操作系统实验之二--内存分配算法的模拟实现

原创

takefetter 于 2018-01-11 13:37:37 发布
分类专栏： 操作系统 文章标签： 实验 操作系统实验 内存分配算法

发一些大三操作系统的实验代码吸引阅读量吧，当时做实验的时候看见网上很多人写的代码并不好，而且很多人

常见的内存分配算法有首次适应算法（First Fit，FF）、循环首次适应算法（Next Fit，NF）、最佳适应算法（Best Fit，BF）、最差适应算法（W

代码写的比较差请轻喷，有问题欢迎吐槽。

代码如下：

```
#include <iostream>
#include <sstream>
#include <iomanip>
#include <string>
#include <list>
using namespace std;
int const MAX=1024;//使用代表内存大小
int SIZE[MAX];//tip:起始位置为0;
class subAreaNode
{
public:
int addr;// 起始地址
int end;//终止地址
int size;// 分区大小
string taskId;// 作业号
};
class NULLArea
{
public:
 int ID;
 int size;
 int begin;
 int end;
};
list<subAreaNode>s;
list<NULLArea>n;
bool comp_BestFit(const NULLArea &lhs, const NULLArea &rhs)
{
 return lhs.size < rhs.size;
}
bool comp_WorstFit (const NULLArea &lhs, const NULLArea &rhs)
{
 return lhs.size > rhs.size;
```

```
}
int Fitcount = 1;
string Fitstring = "Fit";
void cleansize ()
{
 for (int i = 0; i < MAX; i++)
  SIZE[i] =0;
}
subAreaNode setsize(int addr, int size,string taskId)
{
 subAreaNode a;
 for (int i = addr; i < size+addr; i++)
 {
  SIZE[i] = 1;
 }
 a.addr = addr;
 a.size = size;
 a.taskId = taskId;
 a.end = addr + size - 1;
 return a;
}
NULLArea setNULLarea (int begin, int end, int size, int ID)
{
 NULLArea b;
 b.begin = begin;
 b.end = end;
 b.size = size;
 b.ID = ID;
 return b;
}
void ShowAllsubAreaNode ()
{
 cout << "进程已占用内存表如下"<<endl;
 if (s.size() == 0) {
  cout << "tip:内存中没有进程"<<endl;
  cout << "开始地址为0";
  cout << "终止地址为" << MAX - 1 << endl;
    return;
 }
 cout <<setw(6)<< "begin"<<setw (6)<< "end" <<setw (6) << "size" << setw (6) << "ID" << setw (6) << endl;
 for each (subAreaNode a in s)
 {
  cout << setw (6) << a.addr << setw (6) <<a.end << setw (6) << a.size<< setw (6) << a.taskId <<endl;
 }
}
void ShowAllNullArea ()
{
 cout << "空闲内存块表(ID根据有小到大的顺序排列)"<<endl;
 if (n.size () == 0)
 {
  cout << "内存中无空闲区域! ";
  return;
 }
 cout << setw (6) << "begin" << setw (6) << "end" << setw (6) << "size" << setw (6) << "ID"<<endl;
 for each (NULLArea a  in n )
 {
  cout << setw (6) << a.begin << setw (6) << a.end << setw (6) << a.size << setw (6) << a.ID <<endl ;
 }
}
void loadNullArea ()
```

```
{
 n.clear ();
 int number = 1;
 int c, i;
for( i=0;i<MAX;i++)
{
if(SIZE[i]==0)
{
 for (c = i; c < MAX; c++)
 {
  if (SIZE[c] == 1)
  {

   n.push_back (setNULLarea(i,c-1,c-i,number));
   number++;
   i = c;
   break;
  }
  if ((c == MAX - 1) && (SIZE[c] == 0))
  {
   n.push_back (setNULLarea (i, c - 1, c - i, number));
   number++;
   return;
  }
 }
}
}
}
bool cleanAREA (string taskID)
{
 list<subAreaNode>::iterator itor;
 for (itor = s.begin (); itor != s.end ();itor++)
 {
  if (itor->taskId == taskID)
  {
   for (int i = itor->addr; i <  itor->addr+itor->size; i++)
   {
    SIZE[i] = 0;
   }
   itor = s.erase (itor);
   return true;
  }
 }
 if(s.end()->taskId==taskID)
 {
  for(int i=s.end()->addr;i<s.end()->size;i++)
  {
   SIZE[i] = 0;
  }
  s.erase (s.end ());
  return true;
 }
 return false;
}
subAreaNode searchAreaNode (int addr, int size)
{
 subAreaNode a;
 ostringstream ss;
 string taskID;
```

```
 ss << Fitstring << Fitcount;
 Fitcount++;
 taskID = ss.str ();
 for (int i = addr; i < size + addr; i++)
 {
  SIZE[i] = 1;
 }
 a.addr = addr;
 a.size = size;
 a.taskId =taskID;
 a.end = addr + size - 1;
 return a;
}
bool searchNullArea (int a)
{
 list<NULLArea>::iterator itor;
 for (itor = n.begin (); itor != n.end (); ++itor)
 {
  if (itor->size >= a)
  {
   for (int i = itor->begin; i < itor->begin + a; i++)
   {
    SIZE[i] = 1;
    }
   s.push_back (searchAreaNode (itor->begin, a));
   itor->begin = itor->begin + a;
   itor->size = itor->size - a;
   return true;
  }
 }
 return false;
}
bool searchNullArea (int a,int &number)//为nextfit重载的函数
{
 list<NULLArea>::iterator itor;
 for (itor = n.begin (); itor != n.end (); ++itor)
 {
  if (itor->size >= a)
  {
   for (int i = itor->begin; i < itor->begin + a; i++)
   {
    SIZE[i] = 1;
   }
   number= itor->ID;
   s.push_back (searchAreaNode (itor->begin, a));
   itor->begin = itor->begin + a;
   itor->size = itor->size - a;
   return true;
  }
 }
 return false;
}
void sortforNextFit (int &number)
{
 while (number!=n.begin()->ID)
 {
  n.push_back (n.front());
  n.pop_front ();
 }
}
```

```
void firstFit ()
{
 cout << "------------First Fit------------------" << endl;
 while (1)
 {
  loadNullArea ();
  ShowAllsubAreaNode ();
  cout << "分配前所用的空间内存区域表如下" << endl;
  ShowAllNullArea ();
  int a,b;
  cout << "请输入要分配的地址空间" << endl;
  cin >> a;
  bool c;
  c= searchNullArea (a);
  if (c == true) cout << "分配成功";
  else cout << "分配失败，没有可用空间！";
  cout << "当前空分区表如下" << endl;
  ShowAllNullArea ();
  cout << "是否继续分配?0退出其他继续";
  cin >> b;
  if (b == 0) break;
 }
 cout << "退出First Fit！" << endl;
}
void bestFit ()
{
 cout << "--------------Best Fit-----------------" << endl;
 while (1)
 {
  loadNullArea ();
  ShowAllsubAreaNode ();
  n.sort (comp_BestFit);
  cout << "分配前所用的空间内存区域表如下" << endl;
  ShowAllNullArea ();
  int a, b;
  cout << "请输入要分配的地址空间" << endl;
  cin >> a;
  bool c;
  c = searchNullArea (a);
  if (c == true) cout << "分配成功";
  else cout << "分配失败，没有可用空间！";
  cout << "当前空分区表如下" << endl;
  ShowAllNullArea ();
  cout << "是否继续分配?请输入0退出或其他值继续";
  cin >> b;
  if (b == 0) break;
 }
 cout << "退出Best Fit！" << endl;
}
void worstFit ()
{
 cout << "--------------Worst Fit----------------" << endl;
 while (1)
 {
  loadNullArea ();
  ShowAllsubAreaNode ();
  n.sort (comp_WorstFit);
  cout << "分配前所用的空间内存区域表如下" << endl;
  ShowAllNullArea ();
```

```
  int a, b;
  cout << "请输入要分配的地址空间" << endl;
  cin >> a;
  bool c;
  c = searchNullArea (a);
  if (c == true) cout << "分配成功";
  else cout << "分配失败，没有可用空间！";
  cout << "当前空分区表如下" << endl;
  ShowAllNullArea ();
  cout << "是否继续分配?请输入0退出或其他值继续";
  cin >> b;
  if (b == 0) break;
 }
 cout << "退出Worst Fit！" << endl;
}
void nextFit ()
{
 cout << "--------------next  Fit----------------" << endl;
 int number=1;
 int count;
 while (1)
 {
  loadNullArea ();
  count = n.size ();
  if (number > count) {
   number = 1;
  }
  sortforNextFit(number);
  ShowAllsubAreaNode ();
  cout << "------------------------------------------------" << endl;
  cout << "根据循环首次适应算法本次从第" << number << "块开始寻找"<<endl;
  cout << "分配前所用的空间内存区域表如下" << endl;
  ShowAllNullArea ();
  int a, b;
  cout << "请输入要分配的地址空间" << endl;
  cin >> a;
  bool c;
  c = searchNullArea (a,number);
  number++;
  if (c == true) {
  if (number > count){
   number = 1;
   }
  cout << "分配成功，下次将会从第"<<number<<"块开始查找";
  }
  else cout << "分配失败，没有可用空间！";
  cout << "当前空分区表如下" << endl;
  ShowAllNullArea ();
  cout << "是否继续分配?请输入0退出或其他值继续";
  cin >> b;
  if (b == 0)
  {
   int d;
   cout << "警告:如果退出Next Fit 下次寻找将会从1开始.";
   cout << "确定退出请再次输入0请输入0退出或其他值继续:";
   cin >> d;
   if (d == 0) break;
  }

 }
```

```
 cout << "退出Next Fit! " << endl;
}
```

```
int main ()
{
 cleansize ();
 while (1)
 {
 loadNullArea ();
 int a;
 system ("cls");
 cout << "请输入需要进行的算法或需要进行的操作\n"
   "1.分配内存\n""2.回收内存\n""3.First Fit\n""4.Next Fit\n""5.Best Fir\n""6.Worst Fit\n""7.clean size(删除全部
 cout << "请输入：";
 cin >> a;
 if (a == 1)
 {
  ShowAllsubAreaNode();
  if (n.size () != 0)
  {
   int addr, size;
   string taskId;
   cout << "请输入进程开始地址(或输入-1直接返回):";
   cin >> addr;
   if (addr == -1)  continue;
   cout << "请输入内存大小:";
   cin >> size;
   cout << "请输入作业号(请勿使用Fit作为开头):";
   cin >> taskId;
   s.push_back (setsize (addr, size, taskId));
   loadNullArea ();
   cout << "分配成功"<<endl;
  }
  else cout << "无可用分配空间! " << endl;
  system ("pause");
 }
    if(a==2)
 {
  ShowAllsubAreaNode ();
  string taskId;
  cout << "请输入作业号(或输入-1直接返回):";
  cin >> taskId;
  if (taskId == "-1") continue;
  cleanAREA (taskId);
  system ("pause");
 }
  if (a == 3) {
  firstFit ();
  system ("pause");
 }
  if(a==4){
  nextFit ();
  system ("pause");
 }
  if(a==5){
  bestFit ();
  system ("pause");
 }
```

```
 }
  if(a==6){
  worstFit ();
  system ("pause");
 }
  if (a == 7) {
  int a7;cout << "确认删除么?输入0删除\n请输入:";cin >> a7;
  if (a7 != 0) continue;
  s.clear ();
  n.clear ();
  cleansize ();
  Fitcount = 0;
  cout << "清除成功";
  system ("pause");
 }
    if (a == 8)
 {
  int num = 0;
  for (int i = 0; i < MAX; i++)
   cout << SIZE[i];
  for (int i = 0; i < MAX; i++)
  {
   if (SIZE[i] == 1) {
    num++;
   }
  }
  cout << num << endl;
  system ("pause");
 }
   if (a == 9) {
  exit (0);
  system ("pause");
 }


 }
 return 0;
}
```