实验报告目录

01 实验概述

1.1 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。

1.2 实验内容

请求分页模拟：

- 页面淘汰算法采用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；
- 数据结构：
  - 虚地址结构：页号（10 ～ 16位，共7位）+页内偏移量（0 ～ 9位，共10位）；
  - 页表结构：

```
struct Page
{
    int logicNumber;//页号
    int physicalNumber;//物理块号
    int diskNumber;//在磁盘上的位置
    bool write;//修改位标志
    bool flag;//存在位标志
};
```

- 在模拟试验中，采用预调页的方法。

02 实验设计

将页表项定义为一个结构体，成员变量包含：

- 页号：虚拟地址的页号；
- 物理块号：主存块号；
- 在磁盘上的位置：磁盘号；
- 修改位标志：是否被修改过，当页被修改时，将其写回磁盘；
- 存在位标志：是否在主存块中，当页不在主存块中时，需要将其调入主存块；

将页表定义为一个结构体，成员变量包含：

- 页表：使用vector存储页表项；
- 页大小：用于计算页号与页内偏移（在本实验中页大小为1024字节，即2^{10}字节，10为页内偏移量位数）；
- 主存块最大数：可用的主存块数；
- 淘汰队列：将最近访问的页表项按顺序存入淘汰队列，用于淘汰一页时实现FIFO；
- 空闲队列：将可用的主存块按顺序存入空闲队列；

将页表、主存块的初始化，预调页，打印页表，淘汰一页、访问页等主要实验操作封装为页表的成员函数。

03 实验实现

3.1 页表项结构定义

将页表项定义为一个结构体，包含页号、物理块号、在磁盘上的位置、修改位标志、存在位标志五个属性。

```
// 页表项
struct Page
{
    int logicNumber;    // 页号
    int physicalNumber; // 物理块号
    int diskNumber;     // 在磁盘上的位置
    bool write;         // 修改位标志
    bool flag;          // 存在位标志

    // 初始化页表项
    Page(int logicNumber, int physicalNumber, int diskNumber, bool write, bool flag)
    {
        this->logicNumber = logicNumber;
        this->physicalNumber = physicalNumber;
        this->diskNumber = diskNumber;
        this->write = write;
        this->flag = flag;
    }
};
```

3.2 页表结构定义

将页表定义为一个结构体，包含页表、页大小、主存块最大数、淘汰队列、空闲队列五个属性。并将页表的初始化、主存块的初始化、预调页、打印页表、打印淘汰队列、淘汰一页、访问页等操作封装为页表的成员函数。

```
struct PageTable
{
    vector<Page> pages;         // 页表，存储页表项
    int pageSize;               // 页大小，用于计算页号与页内偏移
    int blockNumberMax;         // 主存块最大数
```

```
    queue<int> removePageQueue; // 淘汰队列，FIFO
    queue<int> emptyBlockQueue; // 空闲队列

    // 初始化页表
    PageTable(int pageSize);

    // 初始化页表，读入页表若干表项的页号、磁盘号
    void InitPages();

    // 初始化主存块，读入可用主存块号，存入空闲队列
    void InitBlock();

    // 预调页，将可用主存块按顺序分配给页表项
    void PreschedulePage();

    // 打印页表
    void PrintPageTable();

    // 打印淘汰队列
    void PrintRemoveQueue();

    // 淘汰一页(先进先出)
    void RemovePage();

    // 访问页
    // logicAddress: 逻辑地址
    // write: 本次访问操作是否修改
    void SchedulePage(int logicAddress, bool write);
}
```

3.3 页表成员函数实现

此部分为实验核心部分，包含页表的初始化、主存块的初始化、预调页、打印页表、打印淘汰队列、淘汰一页、访问页等操作的实现。

3.3.1 初始化页表

构造函数中初始化页大小，在本实验中页大小为1024字节（即2^{10}字节，10为页内偏移量位数）。

```cpp
// 初始化页表
PageTable::PageTable(int pageSize)
{
    this->pageSize = pageSize;
}
```

3.3.2 初始化页表项

读入页表若干表项的页号、磁盘号，存入页表。

```cpp
// 初始化页表，读入页表若干表项的页号、磁盘号
void PageTable::InitPages()
{
    int lNumber, dNumber, idx = 0; // 页号，磁盘号，输入序号
    cout << "创建页表，输入页表的信息(若页号为-1，则结束输入)\n";
    cout << "输入页号和磁盘地址\nIn[" << idx++ << "] > ";
    cin >> lNumber >> dNumber;

    while (lNumber != -1)
    {
        pages.push_back(Page(lNumber, -1, dNumber, false, false));
        cout << "输入页号和磁盘地址\nIn[" << idx++ << "] > ";
        cin >> lNumber >> dNumber;
    }

    cout << "页表初始化完成!\n\n";
}
```

3.3.3 初始化主存块

读入可用主存块号（不超过页表项数），存入空闲队列。

```cpp
// 初始化主存块，读入可用主存块号，存入空闲队列
void InitBlock()
{
    int blockNumber, idx = 0;
    cout << "输入主存块号，主存块数要小于[" << pages.size() << "]，(以-1结束)\n";
    cout << "In[" << idx++ << "] > ";
    cin >> blockNumber;

    while (blockNumber != -1)
    {
        emptyBlockQueue.push(blockNumber);
        cout << "In[" << idx++ << "] > ";
        cin >> blockNumber;
```

```
    }

    // 主存块数最大值为页表项数
    blockNumberMax = emptyBlockQueue.size();
    if (blockNumberMax > pages.size())
    {
        cout << "Error: 主存块数大于页表项数，程序退出!\n";
        exit(0);
    }

    cout << "主存块初始化完成!\n\n";
}
```

3.3.4 预调页

将可用主存块按顺序分配给页表前若干项，并将此顺序存入淘汰队列。

```
// 预调页，将可用主存块按顺序分配给页表项
void PreschedulePage()
{
    int blockNumber;
    cout << "预调页中，将可用主存块按顺序分配给页表项...\n";
    int tableIndex = 0;

    while (!emptyBlockQueue.empty())
    {
        blockNumber = emptyBlockQueue.front();
        emptyBlockQueue.pop();
        pages[tableIndex].physicalNumber = blockNumber;
        pages[tableIndex].flag = true;
        removePageQueue.push(tableIndex);
        cout << "-> 页[" << pages[tableIndex].logicNumber
                << "]调入主存块[" << pages[tableIndex].physicalNumber << "]\n";
        tableIndex++;
    }

    cout << "预调页完成!\n\n";
}
```

3.3.5 打印页表

打印页表，包含每个页表项的页号、物理块号、磁盘块号、修改位、存在位。

```
// 打印页表
void PrintPageTable()
{
```

```
    cout << "====== 页表 ====== >\n";
    cout << "页号\t物理块号\t磁盘块号\t修改位\t存在位\n";

    for (int i = 0; i < pages.size(); i++)
    {
        cout << pages[i].logicNumber << "\t"
                << pages[i].physicalNumber << "\t\t"
                << pages[i].diskNumber << "\t\t"
                << pages[i].write << "\t"
                << pages[i].flag << "\n";
    }
}
```

3.3.6 打印淘汰队列

打印淘汰队列，包含淘汰队列中每个页表项的页号，该队列使得页表项按FIFO顺序淘汰。

```
// 打印淘汰队列
void PrintRemoveQueue()
{
    cout << "内存块淘汰队列（页号）：";
    queue<int> temp = removePageQueue;

    while (!temp.empty())
    {
        cout << pages[temp.front()].logicNumber;
        temp.pop();
        if (!temp.empty())
            cout << " <- ";
    }
    cout << "\n";
}
```

3.3.7 淘汰一页

淘汰一页，即按FIFO从淘汰队列中取出页，若该页被修改，则将其写回磁盘，否则不需要写回磁盘。将该空闲出的主存块加入空闲队列，将该页状态标记为不在主存块中。

```
// 淘汰一页(先进先出)
void RemovePage()
{
    if (removePageQueue.empty())
    {
```

```
        cout << "Error: 淘汰队列为空，程序退出!\n";
        exit(0);
    }

    // 从淘汰队列中取出页，FIFO
    Page &page = pages[removePageQueue.front()];
    removePageQueue.pop();

    // 打印淘汰信息
    cout << "\n-> 淘汰主存块[" << page.physicalNumber
            << "]中的页[" << page.logicNumber << "]"
            << "...\n";

    if (page.write)
    {
        // 若该页被修改，将其写回磁盘
        cout << "-> 页[" << page.logicNumber << "]被修改，将其写回磁盘。\n";
        page.write = false;
    }
    else
    {
        // 若该页未被修改，不需要写回磁盘
        cout << "-> 页[" << page.logicNumber << "]未被修改，不需要写回磁盘。\n";
    }
    cout << "\n";

    // 将该主存块加入空闲队列
    emptyBlockQueue.push(page.physicalNumber);
    // 从主存中删除该页
    page.flag = false;
    page.physicalNumber = -1;
}
```

3.3.8 访问页地址

访问页地址，即根据逻辑地址计算页号与页内偏移，查找页表中是否存在该页，若存在则判断该页是否在主存块中，若在则打印信息，若不在则判断主存块是否已满，若已满则淘汰一页，若未满则将该页调入主存块。若不存在该页，则打印错误信息。

```
// 访问一页
// logicAddress: 逻辑地址
// write: 本次访问操作是否修改
void SchedulePage(int logicAddress, bool write)
{
    // 计算页号与页内偏移
```

```
    int logicNumber = logicAddress / pageSize;
    int offset = logicAddress % pageSize;
    cout << "-> 访问逻辑地址[" << logicAddress
            << "]... \n-> 页号: [" << logicNumber
            << "], 页内偏移: [" << offset << "]\n\n";

    // 查找页表中是否存在该页
    int tableIndex = -1;
    for (int i = 0; i < pages.size(); i++)
        if (pages[i].logicNumber == logicNumber)
        {
            tableIndex = i;
            break;
        }

    // 若页表中不存在该页
    if (tableIndex == -1)
    {
        cout << "Error: 页表中不存在页[" << logicNumber << "] !\n\n";
        return;
    }

    // 若页表中存在该页
    if (pages[tableIndex].flag == true)
    {
        cout << "-> 页[" << logicNumber
                << "]已在主存块[" << pages[tableIndex].physicalNumber << "]中"
                << "\n";
    }
    else
    {
        // 若主存块已满
        cout << "-> 页[" << logicNumber
                << "]不在主存块中"
                << "\n";
        // 若无空闲主存块, 淘汰一页
        if (emptyBlockQueue.empty())
        {
            cout << "-> 主存块已满 ...\n";
            RemovePage();
        }
        int blockNumber = emptyBlockQueue.front();
        emptyBlockQueue.pop();
        pages[tableIndex].physicalNumber = blockNumber;
        pages[tableIndex].flag = true;
        removePageQueue.push(tableIndex);
        cout << "-> 从磁盘第[" << pages[tableIndex].diskNumber
                << "]块中调入页[" << logicNumber
```

```
                << "]到主存块[" << pages[tableIndex].physicalNumber << "]"
                << "\n";
    }

    // 计算物理地址
    int physicalAddress = pages[tableIndex].physicalNumber * pageSize + offset;
    cout << "-> 对应物理地址为[" << physicalAddress << "]"
            << "\n";

    // 若本次访问修改
    if (write)
    {
        pages[tableIndex].write = true;
        cout << "-> 本次访问页[" << logicNumber << "]被修改\n";
    }
}
```

3.4 主函数

主函数中初始化页表、初始化主存块、预调页、打印页表、打印淘汰队列，循环中读入操作类型和逻辑地址，调用访问页地址函数，打印页表、打印淘汰队列。

```
int main()
{
    // freopen("in.txt", "r", stdin);
    // freopen("out.txt", "w", stdout);
    cout << "******************** 请求分页模拟 ********************\n\n";

    // 页表初始化
    int pageSize = 1 << 10; // 页大小
    PageTable pageTable = PageTable(pageSize);

    // 读入页表信息
    pageTable.InitPages();

    // 读入主存块信息
    pageTable.InitBlock();

    // 预调页
    pageTable.PreschedulePage();

    // 打印页表
    pageTable.PrintPageTable();

    // 打印淘汰队列
```

```
    pageTable.PrintRemoveQueue();

    while (true)
    {
        int write, logicAddress;

        // 读入指令性质
        cout << "\n*****************************************************\n";
        cout << "输入指令性质(1-修改，0-不修改，其他-结束程序运行)\n";
        cout << "In > ";
        cin >> write;
        if (write != 1 && write != 0)
            break;

        // 读入逻辑地址
        cout << "输入逻辑地址\n";
        cout << "In > ";
        cin >> logicAddress;
        if (logicAddress < 0)
        {
            cout << "Error: 逻辑地址不能为负数，程序退出!\n";
            exit(0);
        }
        cout << "\n";

        // 访问页
        pageTable.SchedulePage(logicAddress, write);

        // 打印页表、淘汰队列
        cout << "\n";
        pageTable.PrintPageTable();
        pageTable.PrintRemoveQueue();
    }
    cout << "程序运行结束!\n";

    return 0;
}
```

04 实验测试与结果

4.1 实验测试输入

设计如下测试用例，对程序的各个功能和边界条件进行测试。

```
0 10
1 11
2 12
4 20
5 21
6 22
-1 -1

2 3 5 7
-1

0 10
1 1025
0 3072
1 6143
0 6144
1 2058
1 20

-1
```

4.2 实验测试结果

```
PS D:\files\Course\2023Fal-操作系统\lab\lab7>./page.exe
****************** 请求分页模拟 ******************

创建页表，输入页表的信息(若页号为-1，则结束输入)
输入页号和磁盘地址
In[0] > 0 10
输入页号和磁盘地址
In[1] > 1 11
输入页号和磁盘地址
In[2] > 2 12
```

```
输入页号和磁盘地址
In[3] > 4 20
输入页号和磁盘地址
In[4] > 5 21
输入页号和磁盘地址
In[5] > 6 22
输入页号和磁盘地址
In[6] > -1 -1
页表初始化完成!

输入主存块号，主存块数要小于[6]，(以-1结束)
In[0] > 2
In[1] > 3
In[2] > 5
In[3] > 7
In[4] > -1
主存块初始化完成!

预调页中，将可用主存块按顺序分配给页表项...
-> 页[0]调入主存块[2]
-> 页[1]调入主存块[3]
-> 页[2]调入主存块[5]
-> 页[4]调入主存块[7]
预调页完成!

====== 页表 ====== >
页号     物理块号         磁盘块号         修改位   存在位
0        2               10               0        1
1        3               11               0        1
2        5               12               0        1
4        7               20               0        1
5        -1              21               0        0
6        -1              22               0        0
内存块淘汰队列（页号）：0 <- 1 <- 2 <- 4

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 0
输入逻辑地址
In > 10

-> 访问逻辑地址[10]...
-> 页号：[0]，页内偏移：[10]

-> 页[0]已在主存块[2]中
-> 对应物理地址为[2058]

====== 页表 ====== >
```

```
页号    物理块号        磁盘块号        修改位  存在位
0       2               10              0       1
1       3               11              0       1
2       5               12              0       1
4       7               20              0       1
5       -1              21              0       0
6       -1              22              0       0
内存块淘汰队列（页号）：0 <- 1 <- 2 <- 4

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 1
输入逻辑地址
In > 1025

-> 访问逻辑地址[1025]...
-> 页号：[1]，页内偏移：[1]

-> 页[1]已在主存块[3]中
-> 对应物理地址为[3073]
-> 本次访问页[1]被修改

====== 页表 ====== >
页号    物理块号        磁盘块号        修改位  存在位
0       2               10              0       1
1       3               11              1       1
2       5               12              0       1
4       7               20              0       1
5       -1              21              0       0
6       -1              22              0       0
内存块淘汰队列（页号）：0 <- 1 <- 2 <- 4

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 0
输入逻辑地址
In > 3072

-> 访问逻辑地址[3072]...
-> 页号：[3]，页内偏移：[0]

Error：页表中不存在页[3] !


====== 页表 ====== >
页号    物理块号        磁盘块号        修改位  存在位
0       2               10              0       1
1       3               11              1       1
```

```
2       5               12              0       1
4       7               20              0       1
5       -1              21              0       0
6       -1              22              0       0
内存块淘汰队列（页号）：0 <- 1 <- 2 <- 4

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 1
输入逻辑地址
In > 6143

-> 访问逻辑地址[6143]...
-> 页号：[5]，页内偏移：[1023]

-> 页[5]不在主存块中
-> 主存块已满 ...

-> 淘汰主存块[2]中的页[0]...
-> 页[0]未被修改，不需要写回磁盘。

-> 从磁盘第[21]块中调入页[5]到主存块[2]
-> 对应物理地址为[3071]
-> 本次访问页[5]被修改

====== 页表 ====== >
页号    物理块号        磁盘块号        修改位  存在位
0       -1              10              0       0
1       3               11              1       1
2       5               12              0       1
4       7               20              0       1
5       2               21              1       1
6       -1              22              0       0
内存块淘汰队列（页号）：1 <- 2 <- 4 <- 5

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 0
输入逻辑地址
In > 6144

-> 访问逻辑地址[6144]...
-> 页号：[6]，页内偏移：[0]

-> 页[6]不在主存块中
-> 主存块已满 ...

-> 淘汰主存块[3]中的页[1]...
```

```
-> 页[1]被修改，将其写回磁盘。

-> 从磁盘第[22]块中调入页[6]到主存块[3]
-> 对应物理地址为[3072]

====== 页表 ====== >
页号     物理块号          磁盘块号          修改位   存在位
0        -1               10               0        0
1        -1               11               0        0
2        5                12               0        1
4        7                20               0        1
5        2                21               1        1
6        3                22               0        1
内存块淘汰队列（页号）：2 <- 4 <- 5 <- 6

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 1
输入逻辑地址
In > 2058

-> 访问逻辑地址[2058]...
-> 页号：[2]，页内偏移：[10]

-> 页[2]已在主存块[5]中
-> 对应物理地址为[5130]
-> 本次访问页[2]被修改

====== 页表 ====== >
页号     物理块号          磁盘块号          修改位   存在位
0        -1               10               0        0
1        -1               11               0        0
2        5                12               1        1
4        7                20               0        1
5        2                21               1        1
6        3                22               0        1
内存块淘汰队列（页号）：2 <- 4 <- 5 <- 6

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > 1
输入逻辑地址
In > 20

-> 访问逻辑地址[20]...
-> 页号：[0]，页内偏移：[20]

-> 页[0]不在主存块中
```

```
-> 主存块已满 ...

-> 淘汰主存块[5]中的页[2]...
-> 页[2]被修改，将其写回磁盘。

-> 从磁盘第[10]块中调入页[0]到主存块[5]
-> 对应物理地址为[5140]
-> 本次访问页[0]被修改

====== 页表 ====== >
页号     物理块号          磁盘块号          修改位   存在位
0        5                 10                1        1
1        -1                11                0        0
2        -1                12                0        0
4        7                 20                0        1
5        2                 21                1        1
6        3                 22                0        1
内存块淘汰队列（页号）：4 <- 5 <- 6 <- 0

**************************************************
输入指令性质(1-修改，0-不修改，其他-结束程序运行)
In > -1
程序运行结束!

PS D:\files\Course\2023Fal-操作系统\lab\lab7>
```

05 实验总结

本次实验主要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。

在实验实现中，将页表项和页表分别封装定义为结构体，将页表的初始化、主存块的初始化、预调页、打印页表、打印淘汰队列、淘汰一页、访问页等操作封装为页表的成员函数，实现了页式存储管理、FIFO替换页的模拟程序。