Lan Fangzhou
Report for OSlab1
2017.5.20

1. 实现 pmm.c

我们在 pmm 中主要实现 alloc 函数和 free 函数. 对于 alloc 函数, 我们使用两个链来记录内存的使用情况: 其中每一个链上的节点存储一段内存 (起始地址和大小). usedList 记录已经被使用的内存段,freeList 记录尚未被使用的内存段. 初始时 freeList 为整段堆区,usedList 为空. 每次 alloc 时将 freeList 中足够大的一段对齐后放到 usedList,free 时将 usedList 中某段放回 freeList 中. ■

2. 测试 kmm.c

我们试图对 kmm 进行一点测试. 试图进行一系列 alloc 和 free 操作, 发现当分配空间为几 KB 时可以正常的分配很长时间, 但是当分配空间为几 MB 时, 几十次操作之后, 由于回收不全, 就会出现问题. ■

3. 实现 kmt.c

对于 kmt, 我们主要根据课程 slides 和 xv6 的思想进行实现. 自旋锁, 线程的结构体定义主要参考 xv6. 线程的创建: 首先在线程池中寻找一个可用的线程, 填写相关信息, 然后申请一块堆栈 kstack, 再利用 make 函数创建一个新的寄存器现场. 线程的销毁: 删除相关信息, 释放内存线程的调度: 利用时间作为 seed, 生成随机数, 随机的抽取所有等待的线程进行调度. 信号量, 自旋锁: 参考课程 slides 实现 ■

3. 测试 kmt.c

首先测试创建共 16 个输出字母的线程, 运行一段时间没有问题. 然后测试生产者-消费者问题, 左括号数不会少于右括号数, 而且不会同时产生超过 BUFSIZE 个左括号. ■

3. 实现 os.c

在中断处理程序中加入线程相关的内容: 保存旧的寄存器现场, 切换线程, 再返回新的寄存器现场. ■