HW3 about 内存管理

2000011483 喻勃洋

1. 交换技术

课上讲到，交换技术最早是用在小型分时系统里的，比如那种 roll in roll out 机制。它的核心思想其实很直接：当内存快不够用了，系统就把内存里的某些进程整个搬到外存（一般是磁盘上的专门区域，也就是 swap 空间），然后把外存里的进程再搬回内存，这样就能动态地让进程在内存和外存之间切换。

实现的时候就包括：

- 需要换出去的内容
  - 主要是进程运行时产生或者修改的数据，比如堆和栈，这些都得完整地保存到外存上， 换出去的内容一般会放在磁盘上专门划出来的一块区域，这块区域通常是连续的，这样磁盘读写会快一点
- 什么时候触发交换?
  - 一般是在内存快不够用的时候，或者真的不够用了，调度器会提前做准备，避免内存突然用光
- 选哪个进程换出去
  - 要看进程的状态，不能把正在等I/O的进程换出去，优先选那种短时间内不会被调度的进程
- 换回来的进程要不要回原来的位置
  - 其实没必要，系统可以用动态重定位技术，让进程在不同的物理内存位置也能正常运行
  - 如果进程在运行过程中变大了，比如堆或者栈长大了，系统还得给它分配新的内存区域，这样内存管理就更复杂了

其实总的来说可以这样理解：

- 传统的交换技术就是一种比较粗暴的整体搬运，直接把整个进程的内容搬来搬去，这和现在操作系统用的按页管理完全不是一个思路
- 系统里有个叫 swapper 的特殊内核进程，专门负责定期扫描和选择哪些进程要换进换出
- 由于整体交换带来的磁盘I/O开销太大，调度也会变慢，所以现在的系统基本不用这种整进程的交换了，而是用分页交换，也就是按页（一般4KB）来换进换出，结合虚拟内存和缺页异常机制，效率高很多

所以在现代系统里，比如 Linux，虽然还有 swap 分区这个说法，但实际操作已经是以页为单位动态置换了，不再是传统意义上的整体交换了

2. xv6-riscv 内存管理

a. xv6-riscv 的物理页组织和分配（alloc）

- xv6 里，物理内存页就是用一个链表来管理的，链表的节点结构叫 struct run。所有没用的物理页都串在一起。
- 分配的时候（kalloc），就是从链表头拿一页；释放的时候（kfree），就是把不用的页插回链表头。这样谁空闲谁就能被下次用到。

b. 进程虚拟地址空间增长（vmalloc）

- 进程要扩展虚拟空间，比如堆变大，主要靠 uvmalloc 这个函数。
  - uvmalloc一页一页地用 kalloc 分配物理内存
  - 然后用 mappages 把这些物理页和进程的虚拟地址连起来。
    - mappages 就是负责把虚拟地址和物理地址挂钩。这样进程能用的空间就变大了。

c. 在 xv6-riscv 里实现 mmap的关键要点

- 如果要加 mmap，进程结构体里得加个 mmap 区域的链表，记录每个映射的虚拟区间、权限、文件描述符这些。
- 添加 mmap 和 munmap 这两个系统调用，负责分配和释放虚拟区间。
  - 可以先不分配物理页，等访问到的时候再分配（这样更省内存）。
- 处理带来的新的页错误
  - 比如访问到没分配物理页的 mmap 区域时，内核要分配物理页并建立映射。
  - 如果是文件映射，还要把文件内容读进来。同时还有考虑同步和解除映射这些操作。