实例分析-3：虚拟内存

- xv6 项目地址：https://pdos.csail.mit.edu/6.828/2023/xv6.html
- xv6 代码地址：https://github.com/mit-pdos/xv6-riscv
- Linux v4.12 地址：https://github.com/torvalds/linux/tree/v4.12
- 注意：xv6 中内存管理的相关代码联系比较紧密，请各组同学务必阅读全部源码，理解代码中函数的调用关系，不要只关注本组所讲部分。

第一部分：内存初始化和页表切换

本部分所涉及的内容主要在 `kernel` 目录下的 `main.c` 和 `kalloc.c` 和 `proc.c` 和 `trampoline.S` 等文件中。

基础题

- xv6 的虚拟地址是如何布局的?
- `kvminit()` 函数是如何工作的？它如何构建 xv6 的虚拟地址布局?
- `kvminithart()` 函数的作用是什么?
- xv6 中可用物理内存是用什么数据结构进行组织管理的?
- `trampoline.S` 中，汇编命令 `csrw satp` 的作用是什么？在哪些情况下需要执行这一命令?
- 结合 `uint64_t trampoline_userret = TRAMPOLINE + (userret - trampoline);` 代码，解释 xv6 中的 trampoline 机制是如何工作的。

进阶题

- Linux kernel 内存初始化做了哪些工作？请参考相关源码 [`init/main.c` 的 `start_kernel()` 函数] 进行分析。

第二部分：虚拟内存分配

本部分所涉及的内容主要在 `user` 目录下的 `umalloc.c` 等文件中。

基础题

- `malloc()` 选择可用空间的数据结构是什么？这个数据结构的初始状态是什么?
- 若 `malloc()` 空间不足，将会执行哪些操作？哪些情况下会出现空间不足的情况?
- Linux 中可用空间是用何种数据结构组织的？请结合相关源码 [`mm/page_alloc.c`] 进行分析。
- Linux 与 xv6 组织可用空间的数据结构有何区别?
- Linux 中，各进程如何管理自己的虚拟地址空间？请结合相关源码 [`include/linux/mm_types.h`] 进行分析。（无需分析具体的增删等操作流程）

进阶题

- Linux 如何实现页替换（Page Swap）？请结合相关源码 [`mm/vmscan.c`] 进行分析。

第三部分：页表项的增删改查

本部分所涉及的内容主要在 `kernel` 目录下的 `vm.c` 等文件中。

基础题

- xv6 中虚拟地址与物理地址是如何映射的?
- `mappages()` 函数中 `PGROUNDDOWN()` 函数的作用是什么？为何需要调用该函数?
- `mappages()` 函数中 `panic()` 函数的作用是什么？有哪些可能导致 `panic` 的情况?
- `mappages()` 函数中 `*pte = PA2PTE(pa) | perm | PTE_V` 代码的含义是什么?
- `walk()` 函数的具体流程是什么？传入参数 `int alloc` 有何作用?
- `freewalk()` 函数的具体流程是什么？`(pte & PTE_V) && (pte & (PTE_R|PTE_W|PTE_X)) == 0` 判断的作用是什么?

进阶题

- Linux 中虚拟地址与物理地址是如何映射的？虚拟地址又是如何布局的？
- Linux 的页表层次是怎样的？试以 64 位 X86 架构为例进行分析。

第四部分：进程创建和内存加载

本部分所涉及的内容主要在 `kernel` 目录下的 `exec.c` 和 `proc.c` 和 `vm.c` 和 `trap.c` 等文件中。

基础题

- `exec()` 函数中 `uvmalloc(pagetable, sz, sz + 2*PGSIZE, PTE_W)` 语句的意义是什么？
- `exec()` 函数中调用 `loadseg()` 的作用是什么？
- `uvmcopy()` 函数会在何时被调用？该函数具体进行了哪些操作？
- risc-v 的 page fault 相关中断有哪些？xv6 如何处理这些中断？
- 若要在 xv6 中实现 `fork()` 的 Copy-on-Write 功能，可以如何修改逻辑？简要阐述思路即可。（提示：可结合前几问的内容进行思考）

进阶题

- Linux 如何处理 page fault 中断？请结合相关源码 [`mm/memory.c`] 进行分析。