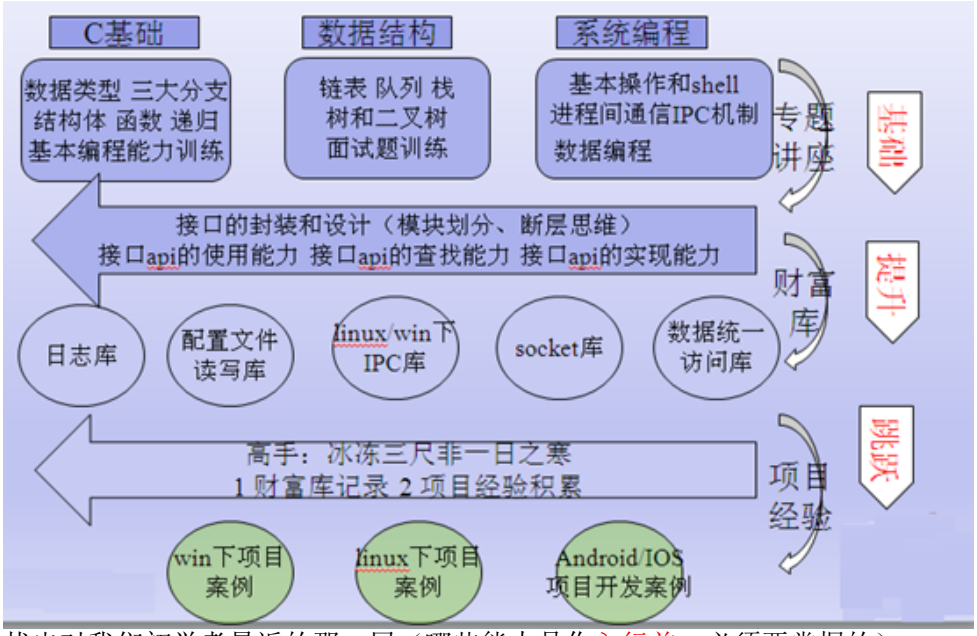
《传智播客 C语言就业班》 第一讲

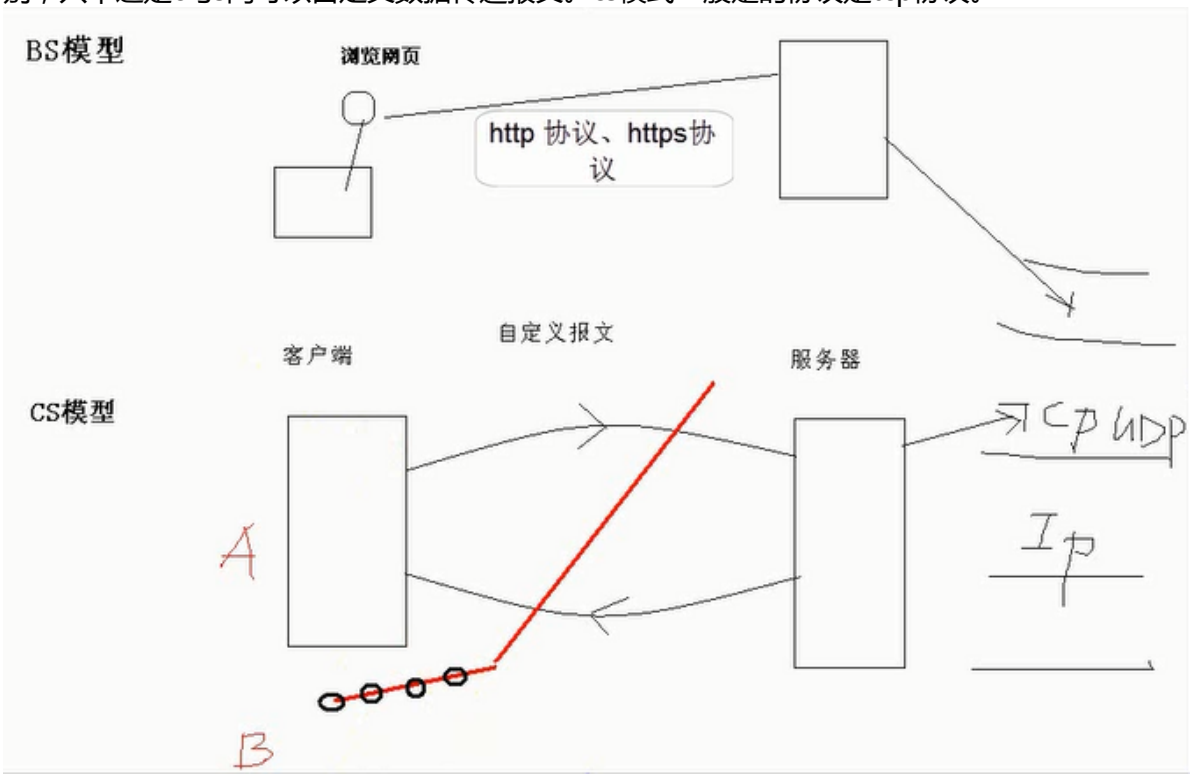
重要的思想： “**接口的封装和设计（模块划分、断层思维）**”
模块划分、业务模型抽象：**比如开发一套位于客户端的信息系统，A做上层应用（信息系统的业务模型），B做底层通讯接口。**



找出对我们初学者最近的那一层（哪些能力是你入行前，必须要掌握的）

bs模型与cs模型

现在的信息系统有两大类：bs系统和cs系统。
bs模式：如IE软件，客户端通过浏览器，浏览web服务器上的网页，这样的模型叫bs模型，b指客户端browser，s指服务端server。在客户端和浏览器端之间走的报文是http协议（即超文本传输协议）。
cs模型：客户端（client）发报文，服务器（server）收报文，服务器收到报文之后处理。这与bs模式没有很大区别，只不过是c与s间可以自定义数据传送报文。cs模式一般走的协议是tcp协议。



C项目开发的套路（一套接口）

```
#ifndef _SOCKETCLIENT_H
#define _SOCKETCLIENT_H   //VS2008：ctrl+u转换成小写，ctrl+shift+u转换成大写；
#ifdef _cplusplus   //为了在C++代码中调用用C写成的库文件，就需要用extern"C"来告诉编译器:这是一个用C写成的库文件，请用C的方式来链接它们
extern "C" {
#endif
    //第一套api函数
    //socket客户端环境初始化
    int socketclient_init(void **handle);
    //socket客户端报文发送
    int socketclient_send(void *handle, unsigned char *buf, int buflen);
    //socket客户端报文接收
    int socketclient_recv(void *handle, unsigned char *buf, int buflen);
    //socket客户端环境释放
    int socketclient_destroy(void *handle);
    //第二套api函数
    //socket客户端环境初始化
    int socketclient_init(void **handle);
    //socket客户端报文发送
    int socketclient_send(void *handle, unsigned char *buf, int buflen);
    //socket客户端报文接收
    int socketclient_recv(void *handle, unsigned char **buf, int buflen);
    //socket客户端环境释放
    int socketclient_destroy(void **handle);
    //技术点分析：1级指针、2级指针
    //void **handle类型封装的概念 业务模型的封装
#ifdef _cplusplus
}
#endif
#endif
```

总结：寻找到学习的标准

需要的能力：

接口的封装和设计（功能抽象和封装）：

- 接口api的使用能力（函数调用）
- 接口api的查找能力（快速上手）
- 接口api的实现能力

建立正确程序运行内存布局图：

- **内存四区模型**图
- **函数调用模型**图

课程大纲
    * C提高
    * C++
    * 数据结构
    * 总体时间1个月

C/C++学习特点
    * Java：学习、应用、做项目
    * C：学习、理解、应用、做项目

C语言学到什么程度，就可以听懂传智播客就业班第一阶段的课程了。有没有一个标准？
* 选择排序或冒泡排序
    * 在一个函数内排序
    * 通过函数调用的方式排序
    * 数组做函数参数的技术盲点和推演

数组做函数参数会退化为指针！如下：

```
#include "stdio.h"

void sortArray(int a[], int num);
void printArray(int a[], int num);

//  排序
void main()
{
	int num = 0;
	int a[] = { 3,1,2,4,5,7,6 };
	num = sizeof(a) / sizeof(a[0]); //由于是数组，num=7
	printArray(a, num);
	sortArray(a, num);
	printf("\n");
	printArray(a, num);
}

// 数组做函数参数会退化为指针！！下面num2=1 ！！
// 分配4个字节的内存而不是4*7字节的内存
// 1 结论：把数组的内存首地址和数组的有效长度传给被调函数
// 2 实参的a和形参的a的数据类型不一样
// 形参中的数组，编译器会把它当成指针处理，这是C语言的特色
// C语言的优势在于，可以通过指针，在主调函数和被调函数之间操作内存
void sortArray(int a[], int num) //等价于void sortArray(int *a, int num)
{
	int i, j, temp;
	int num2;
	num2 = sizeof(a) / sizeof(a[0]);   //num2=1 ！！
	for (i = 0; i < num; i++)
	{
		for (j = i + 1; j < 7; j++)
		{
			if (a[j] < a[i])
			{
				int temp = a[i];
				a[i] = a[j];
				a[j] = temp;
			}
		}
	}
}

void printArray(int a[], int num)
{
	int i;
	for (i = 0; i < num; i++)
	{
		printf("%d ", a[i]);
	}
}
```

内存四区专题讲座（堆区、栈区、全局区、代码区）

数据类型本质分析——是固定大小内存块的别名。要站在编译器的角度去理解。

```
#include "stdio.h"

struct Teacher
{
	char name[64];
	int age;
}Teacher;

typedef struct Teacher2
{
	char name[64];
	int age;
}Teacher2; //把结构体数据类型重命名
```

```
void main()
{
    //b &b 数组数据类型定义一个（数组类型2.数组指针3.数组类型和数组指针类型的关系）
    //-----------------------------
    //定义数据类型
    int a; //告诉C编译器分配4个字节的内存
    int b[10];  //分配40个字节的内存
    struct Teacher t1;  //必须加struct
    Teacher2 t2;       //如果把struct数据类型typedef重命名，则可以省略struct（C语言）
    t1.age = 31;
    t2.age = 30;
    printf("%d\n", sizeof(a));
    printf("%d %d %d %d \n", b, b + 1, &b, &b + 1);
    // b和&b所代表的数据类型不一样！！
    // b代表数组首元素的地址
    // &b代表整个数组的地址
}
```

数据类型的封装

void* 代表无类型指针，可以指向任何类型的数据；
如内存操作函数的函数原型

```
void* memcpy(void* dest, const void *src, size_t len);
void* memset(void* buffer, int c, size_t num);
```

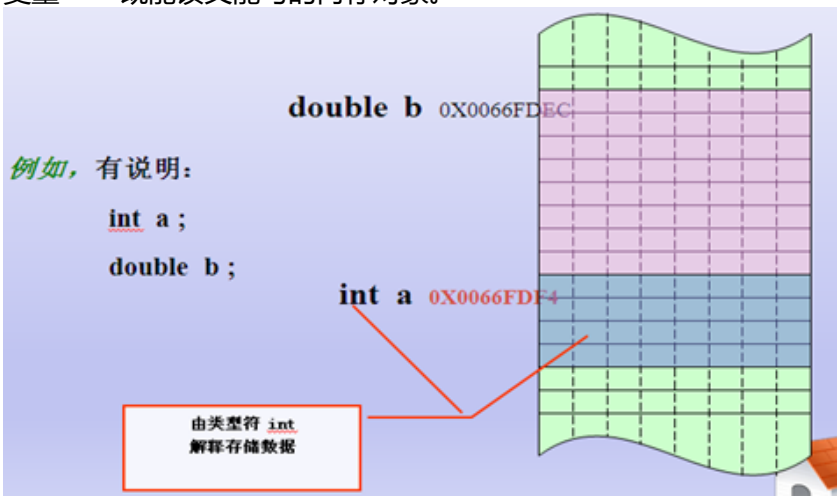
C语言规定只有相同类型的指针才可以相互赋值。
void* 指针作为左值用于“接收”任意类型的指针；
void* 指针作为右值给其他指针时需要强制类型转换！

```
char* p2=NULL;
p2 = (char*)(malloc(sizeof(char)*100))
void* p1=NULL;
p1 = &p2;
```

不存在void变量的类型，如void a错误，编译器不知道如何分配内存。

变量本质分析

变量——既能读又能写的内存对象。

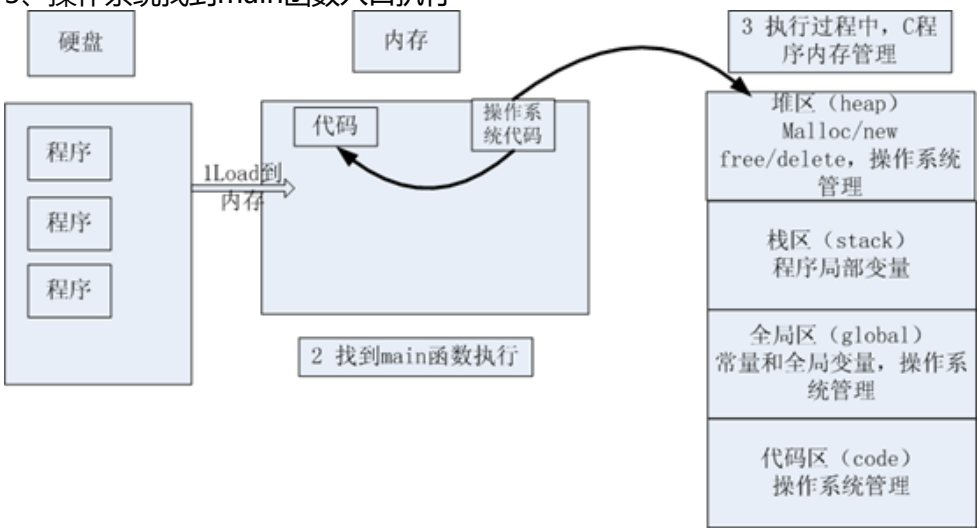


变量的本质 —— （一段连续）**内存空间的别名（是一个“门牌号”，变量是内存的标号；画图时不要把标号画到内存里面），通过变量往内存读写数据，而不是向变量读写数据！ 变量跑到了代码区；**

内存四区的建立流程

流程说明
1、操作系统把物理硬盘代码load到内存

2、操作系统把c代码分成四个区
3、操作系统找到main函数入口执行



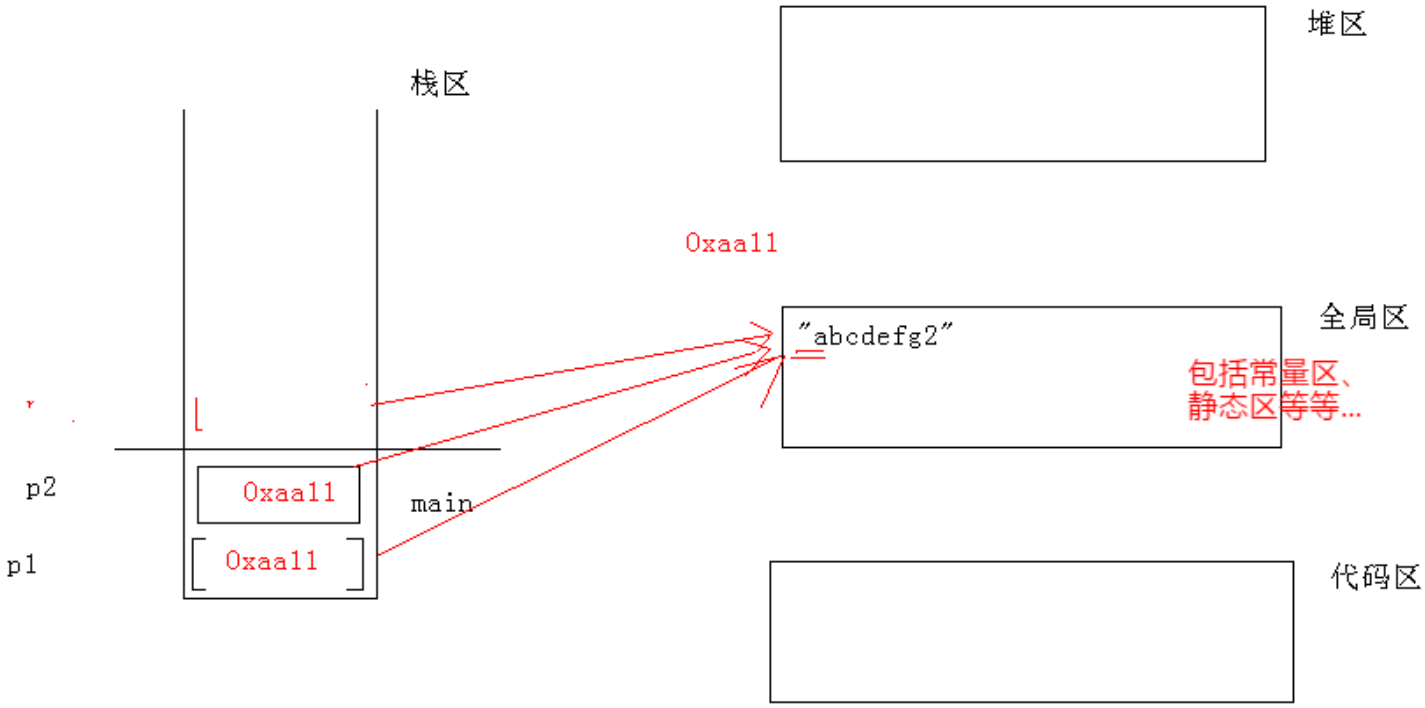
各区元素分析

栈区（stack）：由编译器自动分配释放，存放函数的参数值，局部变量的值等。
堆区（heap）：一般由程序员分配释放（动态内存申请与释放），若程序员不释放，程序结束时可能由操作系统回收。
全局区（静态区）（static）：全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域，未初始化的全局变量和未初始化的静态变量在相邻的另一块区域，该区域在程序结束后由操作系统释放。
常量区：字符串常量和其他常量的存储位置，程序结束后由操作系统释放。
程序代码区：存放函数体的二进制代码。

变量的三要素：名称、大小、作用域（函数的三要素：名称、参数、返回值）；
变量的生命周期如何管理？

注意：下面代码如果改成 char* p1 = abcd2，那么打印p1、p2的地址值会发现一样！！因为全局区由操作系统管理，调用完getStr1()后，全局区的内存空间并不会析构掉

```
#include "stdio.h"
char* getStr1()
{
    char* p1 = "abcd1";
    //注意：如果改成abcd2，那么打印p1、p2的地址值会发现一样！！因为
    //全局区由操作系统管理，调用完getStr1()后，全局区的内存空间并不会析构掉
    //C++编译器会做代码优化，全局区只保留一份"abcd2"，因此也就只有一个地址
    return p1;
}
char* getStr2()
{
    char* p2 = "abcd2";
    return p2;
}
int main()
{
    char *p1 = NULL;
    char *p2 = NULL;
    p1 = getStr1();
    p2 = getStr2();
    //打印p1、p2所指向的内存空间的数据
    printf("p1：%s,p2：%s \n", p1, p2);
    //打印p1、p2的值
    printf("p1：%d,p2：%d \n", p1, p2);
    return 0;
}
```



1 指针指向谁 就把谁的地址赋给指针

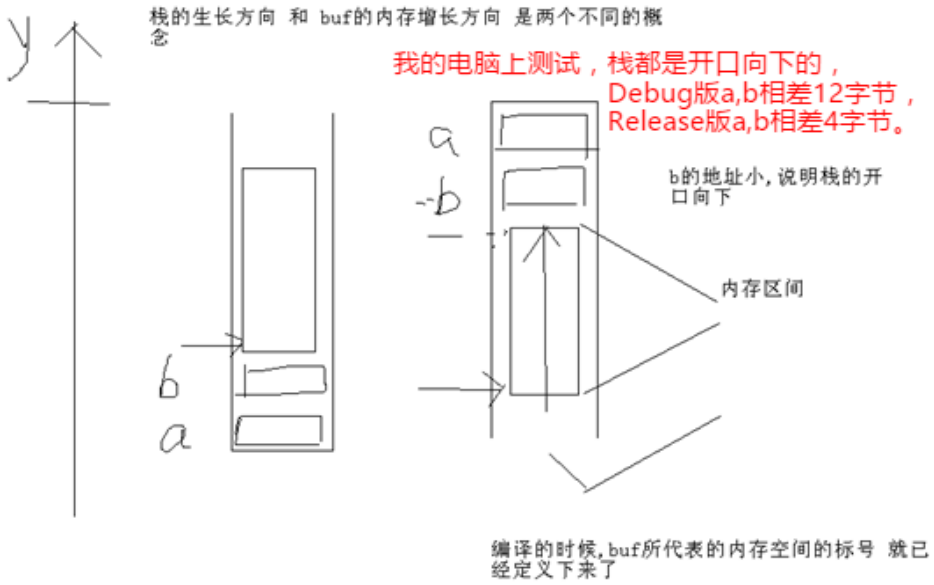
2 指针变量 和 他所指向的内存空间变量是两个不同的概念

3 理解指针的关键在于内存！！！没有内存何来指针！

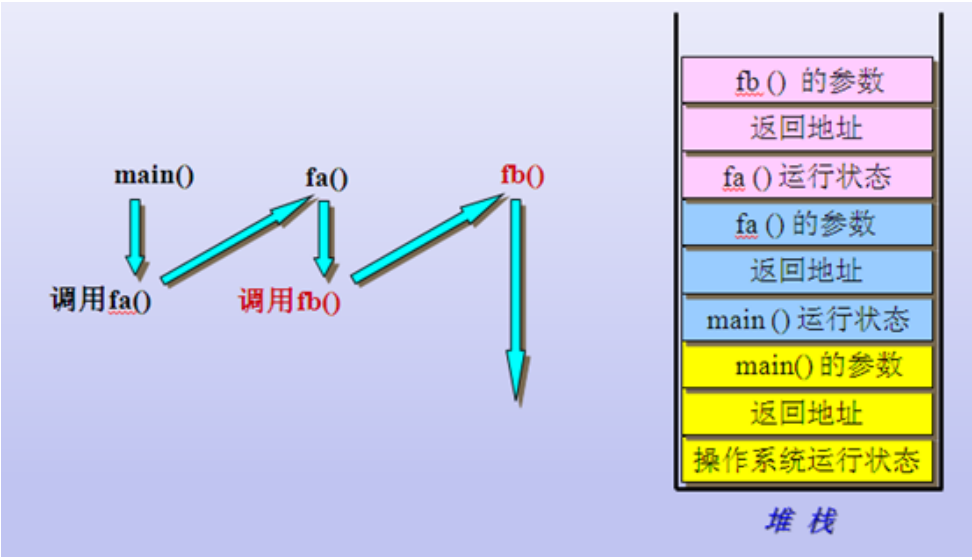
```
#define _CRT_SECURE_NO_WARNINGS
#include "stdio.h"
#include "stdlib.h"
#include "string.h"

//堆
char* getMem(int num)
{
	char *p1 = NULL;
	p1 = (char*)malloc(sizeof(char)*num);
	if (p1 == NULL)
	{
		return NULL;
	}
	return p1;
}

//栈
char* getMem2()
{
	char buf[64]; //临时变量、栈区存放
	strcpy(buf, "123456789");
	return buf; //错误！Debug编译失效(系统会自动调用析构函数，释放掉栈空间)
			//虽然Release编译有效，但程序是不稳定的；
}

int main()
{
	char *tmp = NULL;
	tmp = getMem(10);
	if (tmp == NULL)
	{
		return;
	}
	strcpy(tmp, "111222");  //向tmp所指的内存空间中copy数据
	printf("hello..tmp : %s \n", tmp);
	tmp = getMem2();
	printf("hello..tmp : %s \n", tmp);
}
```

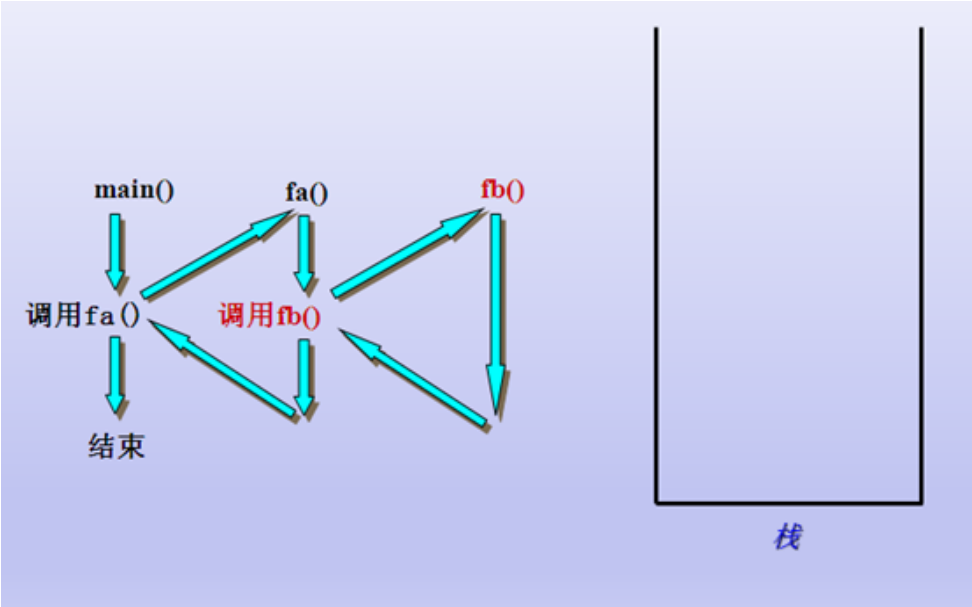
一般情况认为，栈的开口向下（避免栈的溢出）；但是char buf[128]，buf的内存地址buf+1永远是向上的。

- **“栈的生长方向” 和 “buf的内存增长方向” 是两个概念**，不要混淆；
- **堆的生长方向向上**；



函数调用





main函数中可以在栈、堆或全局区分配内存；可以被被调用函数fa，fb使用。
fb函数申请的内存？——在栈上分配的内存，（调用完析构）不能被fa和main函数使用；
但**fb中的malloc内存（堆），可以被fa和main函数使用**；在全局区分配的内存如"abcdefg"内存，可以被fa和main函数使用。
总结：主调函数分配的内存可以传给被调用函数使用；在被调用函数里面分配的内存，堆上分配的内存可以被主调函数使用，栈上的不可以；主调函数如何使用被调函数分配的内存——指针做函数参数。