**Lab2实验报告**

1. **练习0:填写已有实验**

利用Understand中的Compare完成此练习。

1. **练习1:实现firstfit连续物理内存分配算法**

**大体思路：**

物理内存页管理器顺着双向链表进行搜索空闲内存区域，直到找到一个足够大的空闲区域，这是一种速度很快的算法，因为它尽可能少地搜索链表。如果空闲区域的大小和申请分配的大小正好一样，则把这个空闲区域分配出去，成功返回;否则将该空闲区分为两部分，一部分区域与申请分配的大小相等，把它分配出去，剩下的一部分区域形成新的空闲区。其释放内存的设计思路很简单，只需把这块区域重新放回双向链表中即可。

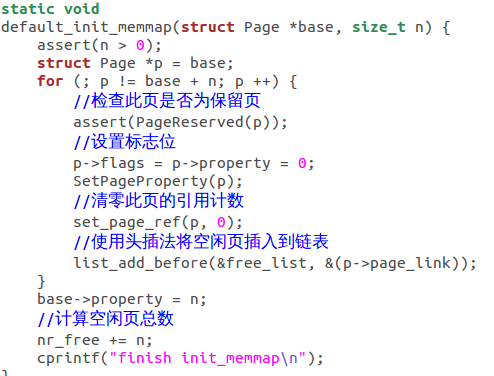
**实现目标：**

重写default\_init\_memmap()，default\_alloc\_pages()，default\_free\_pages()函数。

**具体细节：**

1. default\_init\_memmap()函数

这个函数是用来初始化空闲页链表的。主要有两个步骤：初始化每一个空闲页，计算空闲页的总数。



注意：1.使用头插法是因为地址是从低地址向高地址增长。

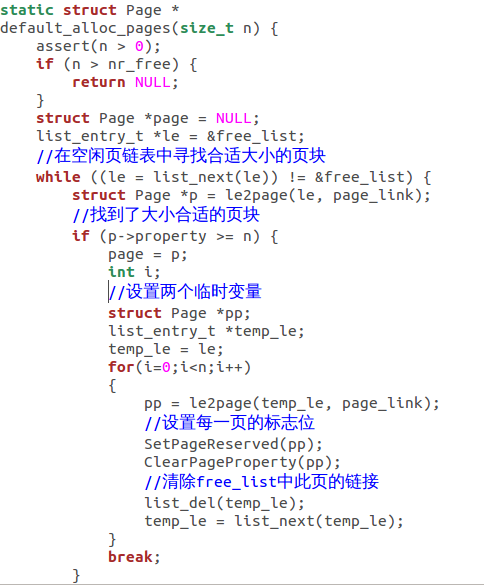
2.p->flags = 0语句已经将PG\_reserved标志位置零。

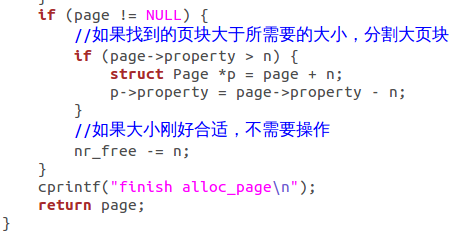
1. default\_alloc\_pages()函数

这个函数是用来分配空闲页的。

主要步骤如下：

1. 寻找足够大的空闲块
   1. 如果找到了，重新设置标志位
   2. 从空闲链表中删除此页
   3. 判断空闲块大小是否合适
      1. 如果不合适，分割页块
      2. 如果合适则不进行操作
   4. 计算剩余空闲页个数
   5. 返回分配的页块地址





备注：在参考答案中，我认为有些语句是冗余的，如图：



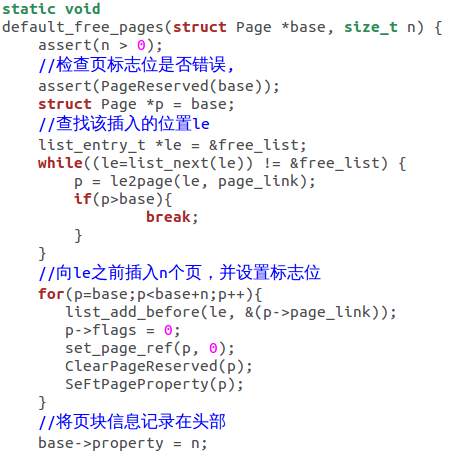
**验证：**在第二次重置标志位前后，分别输出标志位的值，发现，flags并没有发生变化。然后将这两句话注释，编译后运行，依旧可以得到正确答案。所以我认为这两句话是没有必要的。

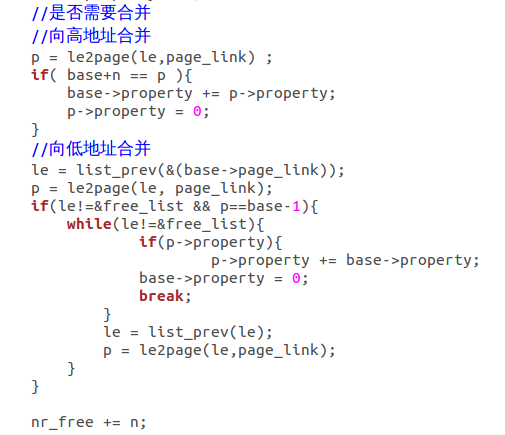
1. default\_free\_pages()函数

这个函数的作用是释放已经使用完的页，把他们合并到freelist中。

具体步骤如下：

1. 在freelist中查找合适的位置以供插入
2. 改变被释放页的标志位，以及头部的计数器
3. 尝试在freelist中向高地址或低地址合并

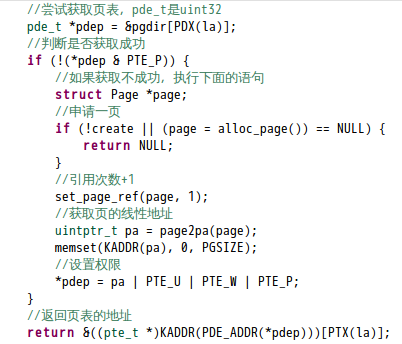




注意：在合并后，需要更新头部的页块大小的值即property。

1. **练习2:实现寻找虚拟地址对应的表项**

**大体思路：**尝试获取页表的地址，如果获取不到就新建一个页表。



1. **练习3:释放某虚地址所在的页并取消对应二级页表的映射**

**大体思路：**判断此页被引用的次数，如果仅仅被引用一次，则这个页也可以被释放。否则，只能释放页表入口。

