**操作系统课程设计实验报告**

1. 实验内容：

在xv6系统中实现：修改内存布局，将堆栈移动到地址空间的顶部并实现堆栈增长。

1. 实验分析：

程序的内存映射取决于我们如何将程序加载到内存中并设置页表（从而使它们指向正确的物理页）。这都是在exec.c中实现的，作为exec系统调用的一部分，使用vm.c中提供的底层支持来实现虚拟内存。因此要更改内存布局，我们必须更改exec代码以以我们想要的新方式加载程序并分配堆栈。具体来说，首先打开exec.c检查执行系统调用的exec(…)函数。Exec执行以下操作:

1. 打开可执行文件并解析它。这一段的其余部分仅供参考。通常，可执行文件由一个包含信息的头组成，这些信息允许我们索引文件的其余部分。之后的文件由部分组成，包括代码、全局数据，有时还有其他部分，如未初始化的数据。这些是我们需要从可执行文件中初始化的内存部分。头信息包括节的数量、文件中每个节的开始、它在虚拟内存中的映射位置以及每个节的长度。
2. 使用setupkvm()初始化内核内存，它将内核的页面映射到进程地址空间。
3. 然后，它继续使用loaduvm()将可执行文件的各个节加载到内存中，loaduvm()为每个节创建内存页，并将它们映射到地址空间。

然后，我们可以方便地跟踪用户地址空间的结束位置，这也使用一个值定义了进程的大小(proc->sz)。因此，当我们映射新页面时，sz(向上四舍五入到下一页)可以作为它们的虚拟地址，因为我们只是按顺序填充地址空间。

我们希望从用户地址空间的顶部开始定位堆栈，以便给它空间来增长。allocuvm它有3个参数:

（1）页表(pgdir)。

（2）我们正在映射的第一个页面的虚拟地址——这需要更改为指向内存中用户部分的顶部页面(就在KERNBASE下面)。

（3）我们正在映射的最后一个页面的虚拟地址。对于我们来说，我们创建的堆栈只有一个页面，这样就可以在同一页面中创建一个比第一个地址稍大的地址。

allocuvm分配页面，并将其映射到页表。所以，基本上我们已经完成了移动栈的操作，只需要将这些参数更改为正确的值。

同时，所有在sycall .c(和sysfile.c)中定义的用于访问用户堆栈的函数都要进行一些检查，看看地址是否确实在堆栈上。这些检查将地址与sz进行比较，因为sz在旧的实现中是堆栈的顶部。因此我们必须更改这些检查。

我们必须跟踪堆栈的大小，并添加另一个循环来遍历堆栈页（kmallocs为每个页，memmoves创建一个来自父页的副本，然后mapages()将其添加到页表）。除了遍历的虚拟地址范围之外，循环将非常相似。在添加堆栈增长之前，该堆栈只有一个页面，但随着堆栈增长，我们需要跟踪堆栈页面的数量。为了做好准备，我们需要在struct proc中添加一个变量来跟踪堆栈的大小。这个计数器从一页的堆栈开始并在exec()中设置。