第三次实例分析

注意：

本次实例分析使用的是xv6源码版本为rev11；linux代码版本为4.12.10。

Xv6中内存管理和设备管理的相关代码，内存部分的代码联系比较紧密，请各组同学务必阅读内存管理全部源码，理解代码中函数的调用关系，不要只关注讲的部分。

第一部分：内存的初始化

相关文件及函数：

main.c文件中的main()函数，主要调用kinit1()、kvmalloc()、seginit()、kinit2()、userinit()、

kalloc.c文件

重点关注问题：

1. 初始化的时候给内核分配来多大的空间(几个页)？为什么？每个页都存放了哪些内容?

2. 总共分配了几次？为什么?

3. setupkvm()函数的作用是什么?

4. xv6中可用物理内存是用什么数据结构进行组织管理的?

5. mappages()和walkpgdir()函数的作用分别是什么？（不用讲，由第三组同学来讲）

6. 进阶：linux内核内存初始化是都做了哪些工作？（64位系统）请参考相关源码[init/main.c文件中的start_kernel()函数]进行分析。

第二部分：虚拟内存的分配

相关文件及函数：

umalloc.c文件中的malloc()、free()、morecore()函数

重点关注问题：

1. malloc()从哪个数据结构中选择可用空间？这个数据结构的初始状态是什么？如何进行系统的第一次分配?

2. 空间不够出现在哪些情况下？空间不够时做哪些操作，调用了哪些函数?

3. mappages()和walkpgdir()函数的作用分别是什么？（不用讲，由第三组同学来讲）

4. sbrk()函数：也即sys_sbrk()函数中为何调用switchuvm()?

5. 进阶：linux代码中可用空间是用何种数据结构组织的，请结合相关源码[mm/alloc_page.c]进行分析。与xv6中数据结构有何区别?

第三部分：页表项的建立和换页

相关文件及函数：

vm.c文件中mappages()和walkpgdir()函数

重点关注问题：

1. mappages()函数中PGROUNDDOWN()函数的作用是什么？为何需要调用该函数?

2. mappages()函数中“*pte = pa | perm | PTE_P;”代码的含义是什么?

3. walkpgdir()函数中“*pde = V2P(pgtab) | PTE_P | PTE_W | PTE_U;”代码的含义是什么?

4. virtual address与physical address的映射关系是什么？对虚拟内存是如何实现的?（page swap）

5. 进阶：linux代码中virtual address与physical address是如何映射的？如何实现了虚拟内存(page swap)？请结合相关源码[mm/vmscan.c]进行分析。

第四部分：进程创建和加载内存过程、内存的保护模式

相关文件及函数：

proc.c文件中fork()函数，主要调用copyuvm()

exec.c文件中的exec()函数，主要调用allocuvm()、loaduvm()、clearpteu()、switchuvm()

entryother.S文件，trap.c文件中trap()函数的default分支(//PAGEBREAK: 13)；

重点关注问题：

1. exec()函数中“allocuvm(pgdir, sz, sz + 2*PGSIZE)”的理解。

2. loaduvm()函数中设置panic的原因是什么?

3. switchuvm()函数中寄存器是如何设置的，pushcli()和popcli()的含义。

4. 哪个寄存器中的哪位表示了内存的保护模式？14号中断(T_PGFLT)代表了什么？xv6操作系统是如何处理的?

5. 进阶：linux 系统中 14 号中断是如何处理的？请结合相关源码[mm/memory.c]进行分析。