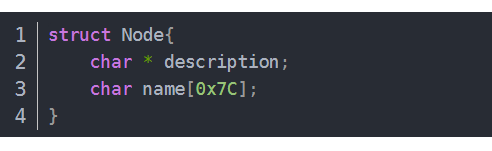
首先说一下这道题也是本地不通，远程打通，换一下libc就行了，注意要在pwntools里面换，直接用patchelf不行---（**搞错了**原来patchelf后失败是因为在pwntools脚本里面还用的是原来的那个二进制文件process，换为patchelf后的就行了……）

这题本身倒不是很难，主要是edit里面那个检查长度的if判断一开始总是不理解是什么意思

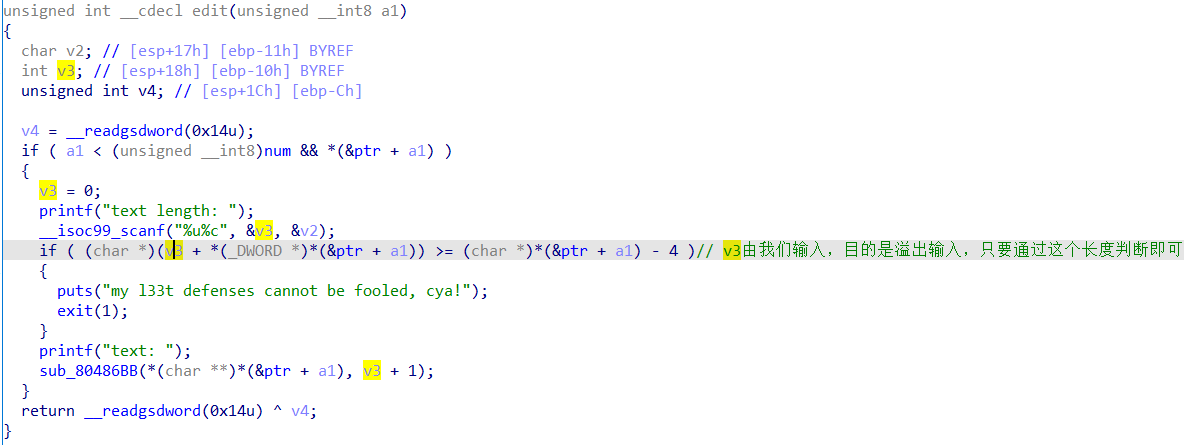
先给一下结构体



Ptr里面存的是node的指针

然后就是关键的判断语句也是漏洞语句了，在edit函数里面

注意在add函数里面已经分配好了description的大小，edit是给定一个输入长度v3然后对description填充，我们的目的是堆溢出，所以就是要通过v3溢出，而v3要先通过一个if判断，我就是卡在了这里不理解是什么意思



其检查数据长度的判断其实是以description堆块的起始地址和name的起始地址之间的长度。注意一个很关键的点，（char\*）这种只是改变数据类型不改变数据值

所以

(char \*)\*(&ptr + a1)代表取ptr[a1]指向的数值，即Node的地址

(char \*)(v3 + \*(\_DWORD \*)\*(&ptr + a1))则是在ptr[a1]的基础上又取了其指向的地址值，即description的地址+v3，最后还有个-4代表Node所在chunk的开头，因为Node本身也是一个chunk，所以Node的地址前面还有一个pre\_size和size合起来是4个字节，所以综上所述，判断的意思就是从description的地址开始读入v3个字节不能覆盖到对应Node所在chunk，即使是pre\_size和size也不能覆盖。

（注：这里有个重要前提：因为有description的分配在Node之前，所以其chunk也在Node chunk相邻上部）

看到这里应该就理解本题的堆溢出是怎么实现的了，这里不赘述了，直接贴一张师傅博客的图

