动态内存分配与虚拟地址空间

在完成了多道应用程序(即充分利用内存空间，将所有的应用程序加载到内存中，一个接一个地运行完成)，分时多任务系统（即多个任务交替运行）后，这一节实验我们完成的是一个不需要指定应用程序内存布局，将其固定到特定位置，而是通过虚拟地址空间的抽象，动态加载应用程序到内存中，通过基于段页式虚拟地址空间的内存映射机制，还进一步加强了应用于内核的隔离。

在日常的使用操作系统过程中，运行一个程序时，其给人的直接感受就是此时这个应用程序独占cpu，但实际上，cpu是通过时分复用的方法，即交替地运行许多应用程序，但由于其切换的速度很快，肉眼来说我们仍然觉得cpu就是为当前这个应用程序服务。

首先需要的第一个问题即是如何充分利用内存空间，在前面完成的操作系统中，所有的应用被加载到内存指定位置，内核代码于用户程序代码均可以访问整个内存空间，这会导致用户程序有机会篡改内核代码，带来一系列错误。现代操作系统均使用基于段页式的地址空间映射方式，每个程序在其所拥有的地址空间空间中可以做任何事，但其不能访问其它应用程序的和内核的地址空间内容。

为了完成这个任务，需要解决下列的一系列问题

- 硬件中物理内存的范围是什么?
- 哪些物理内存空间需要建立页映射关系?
- 如何建立页表使能分页机制?
- 如何确保OS能够在分页机制使能前后的不同时间段中都能正常寻址和执行代码?
- 页目录表（一级）的起始地址设置在哪里?
- 二级/三级等页表的起始地址设置在哪里，需要多大空间?
- 如何设置页目录表项的内容?
- 如何设置其它页表项的内容?
- 如果要让每个任务有自己的地址空间，那每个任务是否要有自己的页表?
- 代表应用程序的任务和操作系统需要有各自的页表吗?
- 在有了页表之后，任务和操作系统之间应该如何传递数据?

总体的实现思路如下:

1. 将应用程序的内存布局起始地址改为0，使其与现代操作系统的结果保持一致
2. 在内核中实现动态内存分配
3. 编写页表项和，地址空间，段等数据结构抽象
4. 建立虚拟内存映射关系，将虚拟空间的地址映射到真实的物理地址的实现
5. 为内核地址空间建立映射关系
6. 为用户程序地址空间建立映射关系，这里需要解析ELF文件
7. 重新构建用户程序任务管理器
8. 重新配置trap与任务切换，从而使得任务切换与异常可以正常使用，这里设计到指令平滑的思想。
9. 为内核访问用户空间的数据建立相应的函数功能，内核代码需要访问用户空间缓冲区的内容。