提交评测 2024-04-10 20:58:08

提交

更新

Lab 2 Extra

准备工作：创建并切换到 `lab2-extra` 分支

请在**自动初始化分支后**，在开发机依次执行以下命令：

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

初始化的 `lab2-extra` 分支基于课下完成的 `lab2` 分支，并且在 `tests` 目录下添加了 `lab2_buddy` 样例测试目录。

题目背景

在理论课程中，我们学习了**伙伴系统**（buddy system）。伙伴系统是 Linux 系统采用的介于固定分区与可变分区之间的动态分区技术。某些情况下，操作系统需要分配连续的物理页，例如 DMA 控制器传输数据时一般要求缓存区的物理地址连续。分配连续的物理页也在一定程度上能提高操作系统的效率，伙伴系统可以满足操作系统分配连续物理页的需求。

伙伴系统在分配内存空间时，一个大的内存区间会分裂成两个大小相等的小区间，这两个小区间就称为“伙伴”。内存空间经过不断的划分可能会形成若干个不连续的空闲内存区间，操作系统以链表维护每种大小的空闲内存区间。

题目描述

在本题中，你需要对 MOS 的高地址 4MB 物理内存建立伙伴系统。

伙伴系统中无论是已分配内存区间或空闲内存区间，其大小均为 2 的 k 次幂，k 为整数，$n \leq k \leq m$。在本题中，为简化实现，2^n 为 MOS 的页面大小 `PAGE_SIZE` 即 4KB；2^m 为 `2 * PAGE_SIZE` 即 8KB。

伙伴系统在收到分配 x 字节内存的请求时，会分配不小于 x 字节的最小的 2 的幂次字节的空闲内存区间。例如，5KB 的请求会向上取整为 8KB 的空闲区间的请求。若所需要的 2^k 字节空闲区间链表非空，则直接分配。否则 2^k 字节的空闲区间已经耗尽，寻找 2^{k+1} 字节的空闲区间并将其分为两个 2^k 字节的空闲区间，这

提交评测 2024-04-10 20:58:08

伙伴系统在收到释放内存区间的请求时，会尝试将所需要释放的内存区间与其伙伴合并为更大的空闲区间，不断合并直到不能合并为止。注意：两个物理地址相邻且大小相同的内存区间不一定为伙伴，只有伙伴才能被合并。

伙伴系统的实现要求与细节请参看**题目要求**部分。

题目要求

在本题中，你需要使用内存区间分别为 4KB 与 8KB 的空闲链表建立高地址 4MB 物理内存的伙伴系统。

- 伙伴系统的空闲链表为 `buddy_free_list[0]` 与 `buddy_free_list[1]`，分别维护大小为 4KB 与 8KB 的空闲内存区间。
- 由于在 MOS 中一个页控制块代表 4KB 的物理页，因此规定：对于 8KB 的内存区间，以区间中**低地址**物理页的页控制块作为代表。例如，物理地址 `[0x03C00000, 0x03C02000)` 所对应的 8KB 内存区间，在链表中将以 `[0x03C00000, 0x03C01000)` 对应的页控制块存储，分配或释放时也用相应页控制块作为代表。
- 初始时，高地址 4MB 物理内存将划分为 512 个 8KB 的空闲区间。高地址 4MB 的所有页控制块将从 `page_free_list` 中移除，代表各个 8KB 空闲区间的页控制块将插入至 `buddy_free_list[1]` 中。

同时，我们将提供部分代码（请参看**实验提供代码**部分），你需要将其粘贴至 `kern/pmap.c` 之后，并补全或者实现如下函数：

内存区间的分配（`int buddy_alloc(u_int size, struct Page **new)`）

本函数的功能为：

- 通过**伙伴系统**分配大小不低于 `size` 字节的内存空间：
  - 分配成功时：
    - 将分配的内存区间对应的页控制块填入 `new` 指向的变量。
    - 返回值为分配的内存区间所包含的物理页数。
  - 分配失败时，返回值为 `-E_NO_MEM`。
- 分配的内存区间需要满足以下条件：
  - 内存区间空闲，也即未被分配。
  - 内存区间大小为不小于 `size` 的最小的 2 的幂次字节。

1. 计算需要分配的字节数。
2. 当所需大小对应的空闲链表非空时，优先选择该链表中的一个页控制块对应的内存区间分配。
3. 当需要分配 4KB 空闲区间，但 4KB 空闲链表为空时：
    i. 若 8KB 空闲链表为空，分配失败。
    ii. 选择空闲链表中的一个页控制块对应的 8KB 空闲区间，将其分为两个等大小的伙伴区间。
    iii. 分配低地址的 4KB 空闲区间，并将高地址的 4KB 空闲区间插入至对应空闲链表。
4. 当需要分配 8KB 空闲区间，但 8KB 空闲链表为空时分配失败。

注意：

- 本函数返回的内存区间必须从伙伴系统的空闲链表中取得。
- 保证调用函数时参数 `size` 不为 `0`，也不超过 `8192`，也即所需分配的内存空间大小不超过 8KB。
- 保证调用函数时参数 `new` 指向的变量空间存在且合法。
- 不需要考虑清空对应物理页面中的数据。

内存区间的释放（`void buddy_free(struct Page *pp, int npp)`）

本函数的功能为：

- 通过**伙伴系统**释放页控制块 `pp` 对应的物理页代表的内存区间，`npp` 代表待释放的内存区间所包含的物理页数。

以下是 `buddy_free` 函数的参考实现方案（你也可以自行实现本函数，但必须保证满足函数定义与功能约束）：

1. 获得内存区间对应的页控制块与内存区间大小。
2. 当需要释放 4KB 空闲区间，且伙伴空闲时：
    i. 将伙伴区间的页控制块从空闲链表中移出。
    ii. 将所需释放的空闲区间与伙伴合并为 8KB 空闲区间。
    iii. 将 8KB 空闲区间对应的页控制块插入对应空闲链表，完成内存区间的释放。
3. 当需要释放 4KB 空闲区间且伙伴已分配，或是需要释放 8KB 空闲区间时：
    i. 将页控制块插入对应空闲链表，完成内存区间的释放。

注意：

提交评测 2024-04-10 20:58:08

- 保证调用函数的参数所对应的内存区间一定是通过 `buddy_alloc` 函数分配得到的，且不会被调用者分割或合并。

提交

更新

任务总结

在提交前，你需要完成以下任务：

- 完成 `buddy_alloc` 函数，维护 `buddy_free_list` 链表。
- 完成 `buddy_free` 函数，维护 `buddy_free_list` 链表。

本题不涉及课下代码的修改。

实验提供代码

请将本部分提供代码附加在你的 `kern/pmap.c` 的尾部，然后开始完成题目。

```
#include <buddy.h>

struct Page_list buddy_free_list[2];

void buddy_init() {
    LIST_INIT(&buddy_free_list[0]);
    LIST_INIT(&buddy_free_list[1]);
    for (int i = BUDDY_PAGE_BASE; i < BUDDY_PAGE_END; i += PAGE_SIZE) {
        struct Page *pp = pa2page(i);
        LIST_REMOVE(pp, pp_link);
    }
    for (int i = BUDDY_PAGE_BASE; i < BUDDY_PAGE_END; i += 2 * PAGE_SIZE) {
        struct Page *pp = pa2page(i);
        LIST_INSERT_HEAD(&buddy_free_list[1], pp, pp_link);
    }
}

int buddy_alloc(u_int size, struct Page **new) {
    /* Your Code Here (1/2) */
}

void buddy_free(struct Page *pp, int npp) {
    /* Your Code Here (2/2) */
}
```

本地测试说明

你可以使用：

提交评测 2024-04-10 20:58:08

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。

如果样例测试中输出了如下结果，说明你通过了本地测试。

```
buddy_init succeed!
Buddy page test passed!
```

提交评测

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
```

评测说明

评测时使用的 `mips_init()` 函数示意如下：

```
void mips_init() {
    mips_detect_memory();
    mips_vm_init();
    page_init();
    buddy_init();

    buddy_test();

    halt();
}
```

具体要求和分数分布如下：

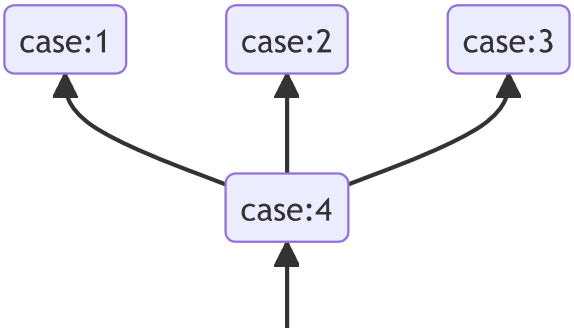
测试点序号	评测说明	分值
1	与本地测试相同	20
2	不涉及内存区间释放	10
3	申请的内存空间均 > 4KB	10

提交评测

2024-04-10 20:58:08

5	综合测试	40

测试点依赖关系如下（箭头方向表示依赖方向）：



提交

更新