|  |  |
| --- | --- |
| **成绩** |  |

西安交通大学城市学院

操作系统 实验报告

班级 软件191 姓名 胡瑞平 学号 19031198 专业 软件191

实验日期： 2021 年 12 月 19 日

审阅教师： 年 月 日

实验名称： 页面置换算法 实验地点： 实验楼105

|  |
| --- |
| 实验任务概要：  最佳页面置换算法  先进先出页面置换算法  最近最久未使用页面置换算法 |
| 学生实验心得及建议：  再接再厉！ |
| 教师批阅记录： |

**计算机系制**

**二 0二 0 年一月**

最佳页面置换算法的C语言实现

void OPT(int \*BlockofMemory, int \*PageNumofRef, int BlockCount, int PageNumRefCount)

{

int i, j, k;

int MaxIndex1, MaxIndex2;

int MissCount = 0;

int ReplacePage;

int EmptyBlockCount = BlockCount;

printf("\*\*\*\*\*\*\*\*\*\*最佳页面置换算法:\*\*\*\*\*\*\*\*\*\*\n");

//输出页面引用串号

OutputPageNumofRef(PageNumofRef, PageNumRefCount);

for (i = 0; i < PageNumRefCount; i++)

{

if (!PageInBlockofMemory(PageNumofRef[i], BlockofMemory, BlockCount)) //页不在内存中

{

MissCount++;

if (EmptyBlockCount > 0)

{

BlockofMemory[BlockCount - EmptyBlockCount] = PageNumofRef[i];

OutputBlockofMemory(BlockofMemory, BlockCount, -1, PageNumofRef[i]);

EmptyBlockCount--;

}

else

{

MaxIndex1 = MaxIndex2 = 0;

//求出未来最长时间不被访问的页

for (j = 0; j < BlockCount; j++)

{

MaxIndex2 = DistanceOpt(BlockofMemory, PageNumofRef, j, i, PageNumRefCount);

if (MaxIndex1 < MaxIndex2)

{

MaxIndex1 = MaxIndex2;

k = j;

}

}

ReplacePage = BlockofMemory[k];

BlockofMemory[k] = PageNumofRef[i];

OutputBlockofMemory(BlockofMemory, BlockCount, ReplacePage, PageNumofRef[i]);

}

}

else

{

OutputBlockofMemory(BlockofMemory, BlockCount, -1, PageNumofRef[i]);

}

}

printf("缺页次数为: %d\n", MissCount);

printf("OPT缺页率为: %.3f\n", (float)MissCount / PageNumRefCount);

}

先进先出页面置换算法的C语言实现

void FIFO(int \*BlockofMemory, int \*PageNumofRef, int BlockCount, int PageNumRefCount)

{

int i;

int ReplacePage;

int ReplaceIndex = 0;

int MissCount = 0;

int EmptyBlockCount = BlockCount;

printf("\*\*\*\*\*\*\*\*\*\*先进先出页面置换算法:\*\*\*\*\*\*\*\*\*\*\n");

//输出页面引用串号

OutputPageNumofRef(PageNumofRef, PageNumRefCount);

for (i = 0; i < PageNumRefCount; i++)

{

if (!PageInBlockofMemory(PageNumofRef[i], BlockofMemory, BlockCount)) //页不在内存中

{

MissCount++;

if (EmptyBlockCount > 0)

{

BlockofMemory[BlockCount - EmptyBlockCount] = PageNumofRef[i];

OutputBlockofMemory(BlockofMemory, BlockCount, -1, PageNumofRef[i]);

EmptyBlockCount--;

}

else

{

ReplacePage = BlockofMemory[ReplaceIndex];

BlockofMemory[ReplaceIndex] = PageNumofRef[i];

ReplaceIndex = (ReplaceIndex + 1) % BlockCount;

OutputBlockofMemory(BlockofMemory, BlockCount, ReplacePage, PageNumofRef[i]);

}

}

else

OutputBlockofMemory(BlockofMemory, BlockCount, -1, PageNumofRef[i]);

}

printf("缺页次数为:%d\n", MissCount);

printf("FIFO缺页率为:%.3f\n", (float)MissCount / PageNumRefCount);

}

最近最久未使用页面置换算法的C语言实现

void Lru(int \*BlockofMemory, int \*PageNumofRef, int BlockCount, int PageNumRefCount)

{

int i;

int MissCount = 0;

int EmptyBlockCount = BlockCount;

int ReplaceIndex, ReplacePage;

int CurrentVisitIndex, NextVisitIndex;

printf("\*\*\*\*\*\*\*\*\*\*\*\*最近最久未使用页面置换算法:\*\*\*\*\*\*\*\*\*\*\*\*\n");

//输出页面引用串号

OutputPageNumofRef(PageNumofRef, PageNumRefCount);

for (i = 0; i < PageNumRefCount; i++)

{

if (!PageInBlockofMemory(PageNumofRef[i], BlockofMemory, BlockCount)) //页不在内存中

{

MissCount++;

//

// 在此添加代码，实现 LRU 页面置换算法

//

if (EmptyBlockCount > 0)

{

BlockofMemory[BlockCount-EmptyBlockCount]=PageNumofRef[i];

OutputBlockofMemory(BlockofMemory, BlockCount, -1, PageNumofRef[i]);

EmptyBlockCount--;

}else{

CurrentVisitIndex = 0, NextVisitIndex = 0;

for(int j = 0; j < BlockCount; j++){

NextVisitIndex = PreviewOpt(BlockofMemory, PageNumofRef,j,i,PageNumRefCount);

if(NextVisitIndex<CurrentVisitIndex){

NextVisitIndex = CurrentVisitIndex;

ReplaceIndex = j;

}

ReplacePage = BlockofMemory[ReplaceIndex];

BlockofMemory[ReplaceIndex] = PageNumofRef[i];

OutputBlockofMemory(BlockofMemory, BlockCount, ReplacePage, PageNumofRef[i]);

}

}

}

else

OutputBlockofMemory(BlockofMemory, BlockCount, -1, PageNumofRef[i]);

}

printf("缺页次数为:%d\n", MissCount);

printf("LRU缺页率为:%.3f\n", (float)MissCount / PageNumRefCount);

}