Linux编程笔记

函数调用栈空间的分配与释放

> 1. 函数执行的时候有自己的临时栈，C++函数执行的时候有两个内存空间，一个是函数本身的临时栈空间，一个是原来栈空间的对象。C语言的只有一个自己的栈空间。
> 2. 函数的参数就在临时栈中。如果函数传递实参，则用来初始化临时的参数变量。
> 3. 通过寄存器返回值。（使用返回值返回数据）
> 4. 通过参数返回值。（参数必须是指针）指针指向的区域必须事先分配。
> 5. 如果参数返回指针，那么参数就是双指针。

C和C++的主要区别的地方：

- bool类型
- **引用**
- **模板**
- 命名空间
- **异常处理**
- 面向对象

函数前面加的编译属性 *_stdcall \fastcall* _cdecl

> 1. 决定函数栈压栈的参数顺序。（三个都是从右到左。）
> 2. 决定了函数栈的清空方式。
>
>   stdcall：调用者负责清空函数栈，函数本身不负责清空临时栈。
>   fastcall:调用者自己清空栈，然后返回值。
>   cdecl:每一个调用者都有自己专门的清空参数。
>
> 3. 决定了函数的名字转换方式。

虚拟内存

一个程序不能访问另外一个程序的地址指向的空间。

理解：

1. 每个程序的开始地址都是0x80084000
2. 程序中使用的地址不是物理地址，而是逻辑地址（虚拟内存），逻辑地址仅仅是编号，编号使用int4字节整数表示（494967296）。每一个程序都提供了4G的访问能力。

逻辑地址与物理地址关联才有意义：过程本身称为内存映射

背景：

虚拟内存的提出是禁止用户直接访问物理存储地址，有助于系统的稳定。

结论：

虚拟地址与物理地址在映射的时候有一个基本单位，为4k,16进制为1000，（内存页）。

段错误：无效访问。

合法访问：malloc分配的空间之外的空间可以访问，但访问非法。能够访问，不意味着合法。

虚拟地址

linux中的虚拟内存空间分为栈和堆

栈：编译器自动生成代码维护

栈：编译器自动生成代码维护

堆：地址是否映射，映射的空间是否被管理。

1. brk/sbrk 内存映射函数

```
分配释放内存
int brk(void *end);//分配空间，释放空间
void *sbrk(int size);//返回空间地址
```

应用：

```
1.使用sbrk分配空间。
2.使用sbrk得到没有映射的虚拟地址
3.使用brk分配空间
4.使用brk释放空间
```

理解

```
+ sbrk(int size) 如果是第一次运行，则返回没有映射的空闲空间，同时产生一个数据：指向地址。sbrk与brk后台系统维护一个指针，指针默认是null,调用sbrk,判定指针是否是0，是的话，系统得到大块的空闲空间的首地址初始化。同时把指针+size。如果是否，返回指针，并且把指针位置+size。
```

补充知识点：帮助手册 man 节（1-8） 关键字 其中的1是linux的系统指令，2是系统函数，3是标准C函数的文档，7是系统的编程帮助。

```
#include <stdio.h>
#include <unistd.h>
int isPrimer(int a)
{
    int r = 0;
    int i;
    for(i = 2;i<a;i++)
    {
        if(a%i==0)
        {
            r = 1;
            return r;
        }
    }
    return 0;
}

main()
{
    int i = 2;
    int b;
    int *r;
    r = sbrk(0);
    for(;i<10000;i++)
    {
        b = isPrimer(i);
        if(b==0)
        {
            brk(4);
            *r = i;
            r = sbrk(0);
        }
    }
    i = 0;
    r = p;
    while(r!=sbrk(0))
    {
        printf("%d\n",*r);
        r++;
    }
    brk(p);
}
```

总结：new malloc brk/sbrk 智能指针 STL

异常处理

异常处理：

- int brk(void*) void *sbrk(int) 如果成功，brk返回0，sbrk返回指针。如果失败，brk返回-1，sbrk返回（void*）-1