OS-Project8

常烁晨-521021910369

Task1 base

1、实验简介：要求实现基本的虚拟内存分配器的功能。主要需要编写的模块就分为三部分：TLB、页表、磁盘。每次进行访存时，在三部分逐级进行访问。同时设置全局变量用于统计出错率。

2、代码实现：

（1）初始化，读取内存地址。

```
TLB_struct TLB[TLB_SIZE];
PageTable_struct PageTable[PAGE_TABLE_SIZE];
Memory_struct Memory[FRAME_NUMBER];

address_file = fopen("addresses.txt", "r");
output_file = fopen("output.txt", "w");
backing_store = fopen("BACKING_STORE.bin", "rb");
```

（2）实现 TLB 功能。

```
for(int i=0; i<TLB_SIZE; i++){
    if(TLB[i].page_number == num){
        tlb_hit++;
        TLB[i].last_use_time = address_count;
        return TLB[i].frame_number;}}
```

（3）当 TLB 未命中，从页表中查询。

```
int min_time = 0x7fffffff, min_time_index = 0;
for(int i=0; i<TLB_SIZE; i++){
    if(TLB[i].last_use_time < min_time){
        min_time = TLB[i].last_use_time;
        min_time_index = i;}}
//以下更新 TLB，略过
```

（4）当出现 Page Fault，从磁盘读取文件，并重新调用函数，使得对应内存空间中的数通过页表完成读取，在页表读取后更新 TLB。

```
Memory[min_time_index].last_use_time =
address_count; page_fault++;
        fseek(backing_store, num * PAGE_SIZE,
SEEK_SET);
        fread(Memory[min_time_index].frame,
sizeof(char), FRAME_SIZE, backing_store);
//以下更新页表，略过
return get_frame(num);//返回函数自身，重新读取。
```

Task2 bonus

本部分只需修改 page fault 的情况。由于物理内存的大小变为虚拟内存的一半，因此程序在发出一个虚拟内存地址后，该地址对应的数据可能不在物理内存中，而是在磁盘中。但是页表的大小与 page 的数量相同，因此需要将其他页替换到页表中。

代码实现：

```
for(int i=0; i<PAGE_TABLE_SIZE; i++){
    if(PageTable[i].frame_number ==
min_time_index){
        PageTable[i].valid = 0; break;}}
    PageTable[num].valid = 1;
    PageTable[num].frame_number = min_time_index;
```

实验结果

```
parallels@ubuntu-linux-22-04-desktop:~/final-src-osc10e/ch10$ gcc virtual.c -o virtual
parallels@ubuntu-linux-22-04-desktop:~/final-src-osc10e/ch10$ ./virtual
TLB hit rate: 5.500000 %
Page fault rate: 24.400000 %
```

```
parallels@ubuntu-linux-22-04-desktop:~/final-src-osc10e/ch10$ gcc virtual_bonus.c -o virtual_bonus
parallels@ubuntu-linux-22-04-desktop:~/final-src-osc10e/ch10$ ./virtual_bonus
TLB hit rate: 5.500000 %
Page fault rate: 53.900000 %
```