提交评测 2023-04-03 21:21:01 | 评测冷却: 86s

提交

更新

Lab 2 Exam

准备工作：创建并切换到 `lab2-exam` 分支

1. **基于已完成的 lab2 提交自动初始化 lab2-exam 分支**
2. 在开发机依次执行以下命令：

```
$ cd ~/学号
$ git fetch
$ git checkout lab2-exam
```

初始化的 `lab2-exam` 分支基于课下完成的 `lab2` 分支，并且在 `tests` 目录下添加了 `lab2_perm_stat` 样例测试目录。

题目背景 & 题目描述

在 Lab2 课下，我们学习了每一页表项的高 20 位为物理页号，低 12 位为权限位。在12 位权限位中，第 0-7 位是给操作系统使用的保留位，第 8-11 位是硬件规定的权限位。操作系统可以通过统计页表中某些权限位不为 0 的页表项个数来实现各种功能。在本题中，我们要求你实现一个权限位的**统计**函数。

实现权限位匹配统计函数 `page_perm_stat` ，该函数的函数声明如下：

```
u_int page_perm_stat(Pde *pgdir, struct Page *pp, u_int perm_mask);
```

该函数各参数的意义，函数的具体功能描述如下：

通过调用此函数，返回页目录 `pgdir` 对应的 **所有二级页表项**（下称 **页表项** ）中，物理页号为 `Page` 结构体 `pp` 对应页号并且权限位满足 `perm_mask` 的**有效**页表项个数。这里的满足指的是， `perm_mask` 含有的权限是页表项中权限的子集。

其中，页表项 `pte` 有效是指：页表项中有效位为 `1` ，也就是说 `pte & PTE_V` 的运算结果非 0。

例如：

- 若 `perm_mask = PTE_D` ，页表项中 `perm = PTE_D | PTE_G` ，则该页表项不计入统计，因为该页表项是 **无效** 的。

具体来说，实现此函数，需要遍历所有**有效**的页表项，统计所有同时满足以下两个条件的页表项个数：

1. 页表项中第 12-31 位（即物理页号 `PFN` ）等于 `Page` 结构体 `pp` 对应的物理页面号
2. 页表项第 0-11 位权限位满足函数参数 `perm_mask`

题目要求

1. 在 `include/pmap.h` 中添加 `page_perm_stat` 函数的声明：

```
u_int page_perm_stat(Pde *pgdir, struct Page *pp, u_int perm_mask);
```

2. 在 `kern/pmap.c` 中实现该函数。

提示

同学们在完成函数时，可以考虑使用 `PTE_ADDR` 、 `page2pa` 等简化代码实现。

样例输出 & 本地测试

对于以下样例：

```
void test_perm_stat() {
    struct Page *p;
    assert(page_alloc(&p) == 0);
    Pde *pgdir = (Pde *)page2kva(p);
    u_int va[4] = {UTEXT, UTEXT + BY2PG, UTEXT + 1024 * BY2PG, UTEXT + 1025 * BY2P
    u_int perm[4] = {PTE_V | PTE_D, PTE_V | PTE_D | PTE_G, PTE_V | PTE_D | PTE_G,
             PTE_V | PTE_G};
    struct Page *pp;

    assert(page_alloc(&pp) == 0);
    assert(page_insert(pgdir, 0, pp, va[0], perm[0]) == 0);
    assert(page_insert(pgdir, 0, pp, va[1], perm[1]) == 0);
    assert(page_insert(pgdir, 0, pp, va[2], perm[2]) == 0);
    assert(page_insert(pgdir, 0, pp, va[3], perm[3]) == 0);
    int r = page_perm_stat(pgdir, pp, PTE_D);
    assert(r == 3);
```

提交评测 2023-04-03 21:21:01 | 评测冷却: 86s

其应当输出：

```
test_perm_stat succeeded!
```

你可以使用：

- `make test lab=2_perm_stat && make run` 在本地测试上述样例（调试模式）
- `MOS_PROFILE=release make test lab=2_perm_stat && make run` 在本地测试上述样例（开启优化）

或者在 `init/init.c` 的 `mips_init` 函数中自行编写测试代码并使用 `make && make run` 测试。

提交评测 & 评测标准

请在开发机中执行下列命令后，在**课程网站上提交评测。**

```
$ cd ~/学号/
$ git add -A
$ git commit -m "message" # 请将 message 改为有意义的信息
$ git push
```

测试点说明及分数分布如下：

测试点序号	评测说明	分值
1	与样例相同	10 分
2	测试数据保证所有非零的页表项有效	20 分
3	测试数据保证所有非零的一级页表项有效	25 分
4	测试数据保证所有非零的二级页表项有效	25 分
5	综合评测	20 分

测试用例补充说明：