

# Lab2 实验报告

---

- 于凡奇 18307130182

## kalloc

---

函数返回指向一个空闲4KB页的指针。`kmem.freelist` 指向一个空闲页的链表，只要此时它不为NULL，说明存在可分配的空闲页，那么 `kalloc` 直接将该指针返回，并让 `kmem.freelist` 指向下一个空闲页。

## kfree

---

在对指针v的地址进行边界检验、对齐检验后，将其进行类型转换后插入到链表 `kmem.freelist` 的头部。

## pgdir\_walk

---

该函数返回某个虚拟地址对应的页表项指针。函数的参数需要指定0级页表，于是从0级页表开始一级级迭代向下走。

- 如果该页表项已被建立，那么函数将一直迭代到3级页表，直接返回对应页表项的指针。
- 如果页表项尚未存在，且参数 `alloc` 为1，那么函数自动为其一级级地建立页表项，最终返回3级页表中新建的页表项的指针。对于0~2级页表中新建的页表项，将PTE\_P和PTE\_TABLE置位，表示表项有效且为Page Table Entry。
- 如果页表项不存在，且参数 `alloc` 为0或者页面分配失败，那么返回NULL。

需要注意的是页表项中储存的是物理地址，在赋值给C语言中的指针时需要借助P2V进行虚实转换；反之，在写入页表时需要用V2P进行转换。在申请页面(`kalloc`)成功后需要用`memset`将页面清空，避免某些位置上的PTE\_P被置位。

## map\_region

---

`pgdir_walk` 为虚拟地址创建了页表项，但并未建立对物理地址的映射，而 `map_region` 正是完成这项工作，将虚拟地址va映射到物理地址pa。当va和pa的地址非页对齐时，将向下取整至最近的页对齐地址进行映射，如果va和pa的页内偏移相同则它们仍能正好对应。具体实现为：

- 计算出待映射区域的起止地址最近的页对齐地址 `a` 和 `last`
- 借助 `pgdir_walk` 找到 `a` 对应的页表项
- 如果该页表项已经被映射(`PTE_P == 1`)则panic
- 向页表项中填入对应的页对齐的物理地址，并配置权限位
- 如果 `a` 和 `last` 不在一页内，则让 `a` 前进一页，重复上述操作，直至区域内的所有页面配置完毕

## vm\_free

---

接受一个页表指针，清空页表上的表项以及表项所覆盖的更低级的页表的表项。

具体实现上，函数采用递归的方法，遍历当前的页表，并对所有有效且为PTE\_TABLE的表项进行vm\_free，子递归调用返回后将表项清零，页面中的512项均清零后使用kfree释放当前页。

## 测试

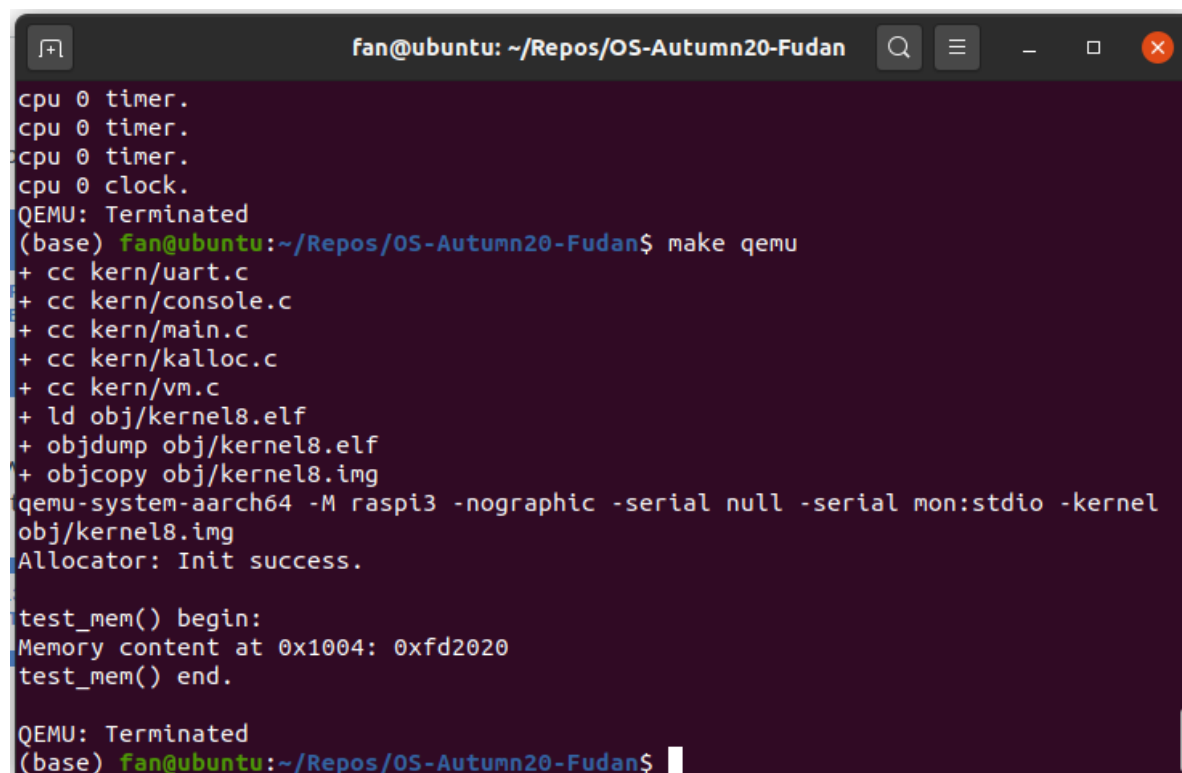
```
void
test_mem()
{
    cprintf("\ntest_mem() begin:\n");

    *((uint64_t *)P2V(4)) = 0xfd2020;
    char *pgdir = kalloc();
    memset(pgdir, 0, PGSIZE);
    map_region((uint64_t *)pgdir, (void *)0x1004, PGSIZE, 4, 0);
    asm volatile("msr ttbr0_el1, %[x]": : [x]"r"(V2P(pgdir)));

    cprintf("Memory content at 0x1004: 0x%x\n", *(uint64_t *)0x1004);
    cprintf("test_mem() end.\n\n");
}
```

先通过ttbr1向物理地址0x4处写入0xfd2020。通过kalloc申请一个页面作为新的0级页表，使用memset初始化页面，利用map\_region将虚拟地址0x1004映射到0x4。此时由于虚拟地址未与页对齐，实际上需要对两个页面进行映射。另外由于0x4和0x1004的页内偏移相同，0x1004依然能映射到0x4处。

开启ttbr0，令其指向我们新配的页表，再访问虚拟地址0x1004，读出物理地址0x4处的0xfd2020。



```
fan@ubuntu: ~/Repos/OS-Autumn20-Fudan
cpu 0 timer.
cpu 0 timer.
cpu 0 timer.
cpu 0 clock.
QEMU: Terminated
(base) fan@ubuntu:~/Repos/OS-Autumn20-Fudan$ make qemu
+ cc kern/uart.c
+ cc kern/console.c
+ cc kern/main.c
+ cc kern/kalloc.c
+ cc kern/vm.c
+ ld obj/kernel8.elf
+ objdump obj/kernel8.elf
+ objcopy obj/kernel8.img
qemu-system-aarch64 -M raspi3 -nographic -serial null -serial mon:stdio -kernel
obj/kernel8.img
Allocator: Init success.

test_mem() begin:
Memory content at 0x1004: 0xfd2020
test_mem() end.

QEMU: Terminated
(base) fan@ubuntu:~/Repos/OS-Autumn20-Fudan$
```

