建议先看这个：[【Linux学习笔记】栈与函数调用惯例—下篇](http://blog.csdn.net/slvher/article/details/8831983)

<http://blog.csdn.net/slvher/article/details/8831983>

若函数P（调用者caller）调用函数Q（被调用者，callee），则Q的参数存放在P的栈帧中（在对应的汇编代码中，表现为在跳转至callee startaddress之前，通过push指令将函数Q所需的参数依次压入栈中，压栈顺序与本文主题—函数调用约定有关，具体实现过程大家看到后面的汇编代码就会清楚）。另外，当P调用Q时，P中的返回地址被压入栈中（call指令实现返回地址压栈并跳转至callee入口地址处），形成P的栈帧末尾。返回地址其实就是P中调用Q的指令执行完后下一条将要执行的指令地址。接着，需要保存P的帧指针（用于从Q返回时恢复P的栈帧结构）并将P栈帧当前的栈顶地址值（存放在%esp中，是P的栈帧边界之一）装入帧指针寄存器%ebp，Q的栈帧边界即从该%ebp开始，可见，经过这样的操作流程，当前的%ebp指向的位置既是P栈帧的结束边界，又是Q栈帧的开始边界。接着是保存其它寄存器的值。最后开始真正执行函数Q包含的功能指令。

P{

Q(q1,q2);

}

q2

q1

P的返回地址

P的帧指针ebp

再看这个：

<http://learn.akae.cn/media/ch19s01.html>

<http://blog.csdn.net/jackyding26791/article/details/7390331>