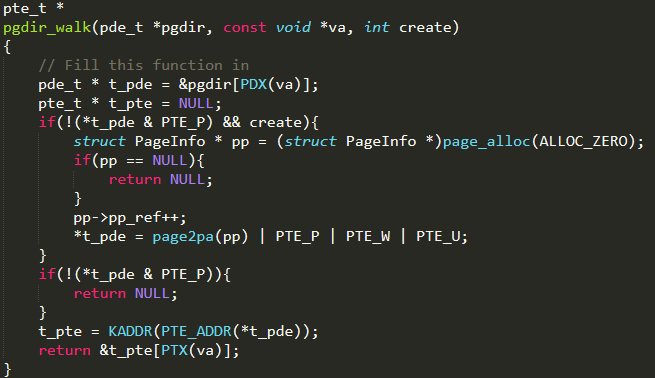
**PartB**

PartB要求实现虚拟内存的映射，即实现把对应的空间映射到PartA完成的页目录中，具体需要实现六个函数，实现思路如下所示：

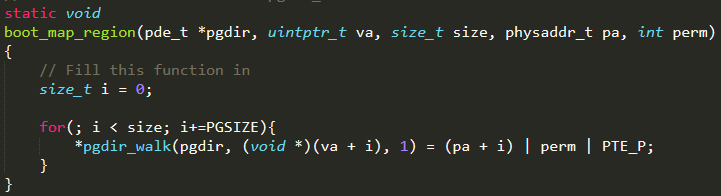
* pgdir\_walk()

给一个页目录，要求按两级页表给出虚拟地址va的页地址，如果找到的页不存在且需要创建则按要求完成。PDX可以找出虚拟地址对应的页表目录偏移，在页表目录中找应的页表项，如果页表项不存在且需要创建一个空页，将该页表链在页表目录中，利用page2pa可以获取一个页对应的物理地址，需要初始化对应的信息位。之后从该页表目录项（即一个页表地址加上其信息位）中提取除页表的地址，并返回该页表中对应的页的条目，具体代码如下所示：



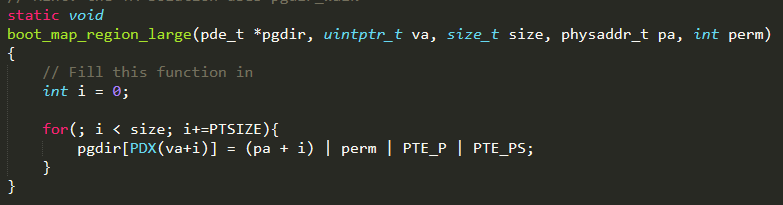
* boot\_map\_region()

该函数实现对页的一段虚拟地址到物理地址的映射，每次映射的单位是一个页的大小，需要借助pgdir\_walk完成。实际上是一个循环赋值过程，等式右边是物理地址，等式左边是对应的虚拟地址（并不是va，而是va对应的页的地址）。已经做了对齐，且我们可以利用pgdir\_walk获取虚拟地址对应的页地址，实现很简单，如下所示：



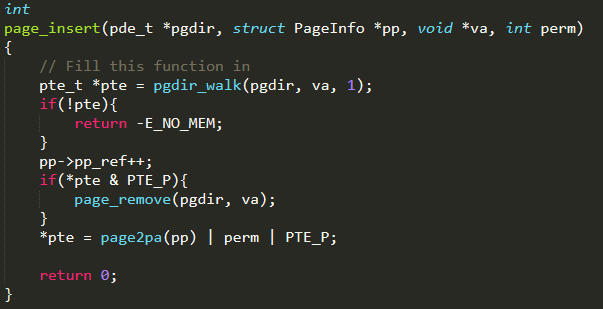
* boot\_map\_region\_large()

该函数实现的是对页表的一段虚拟地址到物理地址的映射，和boot\_map\_region()类似，但是每次映射的单位是PTSIZE大小，即一整个页表（页表目录的一个条目）所映射的内存大小，直接使用PDX获取va对应的页表目录中的页表条目即可，需要注意的是因为是从页表目录的条目直接指向了物理页，因此需要设置PTE\_PS位，代码如下所示：



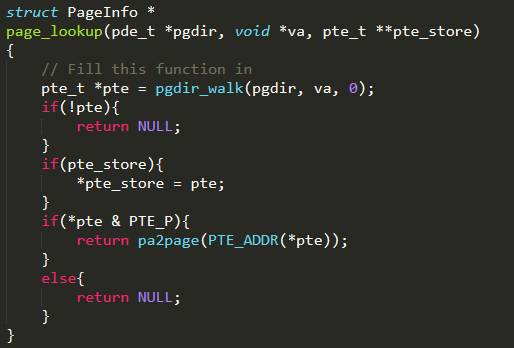
* page\_insert()

该函数将一个物理页映射到一个虚拟地址中，页表项的许可位给出的要求。如果已经有一个页被映射到了该虚拟地址则将其移除。如果页表不存在则创建一个新的页表并插入页表目录中。如果之前有页位于va则必须设置TLB无效。首先利用pgdir\_walk获取虚拟地址对应的页地址，如果不存在该页pgdir\_walk会自己创建一个新页，如果返回值为NULL则是因为内存不够导致分配新页失败。将需要映射的物理页的ref自增，判断va是否已经映射新的页，如果是则将其移除，设置TLB无效将在移除函数中实现，最后将页表条目的值修改为物理页的地址加许可为即可。具体代码如下所示：



* page\_lookup()

该函数返回虚拟地址对应的物理页，如果传入的存储地址不为0同时需要将地址传入该存储地址。同样利用pgdir\_walk获取虚拟地址对应的页表条目，根据情况存入存储地址，如果该页表条目有效则将其物理地址对应的PageInfo返回即可，代码如下所示：



* page\_remove()

该函数解绑虚拟地址和一个物理页，根据要求，首先用page\_lookup()获取物理页的PageInfo，如果存在则调用page\_decref函数减少其ref，如果为0则释放该页。同时需要注意把对应的页表条目清零，利用page\_looku()的第三个参数。此外还要使用tlb\_invalidate函数将TLB中对应的条目无效化。代码如下所示：

