《操作系统实现》上机实验

虚拟内存与页式管理

20201106

Xinu 采用了最基础的段式内存管理机制，所有进程运行在同一个内存地址空间。然而一般的操作系统都支持通过页式管理机制来管理虚拟内存，并使 32 位架构下每个进程的逻辑地址空间达到 4GB。

本实验要求修改 Xinu，实现一个支持页式内存管理的 Xinu-vm 版本。

本次实验预计占用 4 次上机课时间（11/06，11/13，11/20，11/27），截止时间为 12 月 4 日 0 点。

Xinu-vm 应该满足如下要求：

a）每个进程的虚拟地址空间为 4GB

b）尽可能少地进行逻辑地址到物理地址的一对一映射（课堂讲述的示例实现使用了 32MB 内存做一对一映射），其余的物理内存视作"高端"内存，执行动态映射，但需注意，由于 Xinu 的架构与其他系统如 Linux、xv6 不一致，没有内核空间与用户空间的明确区分，因此动态映射并不是映射到内核内存的地址空间，而是"当前"进程的地址空间（课堂上将对这一点进行讲述）

c）进程的栈位于逻辑地址空间的高地址处（不要求如课件上所示的在 0xFF7FF000 页）

d）为方便实现，限制单个进程的栈空间不超过 4MB

e）从逻辑地址的低地址空间分配堆，并假定每次分配应该是一个或多个 4KB 页（堆可以在一对一映射的地址空间中分配，也可以只在"高端"内存区域分配，请自行决策）

f) 堆的分配应该满足在连续可用的空间中分配，如：从 32MB 开始的内存连续分配 3 个堆，大小为 4KB，4MB，4KB，其地址分别为 32MB、32MB+4KB、36MB+4KB。其中 4MB 堆会使用两个页表（前一个页表的 1~1023 项，后一个页表的 0 项）

g) 仔细修改进程创建、进程结束相关的代码，使得进程创建后能正确运行，进程结束时所占据的内存空间（含页目录、页表、堆、栈等）都得到释放并且代码能正确运行

h) 修改 shell 进程相关代码，使得在虚拟内存环境下，至少 echo 命令能够正常运行

i) 将代码与实验报告（设计决策、实验效果等）打包成文件 Xinu-20201106（后缀名为 zip 或 tar.gz）提交（打包前请执行 make clean 清除编译产生的中间文件）

j) 分配释放栈空间的函数名字自取，而进程代码能够直接接触的堆空间函数统一命名为 allocmem 和 deallocmem，并与原 getmem 和 freemem 保持一致的参数列表和返回值

提示：

1) 附带代码包含了修改过的 vm.h，start.S，ctxsw.S 和 reschd.c，其中 start.S 包含了示例实现的栈地址以及对 vminit 的调用，若直接使用该文件，需协调栈地址并实现 vminit；注意修改 prototypes.h 中 ctxsw 函数的声明

2) vm.h 中包含了部分可用的宏，如 PDX(va)将一个给定虚拟地址转换为页目录索引，PTX 则获取虚拟地址所含的页表项索引

3) 由于虚拟内存的引入，在未完全实现之前，可以将系统的大部分功能先去掉，再根据所实现的功能逐步开放。如，一开始可以将 nulluser()中所有代码注释掉，添

加少许代码验证初始化是否成功；然后开启 nulluser()中的各种初始化（仍禁止创建新进程的操作），检查空进程是否正常；然后开启创建新进程（只需创建一个最简单的），测试进程创建是否能正常，新进程是否正确运行在虚拟内存空间；测试分配、释放堆空间是否正确；创建多个简单进程，检查进程正常运行结束与被杀死，进程内存空间是否正确释放；最后开启 shell 进程的执行，检查 echo 命令是否能正常使用

4) 在进程表项中添加一个字段，用于存储该进程的页目录的物理地址。resched.c 中以该字段（pgdir）作为 ctxsw 的第三个参数。可根据自己的喜好取名

5) 原 Xinu 中的内存初始化函数调用（initialize.c 中的 meminit()和 bufinit()）应当被注释掉；nulluser 中统计空闲内存的代码可以注释掉或修改使其适用

6) Meminit.c 中的 setregs()方法及其所用到的数据很重要，请不要删除相关代码或去掉对 setregs()的调用

7) 在 C 代码中，可以使用如下两个宏定义来刷新虚拟地址到物理地址的映射缓存（TLB），其中 invlpg 指令将在课堂上讲解

```
#define lcr3(pgdir)    do {asm volatile("movl %0,%%cr3" : : "r" ((pgdir)));} while(0);
#define invlpg(va)    do {asm volatile("invlpg (%0)" : : "r" ((va)) : "memory"); } while(0);
```