河北师范大学软件学院
Software College of Hebei Normal University

操作系统实践

实验07 内存管理（2）

软件学院 基础组

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

1 实验目的

1、加深对动态分区分配内存管理方式的理解

2、理解空闲分区表以及空闲分区链

3、掌握该管理方式的动态内存分配和内存回收的流程

4、理解动态分区分配算法：首次适应算法，循环首次适应算法，最佳适应算法

本节知识点

1 实验目的

2 实验准备

3 实验内容

4 问题思考

2 实验准备

2.1 动态分区分配原理

动态分区分配是根据进程的实际需要，动态地为之分配内存空间。作业装入内存时，把可用内存分出一个连续区域给作业，且分区的大小正好适合作业大小的需要。

2 实验准备

2.2 实现方式

- 分区分配中的数据结构
  - 空闲分区表

①空闲分区表：记录每个空闲分区的情况。每个空闲分区占一个表目。表目中包括：分区序号、分区始址、分区的大小等。

②空闲分区链：在每个分区的起始部分，设置一些用于控制分区分配的信息，以及用于链接各分区所用的前向指针；在分区尾部则设置一后向指针，在分区末尾重复设置状态位和分区大小表目。

2 实验准备

- 已占分区说明表

结构：作业号；起始地址；大小

分区分配算法

(1) 首次适应算法FF

FF算法要求空闲分区表以地址递增的次序排列。在分配内存时，从表首开始顺序查找，直至找到一个大小能满足要求的空闲分区为止；然后按照作业的大小，从该分区中划出一块内存空间分配给请求者，余下的空闲分区仍留在空闲分区表中。若从头到尾不存在满足要求的分区，则分配失败。

优点： 优先利用内存低址部分的内存空间

缺点： 低址部分不断划分，产生小碎片（内存碎块、内存碎片、零头）；每次查找从低址部分开始，增加了查找的开销

分区分配算法

(2) 循环首次适应算法

在分配内存空间时，从上次找到的空闲分区的下一个空闲分区开始查找，直到找到一个能满足要求的空闲分区，从中划出一块与请求大小相等的内存空间分配给作业。

为实现算法，需要：

- 设置一起始查寻指针
- 采用循环查找方式

优点：使内存空闲分区分布均匀，减少查找的开销

缺点：缺乏大的空闲分区

分区分配算法

(3) 最佳适应算法

所谓“最佳”是指每次为作业分配内存时，总是把能满足要求、又是最小的空闲分区分配给作业，避免“大材小用”。

要求将所有的空闲分区按其容量以从小到大的顺序形成一空闲分区链。

缺点：产生许多难以利用的小空闲区

2 实验准备

➢分区分配算法

为把一个新作业装入内存，需按照一定的分配算法，从空闲分区表或空闲分区链中选出一分区分配给该作业。

- 常用的分配算法：
  - ✓首次适应算法FF
  - ✓循环首次适应算法
  - ✓最佳适应算法

2 实验准备

- 分区分配及回收操作
  - 分配内存

    利用某种分配算法，从空闲分区链(表)中找到所需大小的分区。设请求的分区大小为u.size，表中每个空闲分区的大小表示为m.size，若m.size - u.size ≤ size(规定的不再切割的分区大小)，将整个分区分配给请求者，否则从分区中按请求的大小划出一块内存空间分配出去，余下部分留在空闲链中，将分配区首址返回给调用者。

2 实验准备

回收内存

当进程运行完毕释放内存时，系统根据回收区首址，在空闲分区链(表)中找到相应插入点，此时可能有四种情况：

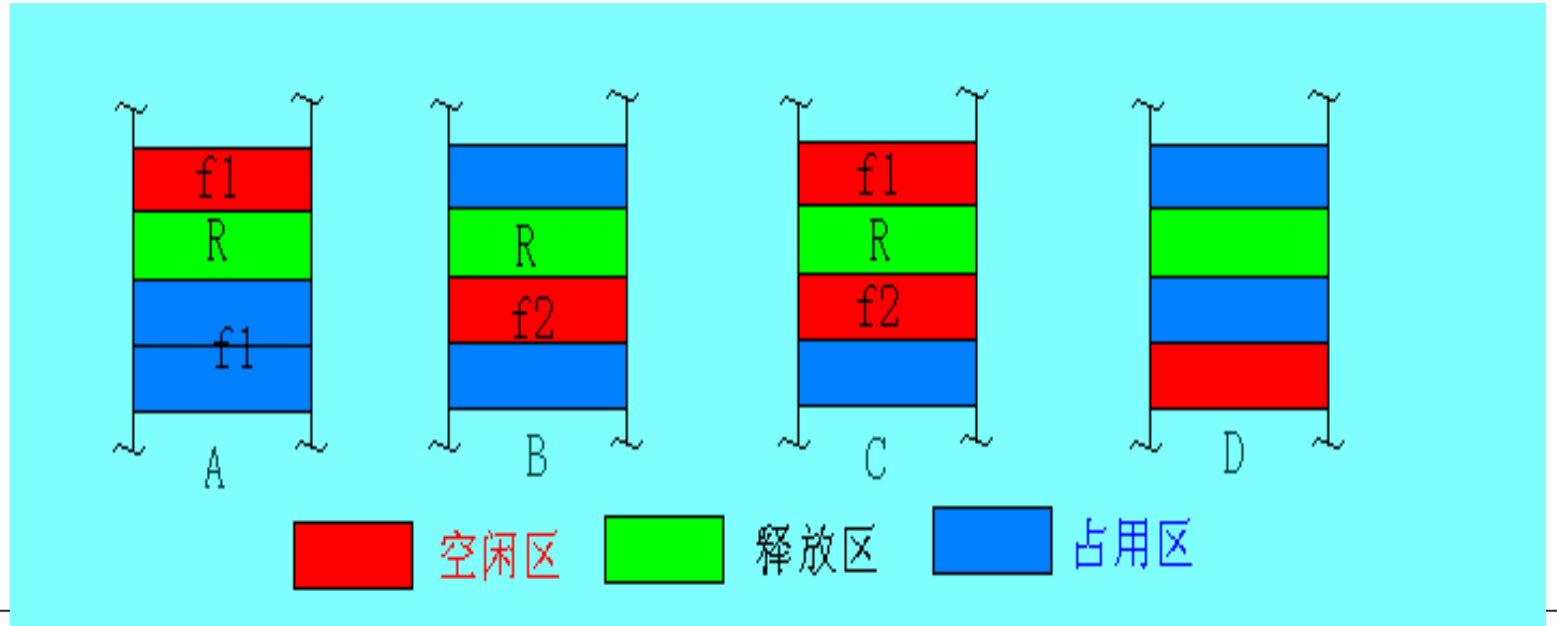
a. 回收区与插入点的前一个分区F1邻接：将回收区与F1合并，修改F1的表项的分区大小

b. 回收区与插入点的后一个分区F2邻接：将回收区与F2合并，修改F2的表项的首址、分区大小

2 实验准备

c. 回收区与插入点的前后两个分区F1、F2邻接：将三个分区合并，使用F1的表项和F1的首址，取消F2的表项，大小为三者之和

d. 回收区既不与F1邻接，又不与F2邻接：为回收区单独建立新表项，填写回收区的首址与大小，根据其首址插到空闲链中的适当位置



注：F1和F2属于空闲区

2 实验准备

2.4 存储形式

- 结构体，数组（或链表）
- PCB中要包含资源信息：分区号、分区大小等信息
- 内存信息存储

2 实验准备

➢存储形式

```
typedef struct {
    char Name[NAME_MAXSIZE];//进程名
    int MemorySize;//内存的大小
    int StartAddress;//内存起始地址
    DistributState DistbutSt;//分配状态
}PCB;

typedef enum{
    FirstPriority, BestAdapt, CycleFirst
}AllocatStrategy;
```

2 实验准备

```
typedef struct {
                                    //分区号用数组下标代替
   int PartitionSize;
   int PartStartAddr;
   char Name[NAME_MAXSIZE];//若为空，则分区空闲
}PartitionInfo;
typedef struct{
   PartiType *elem;
   int listsize;                    //表容量
   int length;                      //元素个数
}SqList, PartTable;                 //分区表
```

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

3 实验内容

3.1 实验思路

先选择内存分配策略，根据进程申请的空间大小，查找分区说明表根据内存分配策略申请空闲区，将其分割并分配。当进程结束时，修改对应的分区表，回收内存空间（针对四种情况进行修改分区说明表），从而完成内存的申请和释放。

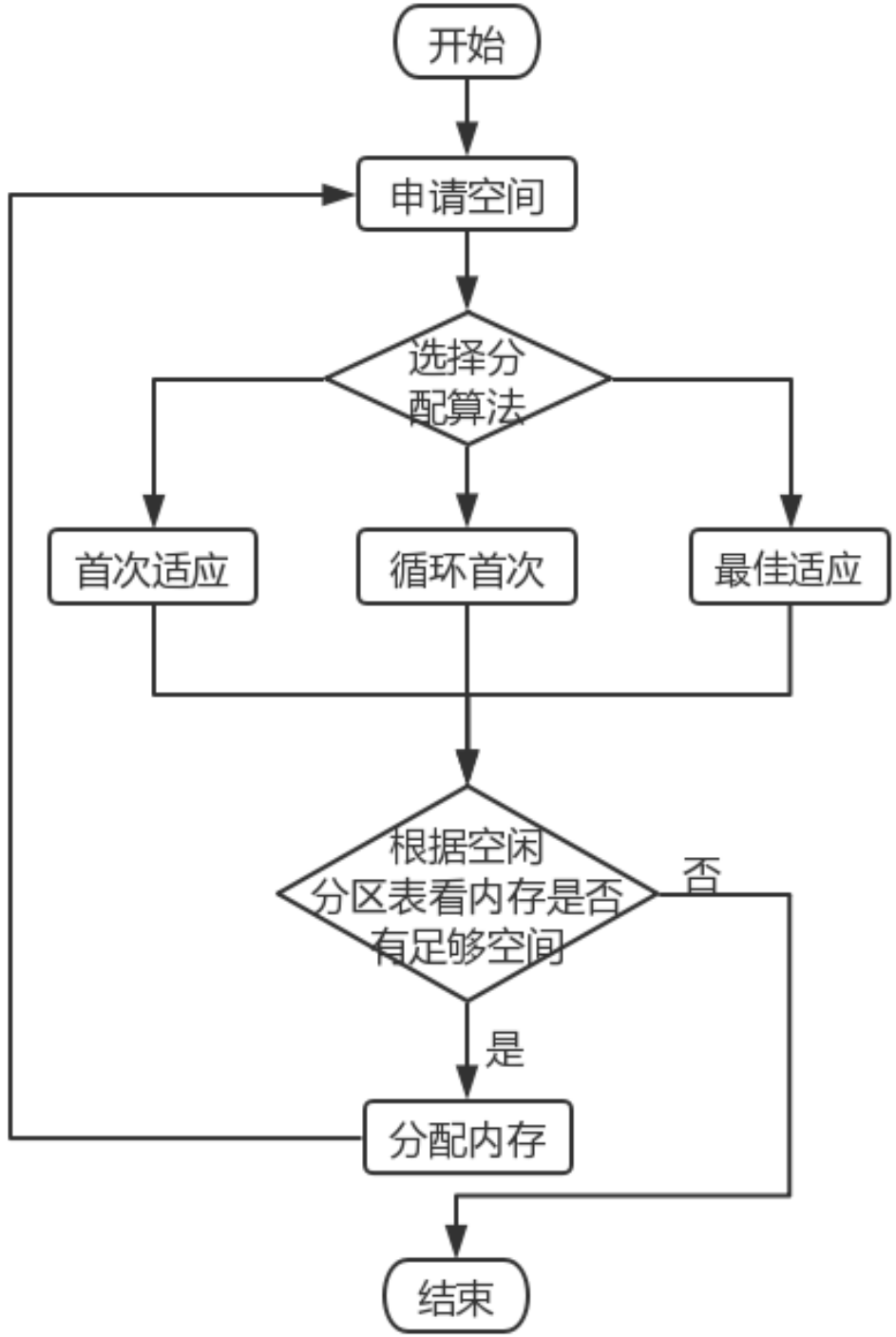
3 实验内容

3.2 动态分区实现步骤

- 1. 初始状态内存含有1块空闲分区块，空闲分区说明表记录内存信息，已占分区表为空；
- 2. 选择内存分配策略，创建多个进程（包含申请的内存大小信息），这多个进程依次申请空间，操作系统采用选取的内存分配策略将内存分给每个进程，并修改空闲分区说明表和已占分区说明表；

3 实验内容

- 3. 通过控制台来退出一些进程，再将空间依次进行回收，修改空闲分区说明表和已占分区说明表；
- 4. 要求当每次修改分区说明表时，都要打印一次分区说明表的信息，以及进程信息。


开始
申请空间
选择分
配算法
首次适应
循环首次
最佳适应
根据空闲
分区表看内存是否
有足够空间
否
是
分配内存
结束

3 实验内容

3.3 结果评价

对不同的分配策略进行进行比较，分析其优缺点。

本节知识点

1 实验目的
2 实验准备
3 实验内容
4 问题思考

4 问题思考

1、连续分配方式会产生“碎片”，为了更好的利用内存空间可以采用离散分配方式。（作业）

2、内存信息为直接给定，不能实现交互。（作业）

谢谢观赏！