哈 尔 滨 理 工 大 学

课　程　设　计

（ 操 作 系 统 ）

题　　目：伙伴系统

班　　级：计算机科学与技术　计 15-5 班

姓　　名：孙喜洋 1504010522

指导教师：尹　芳

系 主 任：林 克 正

2017　年 09 月 01 日

目　　录

1. 题目分析

1.1 题目要求

Linux 中内存分配的伙伴堆算法模拟。
（1）模拟内存实时情况。
（2）实现 Buddy　heap 算法。
（3）通过键盘输入随机产生的申请和释放操作。
（4）每次申请或释放都显示实时的内存分配的对比图

1.2 功能分析

利用数组来实现伙伴系统算法的 3 种功能，定义一个空闲分区数组和已分配分区数组，这 2 个数组都为 2 维数组，分别记录空闲分区块和已分配分区块的大小和内存地址。这样，在实现伙伴系统算法的 3 种功能就转化为对数组的操作。
⑴ 分配内存：修改空闲分区数组，模拟按照伙伴系统算法思想划分合适分区进行分配，然后添加到已分配分区数组。
⑵ 回收内存：修改空闲分区数组，模拟按照伙伴系统算法思想回收内存，然后从已分配分区数组中删除该内存块。
⑶ 输出内存使用情况：输出空闲分区数组和已分配分区数组

2. 数据结构

模拟伙伴堆算法的问题中涉及的数据结构包括申请内存的大小，名字（为了直观表示），需要释放的内存的首地址，当前占用的内存数目，大小，每块内存起止地址，名字

为了清晰实现伙伴算法的模拟，我们在该项目的实现模拟的是 1024 大小的存储空间，我们在程序中用伪代码表示如下：

```
/*  全局变量的声明    */
var buddy = new Array("2","4","8","16","32","64","128","256","512","1024"); //分区大小 2 的k 次幂
var free1 = new Array("0","0","0","0","0","0","0","0","0","1"); //空闲分区的个数    初始内存大小1024
var use =new Array(100);  //已分配分区表  最多为100 个进程分配 记录已分配分区大小，内存地址
var str   = new Array(100);
for (var per = 0; per <= 100; per++)
{
      use[per] = new Array(2);
}
var free_addr = new Array(10); //空闲分区的首地址 [i][j]表示2 的i 次方大小的空闲分区的第j 个分区的首地址
for (var per = 0; per <= 10; per++)
{
       free_addr[per] = new Array(50);
}
free_addr[9][0] = 0;
var maxsize=9;  //最大空闲分区 初始为1024
var usenum=0;    //进程数 初始为0
```

为了实现这些数据结构，用 Javascript 语言定义功能函数如下：

```
function applyin()：/*  申请内存函数  */
function releaseout()/*  释放内存函数 */
function show()/*  显示内存占用和剩余情况 */
function shutwin()/*  关闭  */
function reset()/*  刷新    */
```

3. 流程图

3.1 前导界面流程图

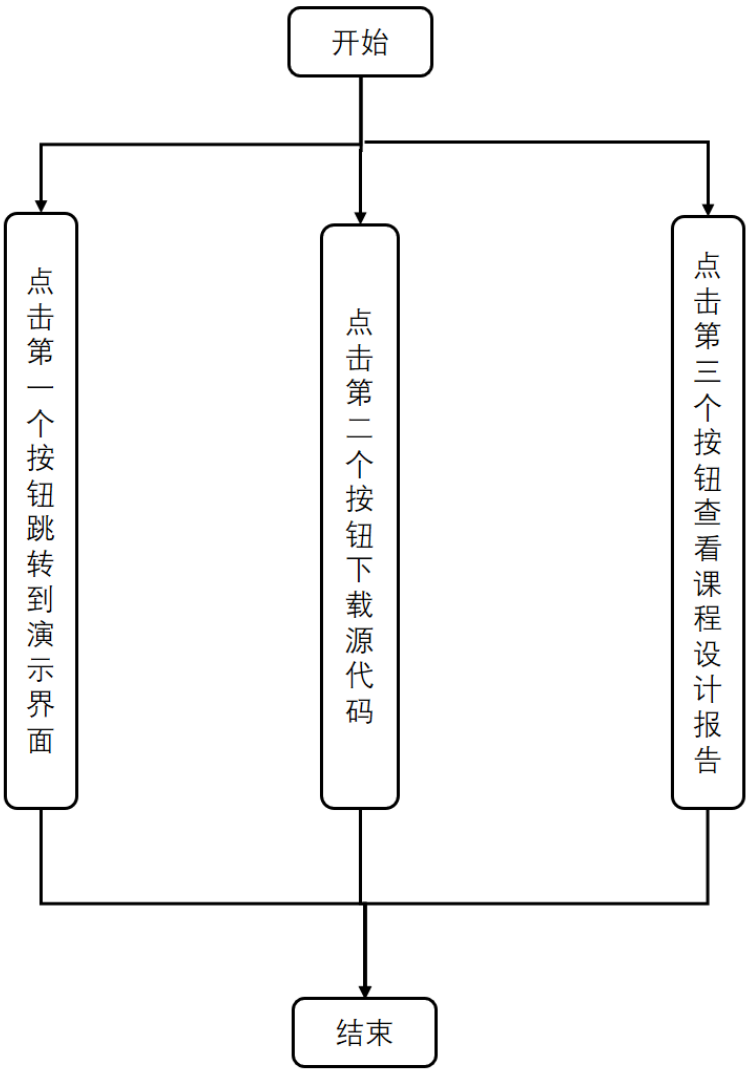

图 1 前导界面流程图

3.2 演示界面流程图

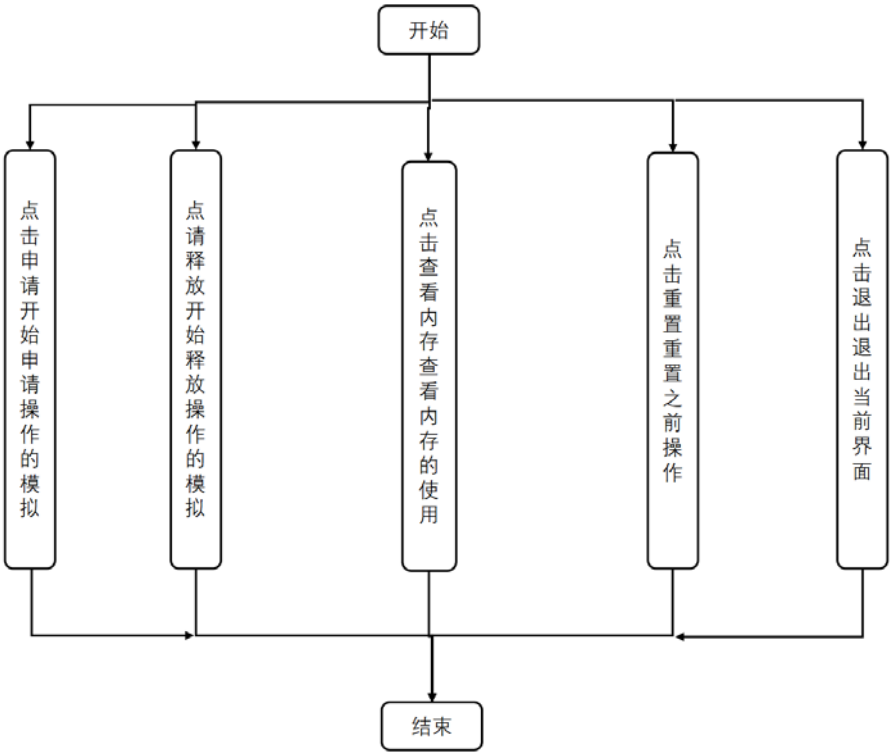

图 2 演示界面流程图

3.3 演示界面——申请内存流程图

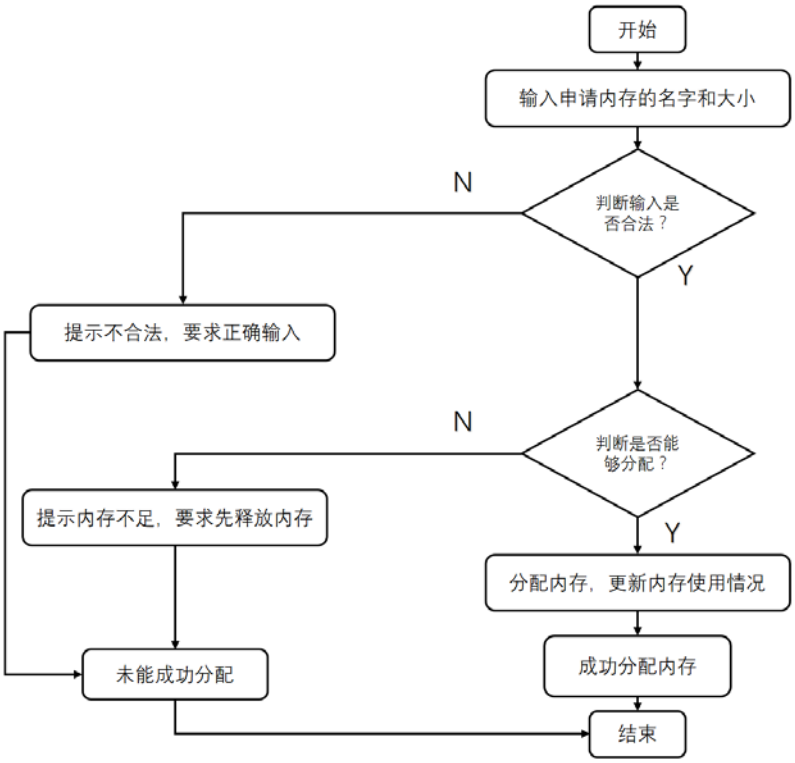

图 3 申请内存流程图

3.4 演示界面——释放内存流程图

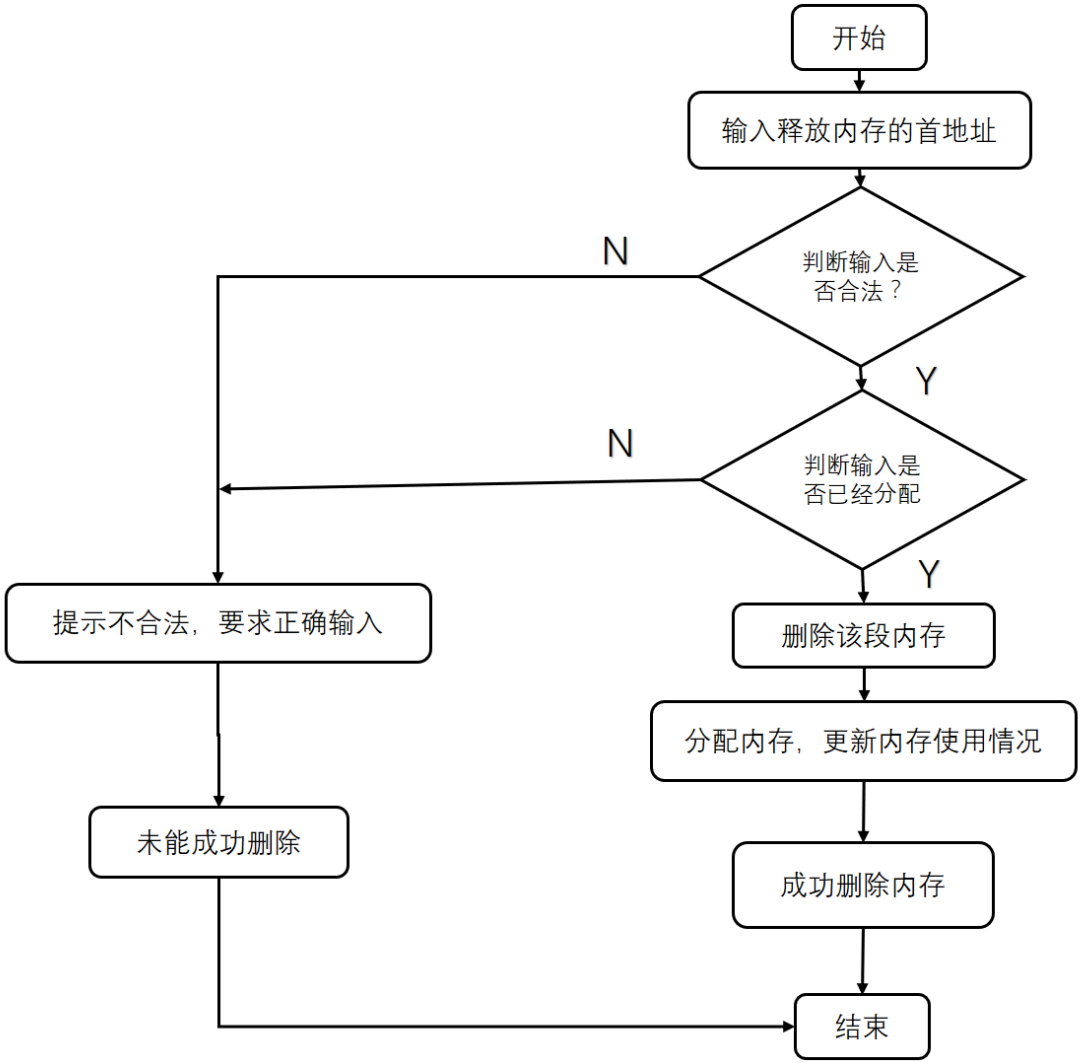

图 4 释放内存流程图

4. 实现技术

4.1 工具环境

为实现上述设计，采用 web 语言，Sublime Text3 开发环境。具体采用的技术如下：

（1）HTML 实现整体元素显示

（2）CSS 实现样式的构建

（3）Javascript / JQuery 事件动态的获取

（4）Chrome 浏览器进行调试

4.2 前导界面部分

4.2.1 前导界面静态

图 5 整体界面-1

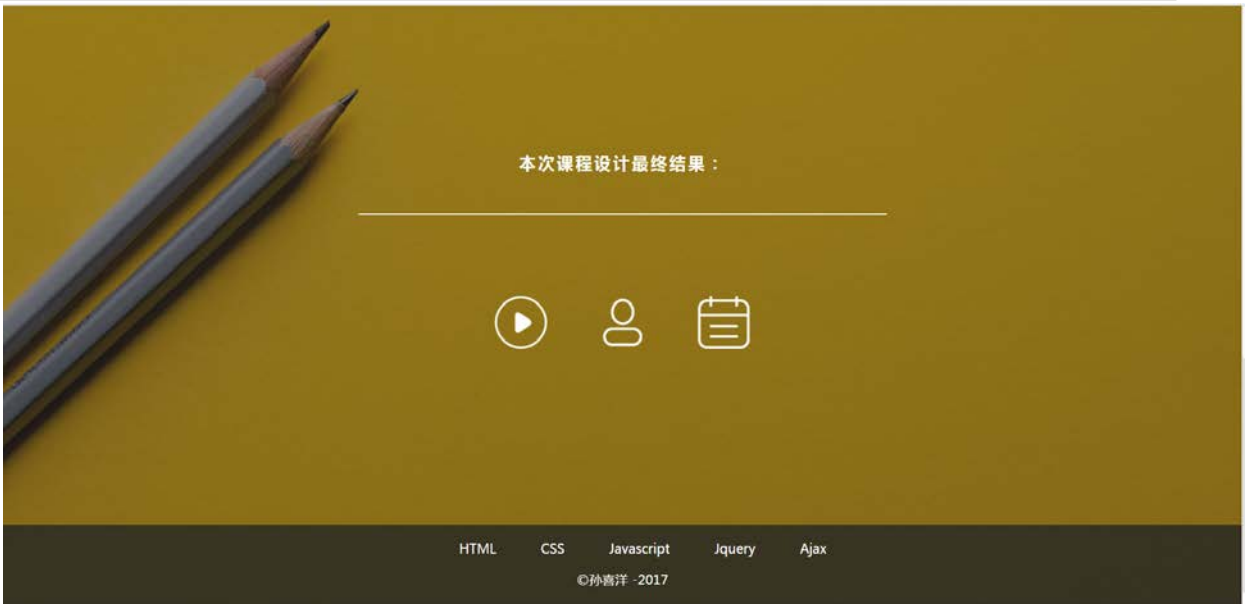

图 6 整体界面-2

4.2.2 前导界面动态

图 7 前导界面动态效果

设计网页样式以及功能，在 HTML 文件中添加标签，定义不同的类名，以便于 CSS 文件中样式的设定，为不同的标签对象设置不同的点击事件，设置点击事件响应。

```
<div class = "content"><!-- 内容 -->
    <section class="green-section">
        <div class="wrap">
            <h2>本次课程设计最终结果：</h2>
            <div class="hr"></div>
        <div class="icon-group">
            <a class = "icon" href="test.html"><img class = "ico-img" src="icon1.png" alt=""></a>
            <a class = "icon" href=""><img class = "ico-img" src="icon2.png" alt=""></a>
            <a class = "icon" onclick = "print_onclick('small.pdf')"><img class = "ico-img" src="icon3.png" alt=""></a>
        </div>
        </div>
    </section>
</div>
```

图 8 添加图标段 HTML 文件

```
<script>
  function print_onclick(url)
{
var a=window.open(url, "_blank","top=0,left=0,height=900,width=1520,status=yes,toolbar=1,menubar=no,location=no,scrollbars=yes");
}
</script>
```

图 9 添加点击响应 js 文件

4.3 演示部分

4.3.1 演示界面整体

点击播放按钮，开始进入演示界面，如图 10 所示。

图 10 演示界面效果

图 11 演示界面鼠标碰触动态效果

通过设置伪类 hover 的样式 当鼠标触碰该 li 标签时，改变 li 样式

```
li:hover {
    background: rgba(50, 48, 48, 0.6);
}
```

图 12 添加鼠标动态 css 文件

4.3.2 申请内存

1. 合法性检查

判断输入申请的内存大小、名字不能为空，内存大小在 0-1024 之间，不合法弹出对应的提示框并将鼠标移动正确的位置，如图 13， 图 14，图 15 所示。

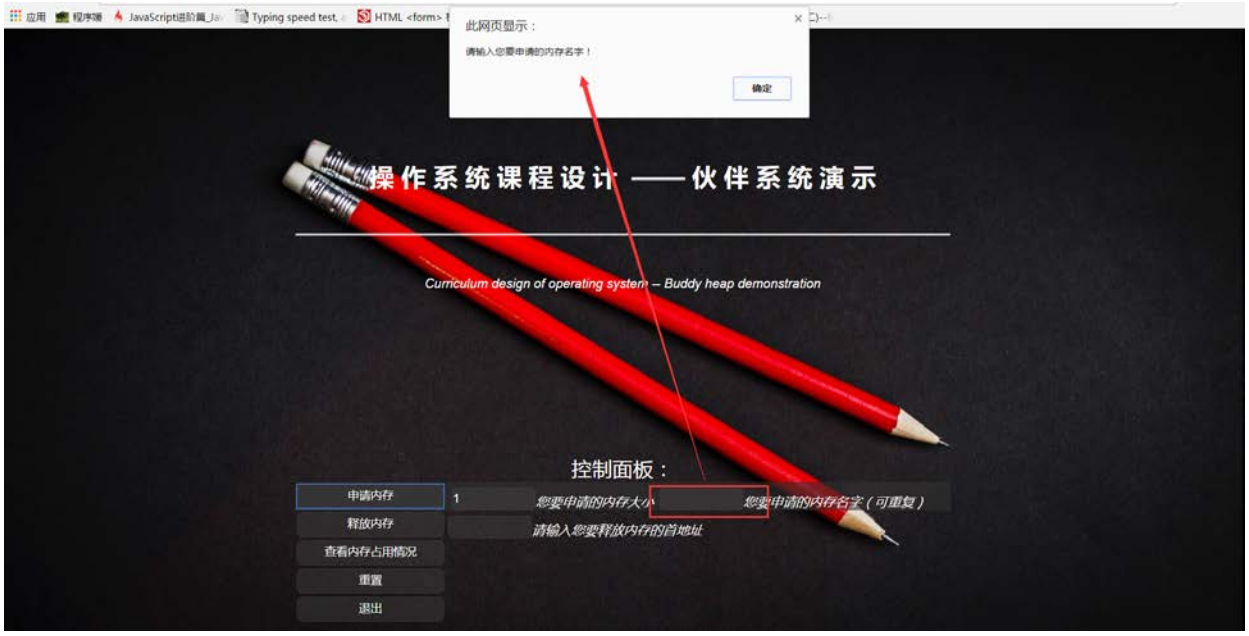

图 13 没输入名字的提示框

图 14 没输入大小的提示框

图 15 输入不在范围的提示框

```
if (input.value=='')
{
      alert("请输入您要申请的内存大小！");
      document.getElementById("apply-size").focus();
}
else  if (name == '')
{
      alert("请输入您要申请的内存名字！");
      document.getElementById("apply-name").focus();
}

else
{
        var size = parseInt(input.value);
        if(size<1 || size>1024)
       {
           alert("请输入1~1024之间的数字！");
           document.getElementById("apply-size").focus();
       }
```

图 16 添加合法性检查 js 文件

2. 申请内存

正确的输入，提交后，会出现申请内存成功的提示框，如图 17 所示。

图 17 申请成功的提示框

在 js 文件中读入输入的申请内存的大小，和名字，在 funtion apply-in 函数中，首先确定分配内存的大小，确定分配的起止地址，更新此时内存状况和此时能过最大存储的空间如图 18 所示。

```
for(i=0; i<10; i++)   ///寻找合适的空闲分区 从小的开始找
{
    if(free1[i]!=0 && buddy[i]>=size)
        break;
}
usepage=i;
free1[i]--;
for(i=0; i<10; i++)    /// 寻找分配的大小
{
    if(buddy[i]>=size)
        break;
}
str[usenum] = name;
console.log('str[usenum]' + str[usenum])
use[usenum][0]=i;   ///记录分配给该进程的空闲区大小 2的i次方
use[usenum][1]=free_addr[usepage][free1[usepage]];  /// 内存起始地址
remain=buddy[usepage]-buddy[use[usenum][0]];  ///剩余大小
start=free_addr[usepage][free1[usepage]]+buddy[usepage]-buddy[usepage-1]; ///空闲区首地址
usenum++;
for(i=usepage-1; i>=0; i--) ///确定其他空闲分区的首地址
{
    if(remain>=buddy[i])
    {
        free1[i]++;
        free_addr[i][free1[i]-1]=start;
        remain-=buddy[i];    ///剩余大小
        start-=buddy[i-1];   ///首地址
    }
    if(remain==0)
        break;
}
if(free1[maxsize]==0)   ///确定最大空闲分区
{
    maxsize=0;
    for(i=9; i>=0; i--)
    {
        if(free1[i]!=0)
        {
            maxsize=i;
            break;
        }

   }
}
```

图 18 添加申请内存的 js 文件

如果此时没有要求申请的内存空间可以分配，则提示要求先释放内存，才能能过申请，如图 19 所示。

图 19 空间不够的提示框

```
}
else if( size >  parseInt(buddy[maxsize]))
{
    alert("没有足够的内存空间，请先释放！") ;
     document.getElementById("release-size").focus();
}
else
```

图 20 添加是否可以分配检查 js 文件

4.3.3 释放内存部分

1 .合法性检测

图 21 释放首地址为空的提示框

图 22 释放首地址错误的提示框

```
if (input.value=='')
{
    alert("请输入您要释放内存的首地址！");
    document.getElementById("release-size").focus();
}
else
{
        for(i=0,flag=0; i<usenum; i++) //匹配已分配首地址
    {
        if(free_start==use[i][1])
        {
            free_start=i;
            flag=1;
            break;
        }
    }
    if(flag==0)
    {
        alert("输入的首地址不存在，请重新输入！\n");
    }
```

图 23 释放首地址检查的 js 文件

2. 释放内存

正确输入后，在 js 中调用 function release-out() 函数如图 24 所示，执行寻找是否否能够有伙伴空闲能过合并，释放该段内存，更新此时内存状况和此时能够存储最大的空间，提示释放内存成功如图 25 所示。

```
    for(i=use[free_start][0]; i<9; i++) //寻找是否有伙伴可以合并
    {
        for(j=0,flag=0; j<free1[i]; j++)
        {
            if(use[free_start][1]+buddy[use[free_start][0]]==free_addr[i][j])
            {
                // 左合并  伙伴内存地址高
                use[free_start][0]++;   //往上去找  还能合并否
                free1[i]--;
                for(k=0; k<free1[i]; k++) // 伙伴的地址被覆盖
                    free_addr[i][j]=free_addr[i][j+1];  //
                flag=1;
                break;
            }
            if(use[free_start][1]-buddy[use[free_start][0]]==free_addr[i][j])
            {
                // 右合并  伙伴内存地址为低
                use[free_start][0]++;
                use[free_start][1]=free_addr[i][j];
                free1[i]--;
                for(k=0; k<free1[i]; k++) // 伙伴的地址被覆盖
                    free_addr[i][j]=free_addr[i][j+1];//
                flag=1;
                break;
            }
            if(flag==0 && j==free1[i]-1)
            {
                // 不能合并
                free1[use[free_start][0]]++;
                free_addr[use[free_start][0]][free1[use[free_start][0]]-1]=use[free_start][1];
                f=1;
                break;
            }
        }

        if(flag==1)
            continue;
        else
            break;
    }
    if(f==0)
    {
        free_addr[use[free_start][0]][free1[use[free_start][0]]]=use[free_start][1];//添加到空闲分区
        free1[use[free_start][0]]++;
    }
    usenum--;
    for(i=free_start; i<usenum+1; i++) // 清除占用的内存
    {
        use[i][0]=use[i+1][0];
        use[i][1]=use[i+1][1];
    }
    alert("释放内存成功！");
}
```

图 24 添加释放内存的 js 文件

图 25 释放内存成功的提示框

4.3.4 查看内存使用部分

图 26 查看内存使用界面

当点击查看内存占用情况按钮时，js 文件调用 function check（） 函数，形成一个背景颜色较深的遮罩层，在遮罩层动态画出此时内存使用情况的表格，如图 27 所示。

```
// 1 .  遮罩层  在show-memory 添加  oMask

/// 遮罩层

    //获取页面宽 高
    var sHeight = document.documentElement.scrollHeight;
    var sWidth = document.documentElement.scrollWidth;
    // 获取可视区域
    var wHeight = document.documentElement.clientHeight;

    var oMask = document.createElement("div");
    oMask.id = "mask";
    document.body.appendChild(oMask);
    oMask.style.width = sWidth + "px";
    oMask.style.height = sHeight + "px";

/// table_wrap

    var table_wrap = document.createElement("div");
   table_wrap.id = "situation";

   table_wrap.innerHTML = "<p style = 'font-size:25px;font-style: bold; font-family:Arial; color:white; text-align: center'>此时内存使用情况：</p>";
    //原宿 宽高
   document.body.appendChild(table_wrap);///顺序
   var dHeight = table_wrap.offsetHeight;//已有的
   var dWidth = table_wrap.offsetWidth;
    table_wrap.style.left = (sWidth - dWidth)/2 + "px";
    table_wrap.style.top = (wHeight - dHeight)/2 + "px";
   // gai
    oMask.onclick = function()
    {
      document.body.removeChild(oMask);
      document.body.removeChild(table_wrap);
      document.body.removeChild(table);
    }
```

图 27 添加遮罩层 js 文件

```
    var start = 0,ct=1;
    var  i=0;
    while(true)
     {
        if (i == usenum)break;
        //  js 添加 HTML
        if (use[i][1] > start)
        {
          var tr = document.createElement('tr');
          var td1 = document.createElement('td');
          var td2 = document.createElement('td');
          var td3 = document.createElement('td');
          var td4 = document.createElement('td');
          // 在容器元素中放入其他元素
           table.appendChild(tr);
           tr.appendChild(td1);
           tr.appendChild(td2);
           tr.appendChild(td3);
           tr.appendChild(td4);
          // 文本
            td1.innerHTML = ct++;
            td2.innerHTML = "空闲";
            td3.innerHTML = "*";
            var addend = parseInt(use[i][1]) -1;
            td4.innerHTML = start + '-' + addend;
           // printf("第 %d 个；状态：空闲 内存起止地址：%4d - %4d \n",ct++,start,use[i][1]-1);
            start = use[i][1];

        }
        if (use[i][1] == start)
        {
          var tr = document.createElement('tr');
          var td1 = document.createElement('td');
          var td2 = document.createElement('td');
          var td3 = document.createElement('td');
          var td4 = document.createElement('td');
           // 在容器元素中放入其他元素
            table.appendChild(tr);
            tr.appendChild(td1);
            tr.appendChild(td2);
            tr.appendChild(td3);
            tr.appendChild(td4);
          // 属性
          // 样式
          // 文本

            td1.innerHTML = ct++;
            td2.innerHTML = "占用";
            td3.innerHTML = str[i];
            var addend = parseInt(use[i][1]) + parseInt(buddy[use[i][0]]-1);
            td4.innerHTML = use[i][1] + '-' + addend;
            //printf("第 %d 个；状态： 内存起止地址：%4d -%4d\n",ct++,use[i][1],use[i][1]+buddy[use[i][0]]-1);
            start = addend +1;
```

图 28 添加弹出层 js 文件

当点击遮罩层非表格部分，会删除，表格删除遮罩层，实现关闭表格，返回控制面板的效果，如 29 所示。

```
oMask.onclick = function()
{
  document.body.removeChild(oMask);
  document.body.removeChild(table_wrap);
  document.body.removeChild(table);
}
```

图 29 添加点击遮罩层关闭查看的 js 文件

图 30 返回控制面板的界面

4.3.5 重置部分

点击重置部分，JS 文件调用 function reset() 函数，将重置之前操作，弹出“刷新成功!”提示框如图 31 所示。

图 31 弹出刷新成功的提示框

```
/* 刷新   */
function reset()
{
   history.go(0);
   alert("刷新成功！");
}
```

图 32 添加重置功能的 js 文件

4.3.6 退出部分

点击退出按钮，js 文件调用 function shutwin() 函数，弹出询问是否关闭提示框，点击关闭，可以关闭本窗口，如图 33 所示。

图 33 弹出确认关闭的提示框

```
/* 关闭   */
function shutwin()
{
  var cmd = confirm("您确定要关闭本窗口？");
  if (cmd == true)
  {
   window.close();
  }
  else
  {

  }
  return ;
}
```

图 34 添加关闭窗口功能的 js 文件

4.4 源代码分享页面

点击第二个个人按钮，进入源代码资源分享页面，如图 34 所示。

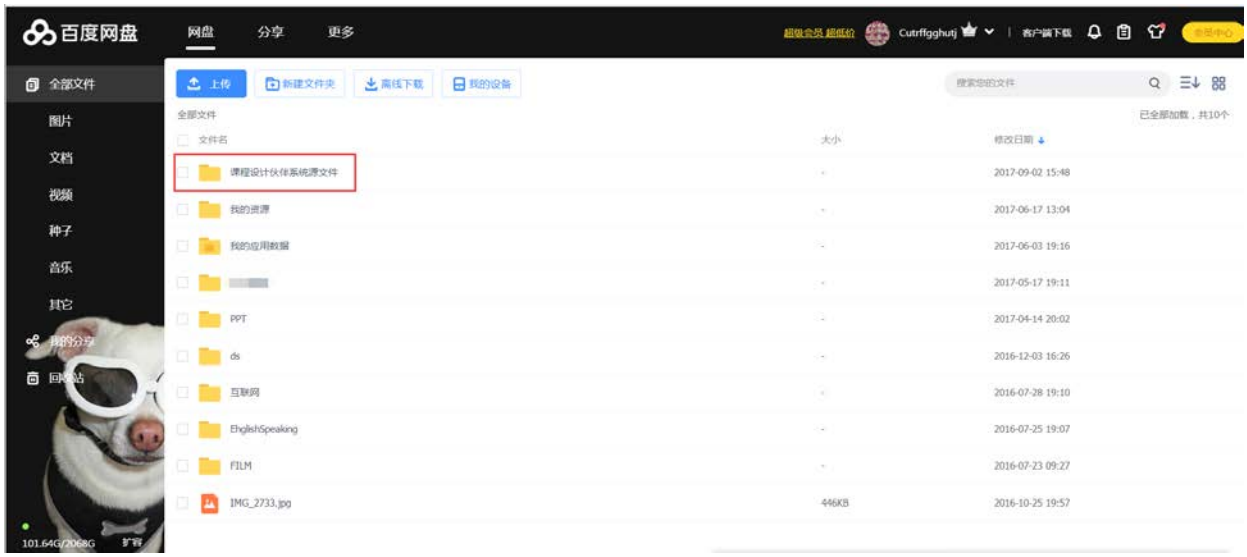

图 34 源代码分享页面

4.5 课程设计报告

点击第三个文档按钮，如图 35 所示，打开课程设计报告。

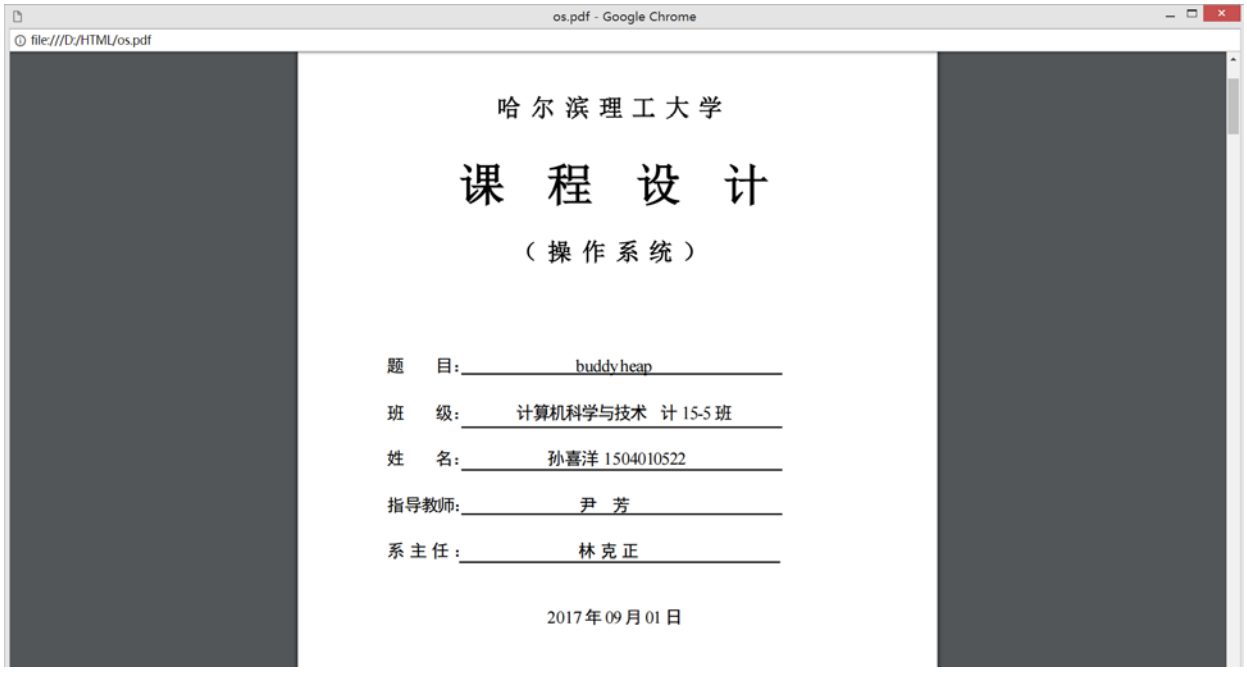

图 35. 打开课程设计报告

5. 设计结论和心得

通过本次的课程设计的学习和锻炼，我也是第一次完成相对完善的界面设计，因为老师在暑假之前就布置了课程设计的内容和要求，我在假期自己学习了，web 前端的语言，HTML/CSS/JavaScript，但是在实际实践中，还是遇到了不少的困难，得到了很多的成长。

在最初的显示内存使用状况的设计，我是把已将分配的和空闲的分开显示，并且把空闲内存的自动分成 2 的 i 次幂存储，在和老师沟通的时候，我意识到，这并不符合伙伴系统的工作原理，伙伴系统是有申请内存的时候才进行分配，并不是在申请之前分配好了。意识到了这点后，我马上做出了修改，改成了实时显示内存状态和首地址的显示方法。

最初我做的查看内存使用情况的页面 是在控制面板下面，只能点击查看才能刷新，并不是有很好的使用体验，我就想着点击查看时，打开新的页面查看内存使用，通过查网上的资料和实际案例，我决定用弹出层实现，HTML 将每次申请的内存大小和名称，释放的内存首地址，以及相应的点击操作，传给 JavaScript 文件，在 JavaScript 文件中实现伙伴算法的模拟，然后在 JavaScript 文件中动态创建 HTML 标签，写回到 HTML 中显示出来。

在做引导网页的时候，设置 HTML 中打开 PDF 文件，我试了好多自己的想法，都是显示下载后才能打开，最后我查资料，不断地实践，发现，在 JavaScript 文件调用 window.open()可以实现。

在这次课程设计开始的时候，我十分不确定我是否能够真的完成这次的题目，因为以前没有完整的设计完成界面，我就先看懂实现的算法和数据结构，用 C++在 codeblocks 实现后，我开始用 JavaScript 上实验，当基本界面完成后，我又开始担心，JavaScript 处理数据能否读出到 HTML,我开始实验一个数据，每当完成一点，我就把所有的情况检查一遍，以便及时发现错误，当我实现了初步的功能，真的特别有成就感，就有了信心，不断查资料，学教程，优化界面和一些功能，我觉得就是自己不断的把大问题划分成小的问题，在慢慢去实现小的问题，“不积小步无以至千里”，当这次课程设计结束了，我的也许不是最好，但心里真的有一种程序员的成就感，和对自己的肯定。

通过完成这次的课程设计，我更加熟悉掌握 JavaScript 的相关知识，和具备开发的一些经验，同时，在实践的过程中，我也发现了自己的问题和缺点，我意识到自己 CSS 有很多不熟练的知识，JQuery 的学习太过浅薄，我完成的课程设计在一些地方还存在一些不足，我会带着对自己这份认知，更好更努力的投入到以后的学习和提升中。