都是可变参数。
  - 功能：获取参数列表中的第一个可变参数的内存地址，结果存放在ap中。
  - 注：_INTSIZEOF() 宏是为了实现地址对齐的

```
#define _INTSIZEOF(n)   ( (sizeof(n) + sizeof(int) - 1) & ~(sizeof(int) - 1) )
```

例如：sizeof(n) = 14B = 1110B， 1110 & FFFFFFFC = 1100 = 12

_INTSIZEOF(n) = (1110+0011) & FFFFFFFC = 10000 = 16

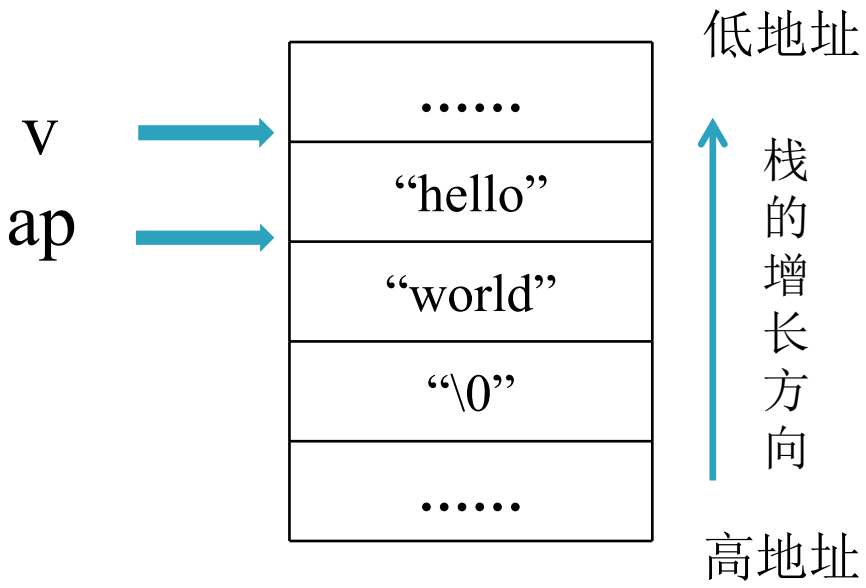
⬇

10001

2. 可变参函数

- 第一个宏：**va_start**

```
#define va_start(ap,v)  ( ap = (va_list)&v + _INTSIZEOF(v) )
```

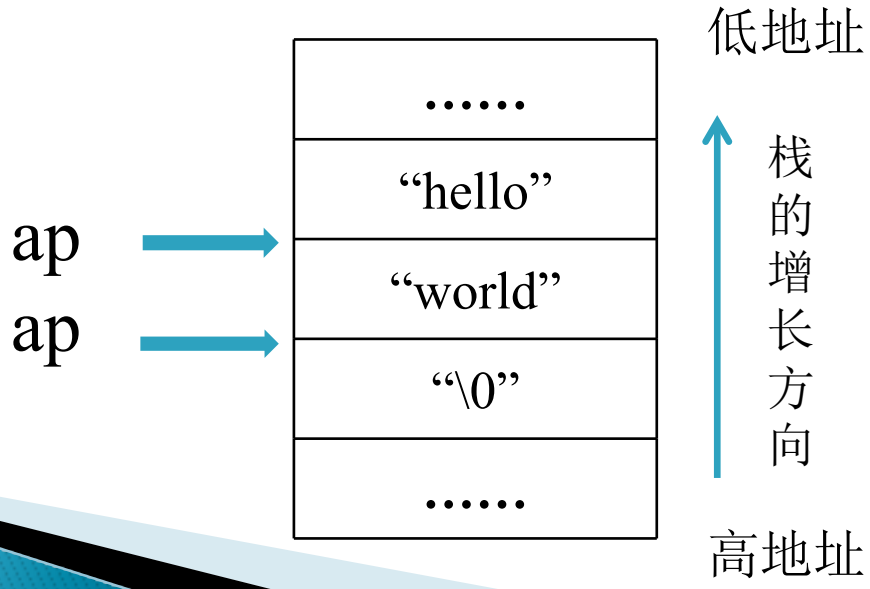


2. 可变参函数

- 第二个宏：**va_arg**

```
#define va_arg(ap,t)  ( *(t *)((ap += _INTSIZEOF(t)) - _INTSIZEOF(t)) )
```

  - 参数： ap 为va_list类型的指针，它指向当前需要获取的参数，t是当前参数的类型
  - 功能：获取ap当前所指向的参数的指针，并将其类型强制转化 t *，然后将ap指向下一个变长参数



2. 可变参函数

- 第三个宏：va_end

```
#define va_end(ap)  ( ap = (va_list)0 )
```

- 参数： ap 为va_list类型指针
- 功能： 将指针指向空，说明不再使用该指针

- 注：C标准要求在同一个函数中 va_start 和 va_end 要配对出现