中南大学

操作系统综合课程设计



**学生姓名** 马福龙

**学 号** 0902150310

**专业班级** 计科1504

**指导教师** 刘丽敏

**学 院** 信息科学与工程学院

**完成时间** 2018年4月

目录

[第一章 概述 3](#_Toc511852219)

[1.1 项目背景 3](#_Toc511852220)

[1.2 编写目的 3](#_Toc511852221)

[1.3 开发环境 3](#_Toc511852222)

[第二章 需求分析 4](#_Toc511852223)

[2.1问题陈述 4](#_Toc511852224)

[2.2功能分析 4](#_Toc511852225)

[第三章 软件功能设计 5](#_Toc511852226)

[4.1 模块描述 5](#_Toc511852227)

[4.2 模块实现 5](#_Toc511852228)

[第四章 界面设计 10](#_Toc511852229)

[第五章 结束语 15](#_Toc511852230)

[第六章 参考文献 15](#_Toc511852231)

1. 概述
   1. 项目背景

1)掌握内存分配FF，BF，WF策略及实现的思路；

2)掌握内存回收过程及实现思路；

3)实现内存的申请、释放的管理程序，调试运行，总结.

* 1. 编写目的

目的：了解操作系统内存分配的算法。

设计要求：

1. 定义一个自由存储块链表，按块地址排序，表中记录块的大小。当请求分配内存时，扫描自由存储块链表，址到找到一个足够大的可供分配的内存块，若找到的块大小正好等于所请求的大小时，就把这一块从自由链表中取下来，返回给申请者。若找到的块太大，即对其分割，并从该块的高地址部分往低地址部分分割，取出大小合适的块返回给申请者，余下的低地址部分留在链表中。若找不到足够大的块，就从操作系统中请求另外一块足够大的内存区域，并把它链接到自由块链表中，然后再继续搜索。

释放存储块也要搜索自由链表，目的是找到适当的位置将要释放的块插进去，如果被释放的块的任何一边与链表中的某一块临接，即对其进行合并操作，直到没有合并的临接块为止，这样可以防止存储空间变得过于零碎。

1. 空闲区采用分区说明表的方法实现（1）中的功能。要求同上。
   1. 开发环境

系统环境：win10

开发IDE：intellij idea17.10

1. 需求分析

2.1问题陈述

（1） 定义一个自由存储块链表，按块地址排序，表中记录块的大小。当请求分配内存时，扫描自由存储块链表，址到找到一个足够大的可供分配的内存块，若找到的块大小正好等于所请求的大小时，就把这一块从自由链表中取下来，返回给申请者。若找到的块太大，即对其分割，并从该块的高地址部分往低地址部分分割，取出大小合适的块返回给申请者，余下的低地址部分留在链表中。若找不到足够大的块，就从操作系统中请求另外一块足够大的内存区域，并把它链接到自由块链表中，然后再继续搜索。

释放存储块也要搜索自由链表，目的是找到适当的位置将要释放的块插进去，如果被释放的块的任何一边与链表中的某一块临接，即对其进行合并操作，直到没有合并的临接块为止，这样可以防止存储空间变得过于零碎。

（2） 空闲区采用分区说明表的方法实现（1）中的功能。要求同上。

2.2功能分析

由问题陈述及需求设计6个模块

1 内存初始化模块

功能描述：主要用来初始化内存空间，并设置为空闲状态。

2 内存申请首次适应算法（First Fit）模块

功能描述：从空闲分区表的第一个表目起查找该表，把最先能够满足要求的空闲区分配给作业，这种方法目的在于减少查找时间。为适应这种算法，空闲分区表（空闲区链）中的空闲分区要按地址由低到高进行排序。该算法优先使用低址部分空闲区，在低址空间造成许多小的空闲区，在高地址空间保留大的空闲区。。

3 内存申请最佳适应算法（Best Fit）模块

功能描述：从全部空闲区中找出能满足作业要求的、且大小最小的空闲分区，这种方法能使碎片尽量小。为适应此算法，空闲分区表（空闲区链）中的空闲分区要按从小到大进行排序，自表头开始查找到第一个满足要求的自由分区分配。该算法保留大的空闲区，但造成许多小的空闲区

4 内存申请最差适应算法（Worst Fit）模块

功能描述：从全部空闲区中找出能满足作业要求的、且大小最大的空闲分区，从而使链表中的结点大小趋于均匀，适用于请求分配的内存大小范围较窄的系统。为适应此算法，空闲分区表（空闲区链）中的空闲分区要按大小从大到小进行排序，自表头开始查找到第一个满足要求的自由分区分配。该算法保留小的空闲区，尽量减少小的碎片产生。

5 内存返还模块

功能描述：通过输入地址空间首地址返还内存，并设置为空闲状态。

6 随机初始化模块

功能描述：主要是用来对三种算法进行可视化对比。

1. 软件功能设计

4.1 模块描述

由问题陈述及需求设计6个模块

1 内存初始化模块

功能描述：主要用来初始化内存空间，并设置为空闲状态。

2 内存申请首次适应算法（First Fit）模块

功能描述：从空闲分区表的第一个表目起查找该表，把最先能够满足要求的空闲区分配给作业，这种方法目的在于减少查找时间。为适应这种算法，空闲分区表（空闲区链）中的空闲分区要按地址由低到高进行排序。该算法优先使用低址部分空闲区，在低址空间造成许多小的空闲区，在高地址空间保留大的空闲区。。

3 内存申请最佳适应算法（Best Fit）模块

功能描述：从全部空闲区中找出能满足作业要求的、且大小最小的空闲分区，这种方法能使碎片尽量小。为适应此算法，空闲分区表（空闲区链）中的空闲分区要按从小到大进行排序，自表头开始查找到第一个满足要求的自由分区分配。该算法保留大的空闲区，但造成许多小的空闲区

4 内存申请最差适应算法（Worst Fit）模块

功能描述：从全部空闲区中找出能满足作业要求的、且大小最大的空闲分区，从而使链表中的结点大小趋于均匀，适用于请求分配的内存大小范围较窄的系统。为适应此算法，空闲分区表（空闲区链）中的空闲分区要按大小从大到小进行排序，自表头开始查找到第一个满足要求的自由分区分配。该算法保留小的空闲区，尽量减少小的碎片产生。

5 内存返还模块

功能描述：通过输入地址空间首地址返还内存，并设置为空闲状态。

6 随机初始化模块

功能描述：主要是用来对三种算法进行可视化对比。

4.2 模块实现

#### 内存初始化模块

public MyMemory() {

init();

}

public static void init(){

myArray=new LinkedList<Mem>();

Mem mem=new Mem(0,GCON.totalMemory,true);

myArray.add(mem);

}

#### 内存申请首次适应算法（First Fit）模块

/\*\*

\* 申请新的内存块 FF申请方式

\* @param length 要申请块的大小

\* @return 成功返回true,申请失败返回false

\*/

public static boolean getNewMem(int length){

if(length>GCON.totalMemory||length<=0)

return false;

for(int i=0;i<myArray.size();i++){

if(myArray.get(i).state&&myArray.get(i).length>=length){

myArray.add(i+1,new Mem(myArray.get(i).start+myArray.get(i).length-length,length,false));

myArray.get(i).length-=length;

if(myArray.get(i).length==0)

myArray.remove(i);

return true;

}

}

return false;

}

#### 内存申请最佳适应算法（Best Fit）模块

/\*\*

\* BF方式申请内存

\* @param length

\* @return

\*/

public static boolean getNewMemBF(int length){

if(length>GCON.totalMemory||length<=0)

return false;

int index=0,minLen=GCON.totalMemory+1;

for(int i=0;i<myArray.size();i++){

if(myArray.get(i).state&&myArray.get(i).length>=length&&myArray.get(i).length<=minLen){

index=i;

minLen=myArray.get(i).length;

}

}

if(minLen==GCON.totalMemory+1)

return false;

if(minLen>=length){

myArray.add(index+1,new Mem(myArray.get(index).start+myArray.get(index).length-length,length,false));

myArray.get(index).length-=length;

// System.out.println(index+" "+myArray.get(index).length);

if(myArray.get(index).length==0)

myArray.remove(index);

return true;

}

return false;

}

#### 内存申请最差适应算法（Worst Fit）模块

/\*\*

\* WF方式申请内存

\* @param length

\* @return

\*/

public static boolean getNewMemWF(int length){

if(length>GCON.totalMemory||length<=0)

return false;

int index=0,maxLen=0;

for(int i=0;i<myArray.size();i++){

if(myArray.get(i).state&&myArray.get(i).length>=maxLen){

index=i;

maxLen=myArray.get(i).length;

}

}

if(maxLen<length)

return false;

else{

myArray.add(index+1,new Mem(myArray.get(index).start+myArray.get(index).length-length,length,false));

myArray.get(index).length-=length;

if(myArray.get(index).length==0)

myArray.remove(index);

return true;

}

}

#### 内存返还模块

/\*\*

\* 给内存的首地址，删除这个首地址的内存，就是把这个内存的占用状态改为空闲状态，并紧凑

\* @param start 内存首地址

\* @return 成功返回true，失败返回false

\*/

public static boolean delMem(int start){

for(int i=0;i<myArray.size();i++){

if(myArray.get(i).start==start&&myArray.get(i).state==false){

myArray.get(i).state=true;

//往上紧凑

if(i<myArray.size()-1&&myArray.get(i+1).state){

myArray.get(i).length+=myArray.get(i+1).length;

myArray.remove(i+1);

}

//往下紧凑

if(i>0&&myArray.get(i-1).state){

myArray.get(i-1).length+=myArray.get(i).length;

myArray.remove(i);

}

return true;

}

}

return false;

}

#### 随机初始化模块

/\*\*

\* 随机初始化

\*/

public static void initNew(int[] arr1){

arr=arr1;

Random ran=new Random();

if(myArray!=null)

myArray.clear();

boolean flag=true;

int start=0;

for(int i=0;i<arr.length;i++){

int t=arr[i];

if(start+t<GCON.totalMemory-1){

myArray.add(new Mem(start,t,flag));

}else{

myArray.add(new Mem(start,GCON.totalMemory-start,flag));

start+=t;

break;

}

flag=!flag;

start+=t;

}

if(start<GCON.totalMemory-1)

myArray.add(new Mem(start,GCON.totalMemory-start,flag));

}

}

1. 界面设计



Figure 1 开始界面



Figure 2 选择内存分配方式



Figure 3 内存申请



Figure 4 内存返还



Figure 5 清除内存



Figure 6 内存随机初始化



Figure 7 更改总内存大小



Figure 8 清空面板信息

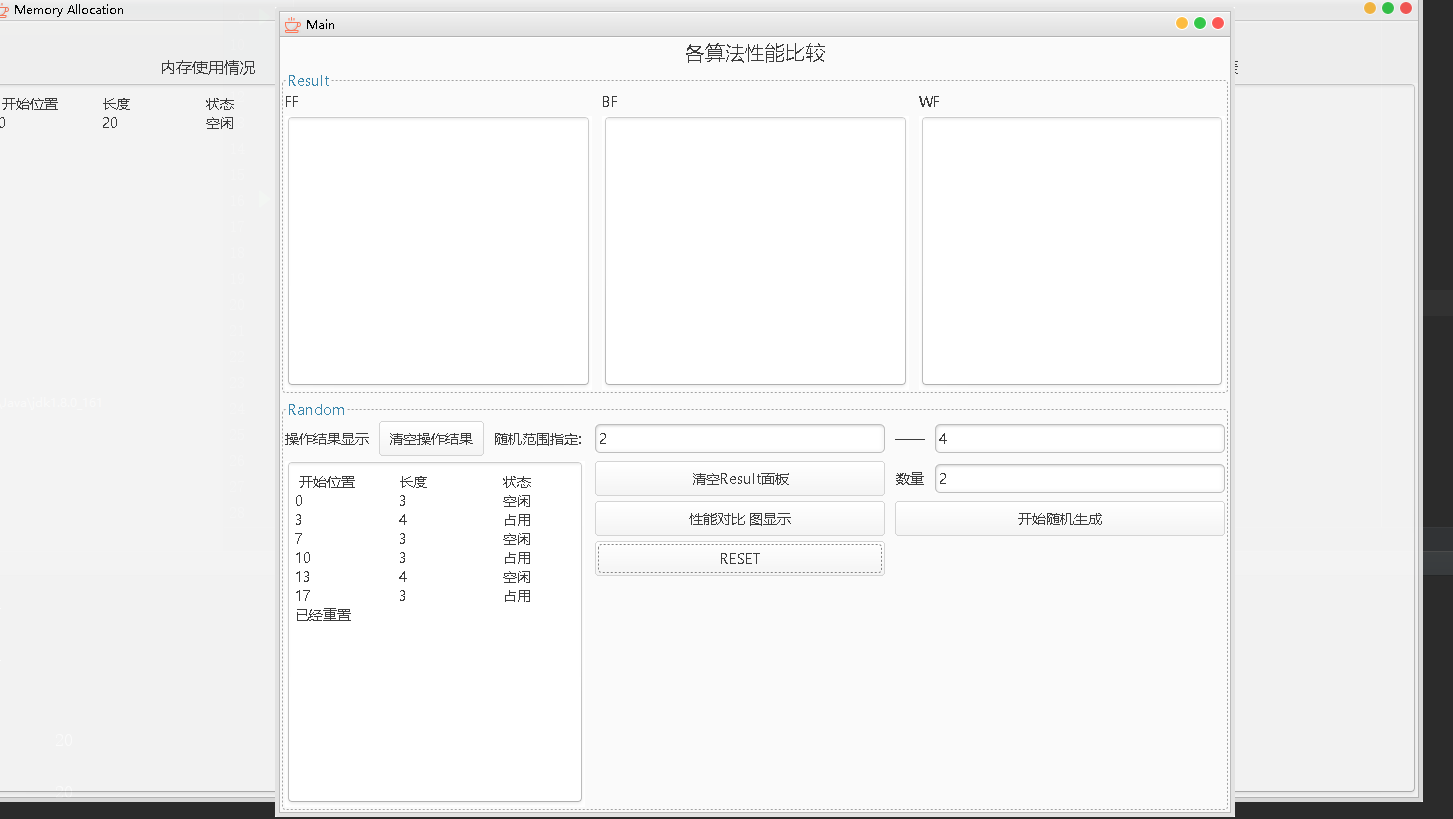


Figure 9 三种算法对比



Figure 10 对比结果

1. 结束语

对于此次的操作系统课程的课程设计，加深了对大学所学的知识，但是由于本人缺乏系统的开发实际经验，对系统的分析还不够彻底，存在了很多缺陷，页面不够美观，没有专业的绘图知识，对代码的运用也不能够很熟的掌握，程序上也有很多需要改进的地方，在未来的日子里，还得要不断学习这方面知识，加深代码编写能力，吸收新的知识，提高自己的工作能力。

通过内存分配算法的编写，加强了我开发系统的能力，对于大学所学的知识又重新加深了了解，是一次很好的学习机会，特别感谢刘丽敏老师的指导！

1. 参考文献

[1] 王珊 萨师煊 数据库系统概论(第4版) 高等教育出版社，2006

[2] 彭伟民.基于需求的酒店管理系统的建模与实现.微机发展，2005.10