1. 函数栈帧

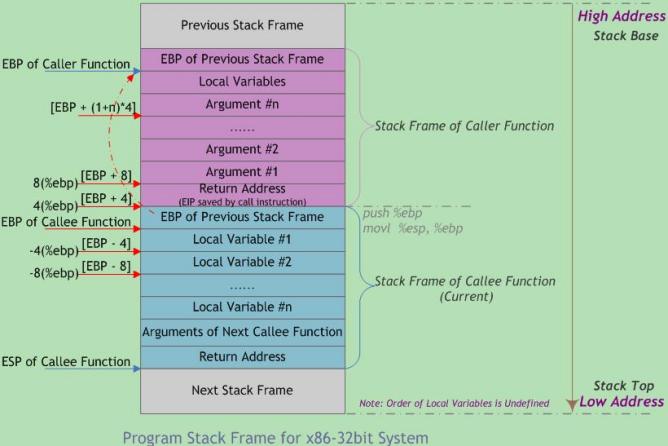
https://blog.csdn.net/stay\_the\_course/article/details/53044575?utm\_medium=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control&dist\_request\_id=1328741.28677.16169280538972775&depth\_1-utm\_source=distribute.pc\_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-3.control

https://www.cnblogs.com/clover-toeic/p/3755401.html

函数栈帧

起始：CPU寄存器的EBP（栈底指针）

结束：CPU寄存器的ESP（栈顶指针），可以根据ESP偏移定位每个变量，但ESP会在函数执行的时候随着变量的压栈与出栈而变动



A）Caller：主调函数

B）Callee：被调函数

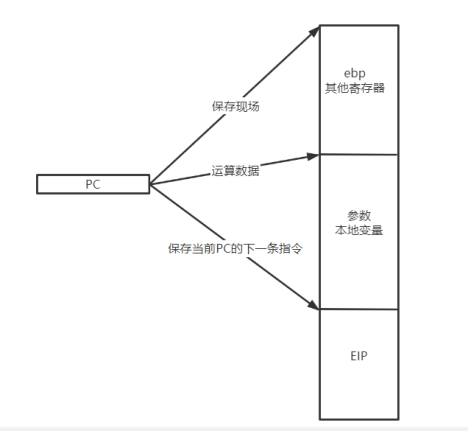
C）明确：

函数首先要有执行空间（空间：esp、ebp）

其次要有数据（寄存器、变量、参数 ）与指令（指令：pc）

1. 每个栈帧：

源头：以下是根据PC指令不断创建出来的，esp不断移动



（1）保存上个函数现场：

ebp、其他寄存器

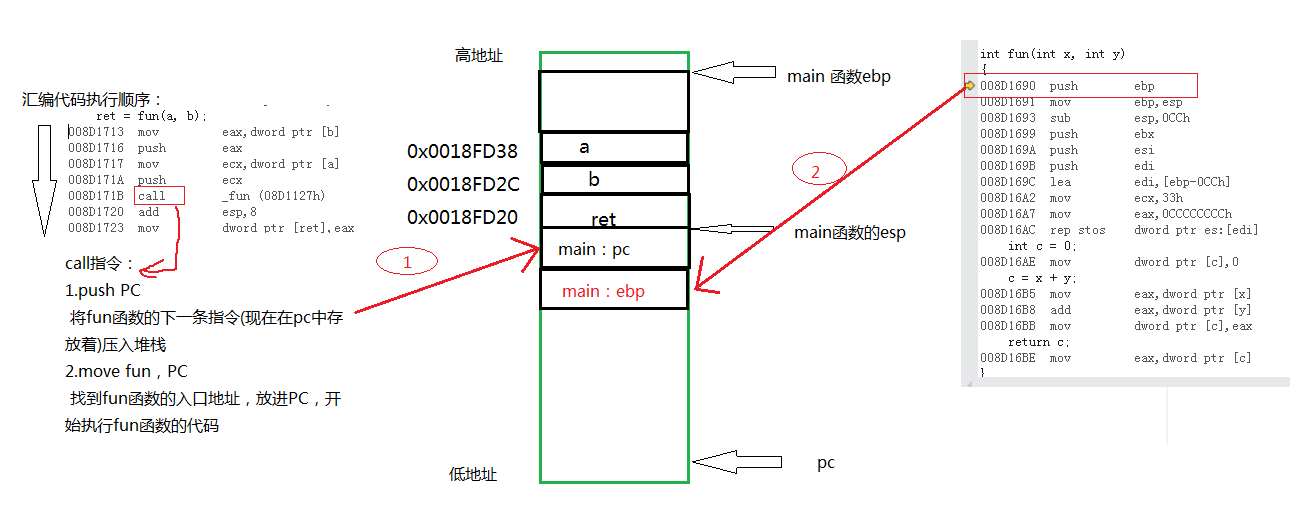
esp不用保存，因为上一个函数的esp与当前函数的ebp是邻居

1. 执行函数内容：

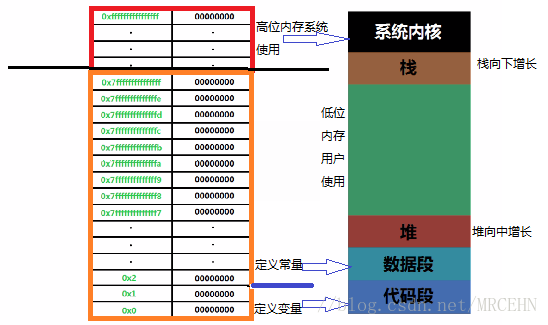
参数、本地变量

1. 保存当前函数下一条指令（返回指令）：

如果调用函数，则需要保存返回调用函数后的执行指令（EIP下一条指令寄存器）

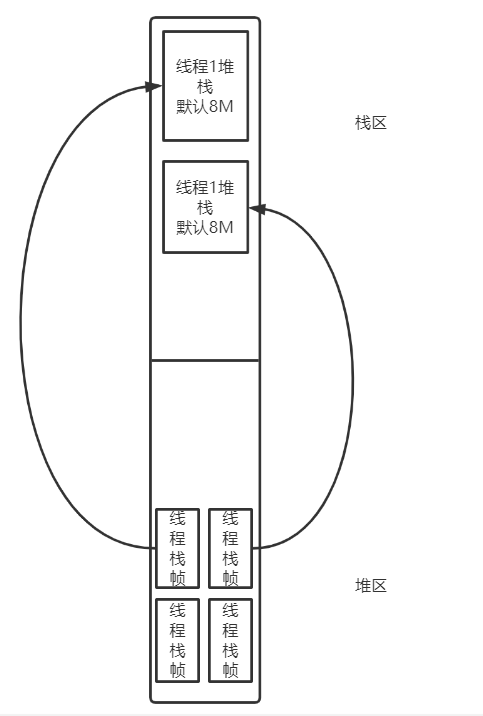


1. 进程栈帧



1. 线程栈帧

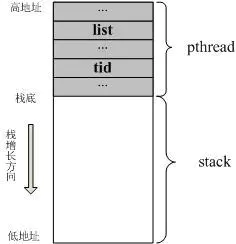
（1）多线程：每线程一个栈帧



1. 每个线程栈内容：

https://www.jianshu.com/p/ea692d4f5e27

struct pthread 是用于用户态描述线程的数据结构，那么显然每个 pthread 都唯一对应一个线程。那么这个变量是存储在哪里的，答案是线程栈内存块的高地址空间中的（这里以 x86 栈向下增长的方式为例）。也就是说，创建线程时为每个线程分配了一块内存，然后这块内存一部分存储了 pthread 变量，剩下的内存才是真正的线程栈。



<https://blog.csdn.net/elfprincexu/article/details/78779158> 测试线程堆栈大小

https://blog.csdn.net/qq\_33921804/article/details/57403414

（1）内容：

线程包含了表示进程内执行环境必需的信息，其中包括进程中标示线程的线程ID，一组寄存器值，栈，调度优先级和策略， 信号屏蔽字，errno变量以及线程私有数据。

1.线程ID  
      每个线程都有自己的线程ID，这个ID在本进程中是唯一的。进程用此来标

   识线程。

    2.寄存器组的值  
       由于线程间是并发运行的，每个线程有自己不同的运行线索，当从一个线

   程切换到另一个线程上 时，必须将原有的线程的寄存器集合的状态保存，以便

   将来该线程在被重新切换到时能得以恢复。

    3.线程的堆栈  
       堆栈是保证线程独立运行所必须的。  
       线程函数可以调用函数，而被调用函数中又是可以层层嵌套的，所以线程

   必须拥有自己的函数堆栈， 使得函数调用可以正常执行，不受其他线程的影

   响。  
    4.错误返回码  
       由于同一个进程中有很多个线程在同时运行，可能某个线程进行系统调用

   后设置了errno值，而在该 线程还没有处理这个错误，另外一个线程就在此时

   被调度器投入运行，这样错误值就有可能被修改。  
       所以，不同的线程应该拥有自己的错误返回码变量。  
    5.线程的信号屏蔽码  
       由于每个线程所感兴趣的信号不同，所以线程的信号屏蔽码应该由线程自己管理。但所有的线程都共享同样的信号处理器。  
    6.线程的优先级  
       由于线程需要像进程那样能够被调度，那么就必须要有可供调度使用的参数，这个参数就是线程的优先级。

每个线程都有自己的堆栈空间，但是线程堆栈的地址是按照进程地址空间统一编址的，所以每个线程堆栈地址空间是不会重复的，即：每个线程堆栈的地址空间不是从0开始编址，而是按整个进程地址空间统一编址的。

（2）大小：

可将线程栈的空间管理交给系统，如果想改变系统默认的栈大小8MB，可以通过

int pthread\_attr\_setstacksize(pthread\_attr\_t \*attr, size\_t stacksize);

// 注：stacksize最小值为16384，单位为字节

由上面的API接口，可以得到，线程栈的stacksize是保存在pthread\_attr\_t中的，可以通过人为的指定，也可以通过在创建线程的时候读取系统的配置文件来初始化stacksize，当初始化完栈的起始地址，和大小后，便可以通过

****（3）总结：****

一个标准的线程由线程ID，当前指令指针(PC），寄存器集合和堆栈组成。

栈：是个线程独有的，保存其运行状态和局部自动变量的。栈在线程开始的时候初始化，每个线程的栈互相独立，因此，栈是　thread safe的。操作系统在切换线程的时候会自动的切换栈，就是切换　ＳＳ／ＥＳＰ寄存器。栈空间不需要在高级语言里面显式的分配和释放。

堆：　是大家共有的空间，分全局堆和局部堆。全局堆就是所有没有分配的空间，局部堆就是用户分配的空间。堆在操作系统对进程初始化的时候分配，运行过程中也可以向系统要额外的堆，但是记得用完了要还给操作系统，要不然就是内存泄漏。

1. 进程调度

通过PCB（在内核中，存储在链表）

1. 线程调度

通过TCB（挂在PCB后面）

