实验五 请求分页模拟实

姓名	学号	学院	日期
臧祝利	202011998088	人工智能学院	2022.11.7

实验目的

通过编写和调试请求页式存储管理的模拟程序以加深对请求页式存储管理方案的理解。

实验内容

页面淘汰算法可采用FIFO置换算法（或者其他置换算法），在淘汰一页时，判断它是否被改写过，如果被修改过，将它写回到外存上。

实验报告

实验环境

系统： Win10

g++： V8.1.0

Code Eidtor： Visual Studio Code

encode： GBK

实验过程

实验思路

页表结构定义如下：

```
#define N 64

struct
{
    int lNumber; //页号
    int pNumber; //物理块号
    int dNumber; //在磁盘上的位置
    int write;   //修改位标志
    int flag;    //存在位标志
} page[N];
```

首先创建页表，输入页表号和在磁盘上的位置，同时记录页数 `cnt`；

输入主存块号，记录个数 `m`；

接着输入指令：

- 从逻辑地址中获得页号
  - 页号大于页表长度--> 越界中断
  - 页不在内存--> 缺页中断
    - `FIFO算法`,选择最先进去的页进行换出；
    - 如果换出的页被修改，则写回外存，同时将外存中的页调入内存，修改页表

- 计算页内偏移量，获得物理地址；

函数实现

void Judge(unsigned logicaddress)

判断逻辑地址是否发生越界中断或缺页中断；

```
void Judge(unsigned logicaddress)
{
    if (pagenum > cnt - 1) //发生越界
    {
        printf("不存在此页!\n");
    }
    else
    {
        if (page[pagenum].flag == 0) //发生缺页中断
        {
            printf("发生缺页中断: %d\n", pagenum);
            pageout = p[cursor];           //取得最先进来的页号
            p[cursor] = pagenum;           //替换页号
            if (page[pagenum].write == 1) //被修改过，要写回页面
            {
                printf("将%d页写回磁盘第%d块\n", pageout, page[pageout].dNumber);
            }
            printf("淘汰主存块%d中的页%d,从磁盘第%d块中调入页%d\n", page[pageout].pNumber, pageout,
page[pagenum].dNumber, pagenum);
            page[pagenum].pNumber = page[pageout].pNumber; //装入pageout页的主存块中
            page[pagenum].flag = 1;                        //存在，置为1
            page[pagenum].write = 0;                       //没被修改，置为0
            page[pageout].flag = 0;                        //不存在，置为0
            cursor = (cursor + 1) % cnt;
        }
    }
}
```

源代码

```
#include <iostream>
#include <cstdio>
using namespace std;

#define N 64
#define LENGTH 10

struct
{
    int lNumber; //页号
    int pNumber; //物理块号
    int dNumber; //在磁盘上的位置
    int write;   //修改位标志
    int flag;    //存在位标志
} page[N];

int p[N]; //存放页号的队列

int lnumber, dnumber, pnumber;        //输入的页号,辅存地址,主存块
int cnt = 0;                          //页数
int m = 0;                            //给进程的内存块数
```

```
unsigned logicaddress, pysicaddress; //逻辑地址，物理地址
int option;                          //指令
int cursor;                          //指针
int pageout;                         //出队页面
unsigned pagenum;                    //计算得到的页号

void Judge(unsigned logicaddress)
{
    if (pagenum > cnt - 1) //发生越界
    {
        printf("不存在此页!\n");
    }
    else
    {
        if (page[pagenum].flag == 0) //发生缺页中断
        {
            printf("发生缺页中断: %d\n", pagenum);
            pageout = p[cursor];           //取得最先进来的页号
            p[cursor] = pagenum;           //替换页号
            if (page[pagenum].write == 1) //被修改过，要写回页面
            {
                printf("将%d页写回磁盘第%d块\n", pageout, page[pageout].dNumber);
            }
            printf("淘汰主存块%d中的页%d,从磁盘第%d块中调入页%d\n", page[pageout].pNumber, pageout,
page[pagenum].dNumber, pagenum);
            page[pagenum].pNumber = page[pageout].pNumber; //装入pageout页的主存块中
            page[pagenum].flag = 1;                        //存在，置为1
            page[pagenum].write = 0;                       //没被修改，置为0
            page[pageout].flag = 0;                        //不存在，置为0
            cursor = (cursor + 1) % cnt;
        }
    }
}

void print()
{
    printf("————————页表————————\n");
    for (int i = 0; i < cnt; i++)
    {
        printf("%d\t%d\t%d\t%d\t%d\n", page[i].lNumber, page[i].pNumber, page[i].dNumber, page[i].flag,
page[i].write);
    }

    printf("————————页表————————\n");
}
int main()
{
    printf("输入页表的信息,创建页表(若页号为-1,则结束输入)\n");
    for (int i = 0; i < N; i++)
    {
        printf("输入页号和辅存地址:");
        scanf("%d %d", &lnumber, &dnumber);
        if (lnumber == -1) //结束输入
        {
            break;
        }
        page[i].lNumber = lnumber; //存入页号和磁盘位置
        page[i].dNumber = dnumber;
        page[i].write = 0; //没有被修改过
        cnt++;             //输入的总个数
```

```
    }

    printf("输入主存块号,主存块数要小于%d,(以-1结束):", cnt);
    for (int i = 0; i < cnt; i++)
    {
        scanf("%d", &pnumber); //输入块号
        if (pnumber == -1)     //结束输入
        {
            break;
        }
        page[i].pNumber = pnumber; //存储块号
        page[i].flag = 1;          //存在置为1;
        p[i] = i;                  //将页调入队列中
    }

    // print();
    while (1)
    {
        printf("输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:");
        scanf("%d %d", &option, &logicaddress);
        if (option != 1 && option != 0)
        {
            printf("程序退出!\n");
            return 0;
        }
        pagenum = logicaddress >> LENGTH; //求得页号
        Judge(logicaddress);              //判断是否越界或缺页中断
        if (option == 1)                  //该页修改，修改位置1·
        {
            page[pagenum].write = 1;
        }
        else if (option == 0) //不做操作
        {
        }
        pysicaddress = page[pagenum].pNumber * (1 << 10) + logicaddress % (1 << 10); //计算物理地址
        printf("逻辑地址是%d, 对应的物理地址是%d\n", logicaddress, pysicaddress);     //输出地址
        // print();
    }
    return 0;
}
```

运行结果

```
PS D:\Codes\page> cd "d:\Codes\page\" ; if ($?) { g++ page.cpp -o page } ; if ($?) { .\page }
输入页表的信息,创建页表(若页号为-1,则结束输入)
输入页号和辅存地址:0 10
输入页号和辅存地址:1 20
输入页号和辅存地址:2 30
输入页号和辅存地址:3 40
输入页号和辅存地址:4 50
输入页号和辅存地址:-1 -1
输入主存块号,主存块数要小于5,(以-1结束):2 3 4 -1
输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:0 0
逻辑地址是0, 对应的物理地址是2048
输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:1 1024
逻辑地址是1024, 对应的物理地址是3072
输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:1 3072
发生缺页中断: 3
淘汰主存块2中的页0,从磁盘第40块中调入页3
逻辑地址是3072, 对应的物理地址是2048
输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:1 0
发生缺页中断: 0
淘汰主存块3中的页1,从磁盘第10块中调入页0
逻辑地址是0, 对应的物理地址是3072
输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:0 10240
不存在此页!
逻辑地址是10240, 对应的物理地址是0
输入指令性质(1-修改, 0-不修改，其他-结束程序运行)和逻辑地址:3 0
程序退出!
```

初始时页表为:

页号	物理块号	外存地址	存在位	修改位
0	2	10	1	0
1	3	20	1	0
2	4	30	1	0
3		40	0	0
4		50	0	0

- 输入0 0时，页号为$\lfloor 0/1024 \rfloor = 0$，偏移量为$0 \mod 1024 = 0$;且存在位为1，物理地址为$2 * 1024 + 0 = 2048$
- 输入1 1024时，页号为$\lfloor 1024/1024 \rfloor = 1$，偏移量为$1024 \mod 1024 = 0$;且存在位为1，物理地址为$3 * 1024 + 0 = 3072$
- 输入1 3072时，页号为$\lfloor 3072/1024 \rfloor = 3$，偏移量为$1024 \mod 1024 = 0$;存在位为0，发生缺页中断，选择2号块，2号块的页0没有被修改，因此淘汰页0，将磁盘地址40的页3放入2号块；物理地址为$2 * 1024 + 0 = 2048$

页表变为

页号	物理块号	外存地址	存在位	修改位
0	2	10	0	0
1	3	20	1	1
2	4	30	1	0
3	2	40	1	1
4		50	0	0

- 输入1 0时，页号为$\lfloor 0/1024 \rfloor = 0$，偏移量为$0 \mod 1024 = 0$;存在位为0，发生缺页中断，因此要写回磁盘，将页1写回磁盘地址20；然后进行置换，选择主存块3中的页1淘汰，把页0放入内存3中，物理地址为3072

页表变为

页号	物理块号	外存地址	存在位	修改位
0	3	10	1	1
1	3	20	0	1
2	4	30	1	0
3	2	40	1	1
4		50	0	0

- 输入0 10240时。页号为10，发生越界中断
- 输入3 0，退出程序；

实验总结

通过这个实验加深了对分页存储管理方式的认识，尤其是淘汰页面时如果页面被修改，操作会涉及到淘汰页面和新换入的页这一过程。