学号 E21514033 专业 软件工程 姓名 何铭春

实验日期 教师签字 成绩

实验报告

【实验名称】 页面置换算法比较

【实验目的】

用高级语言模拟页面置换算法LRU，加深对LRU算法的认识。

同时，使用最佳置换算法和先进先出页面置换算法进行比较。

【实验原理】

LRU（最近最久未使用）算法基本原理为：如果某一个页面被访问了，它很可能还要被访问；相反，如果它长时间不被访问，再最近未来是不大可能被访问的。该算法使用堆栈保存页面号，访问过的页面号会被置于堆栈顶部。需要替换时就把栈底元素置换出去。

最佳置换算法：其所选择的被淘汰页面将是以后永不使用的，或许是很长时间内不再被访问的页面。

先进先出算法：总是淘汰最先进入内存的页面。

【实验内容】

### 1.主函数：

int many=10; //运行十个例子

int limit=4; //可分配块数

int amount=100; //访问次数

//随机算法生成process数组

vector<int> process;

for (int i = 0; i < many; ++i)

{

for(int j=0;j<amount;j++)

{

process.push\_back(rand()%10);

}

//调用置换算法

cout<<"最佳置换算法"<<endl;

best(process,limit,amount);

cout<<"先入先出页面置换算法"<<endl;

first(process,limit,amount);

cout<<"LRU置换算法"<<endl;

LRU(process,limit,amount);

cout<<endl;

process.clear();

}

### 2.LRU算法：

select函数，用于寻找最适合被替换的块

choose为被选中的内存块

①先假设第一个是被选中的。

②计算这个块下一次访问间隔，如果没有下一次出现，那就直接返回choose

③对下一个内存块进行判断，如果间隔大于best，这choose=i（这个内存块号），best更新。如果没有下一次，那就直接返回i。如果小于，那就继续下一次循环。

int select(vector<int> process,vector<int> v,int start,int amount)

{

//choose即被置换的页面，先设置为0，起始

int choose=0;

int best; //best用于保存这个起始值，下一次遇到的间隔，下面为其赋值

for (int i = start; i < amount; ++i)

{

if (process[i]==v[choose])

{

best=i-start;

break;

}

if (i==amount-1)

{

return 0;

}

}

for (int i = 1; i < v.size(); ++i)

{

for (int j = start; j < 20; ++j)

{

if (process[j]==v[i])

{

if ((j-start)>best)

{

best=j-start;

choose=i;

break;

}

else

break;

}

//即下一个页面号不会再出现

if (j==amount-1)

{

return i;

}

}

}

return choose;

}

### 3.先入先出算法：

该算法直接使用堆栈，每一次置换堆栈底部内存块，然后从顶部加入新内存块。

### 4.LRU

使用堆栈，每一次访问的内存块被置于堆栈顶部。如果需要替换，那就删除堆栈底部内存块，在堆栈顶部加入内存块。

getVisitTop函数，将被访问的页面快提到堆栈顶部。

int getVisitTop(vector<int>& v,int visit)

{

int temp=v[visit];

for (int i = visit; i < v.size()-1; ++i)

{

v[i]=v[i+1];

}

v[v.size()-1]=temp;;

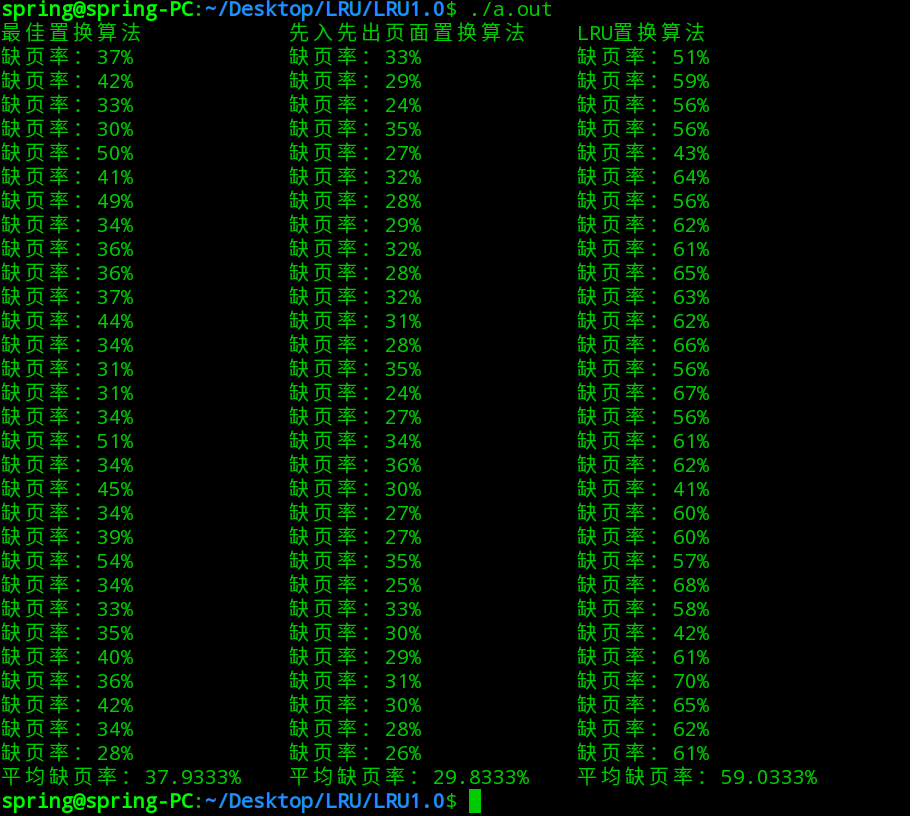
return 0;

}

实验结果：

1.可分配块数：7个 访问页面块号0~9 访问100次

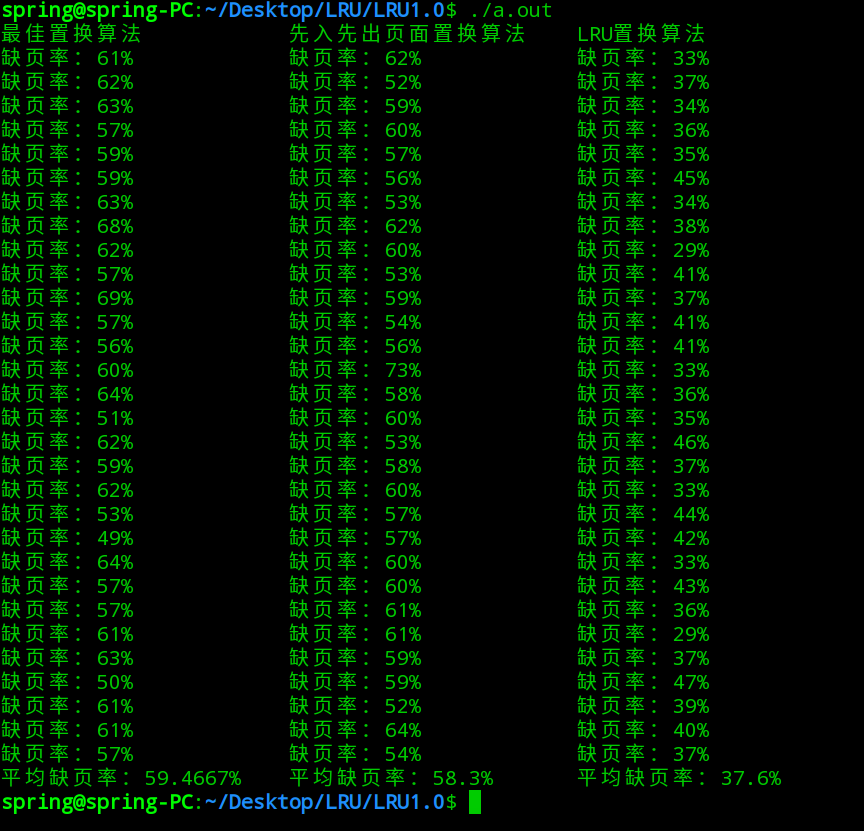
总运行 30次



1.可分配块数：4个 访问页面块号0~9 访问100次

总运行 30次

减少可分配块数为4个



【小结或讨论】

1.由上述实验结果来看，当可分配块数接近所需要的内存块数时，先入先出反而缺页率很少。但是当可分配块数严重不足时，LRU算法效果更好。

2.采用C++ vector做数据结构，方便操作。