《操作系统》
实验指导书

南昌大学软件学院

实验 4： 分区存储管理模拟实验

4.1 实验概述

实验目的：

1. 理解固定式分区及可变式分区两种存储管理模式，知道各自的优缺点；
2. 理解可变式分区方式的三种算法（首先适应算法、最佳适应算法和最差适应算法）的工作原理，理解内存释放的具体实现过程；
3. 在提供的代码框架下根据提示自主编程实现上述三种算法及内存释放过程。

实验语言：c

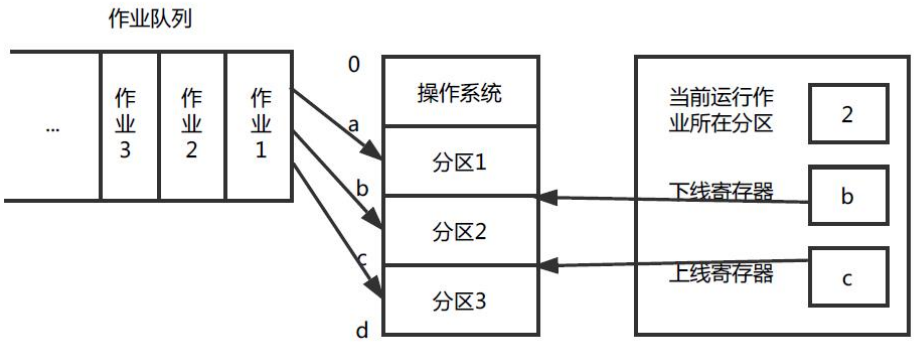
实验环境：linux、gcc

4.2 实验背景

4.2.1 固定分区存储管理

固定分区管理方式是把主存中可分配的用户区域预先划分成若干个连续的分区，每个连续区的大小可以相同，也可以不同。但是，一旦划分好分区之后，主存中分区的个数就固定了，且每个分区的大小也固定不变。这种分区法属于一种静态分区法。

在固定分区方式管理下，每个分区用来装入一个作业或进程。由于主存中有多个分区，所以这种存储管理方式适用于多道程序系统。



固定分区存储管理示意图

（注：图中的上线和下线，应该为上限和下限）

现在以批处理系统中的作业进出内存为例，介绍固定式分区存储管理的具体实现。上图是划分成三个分区的固定分区存储管理方式示意图。等待进入主存的作业排成一个作业队列。当主存中有空闲的分区时，以此从作业队列中选择一个

能装入该分区的作业。当所有的分区都已装有作业时，其他的作业暂时不能再装入，绝对不允许在同一分区中同时装入两个或两个以上的作业。已经装入主存的作业在获得处理机运行时，要限定它只能在所占的分区中执行。

一、主存空间的分配与释放

为了管理主存空间，必须设置一张“主存分配表”，以说明各分区的分配情况。主存分配表中应指出各分区的起始地址和长度，并为每个分区设置一个标志位。当标志位为 0 时，表示对应的分区是空闲分区；当标志位非 0 时，表示对应的分区已被某作业占用。空闲分区可以用来装作业。下表表示主存被静态划分成三个分区，其中分区 2 已装入一个名为 Job1 的作业。

三个分区的主存分配表

分区号	起始地址	长度	占用标志
1	a	L1	0
2	b	L2	Job1
3	c	L3	0

当作业队列中有作业需要装入主存时，存储管理可采用“顺序分配算法”进行主存空间的分配。顺序查看主存分配表，若找到一个标志位为 0 并且长度大于或等于待装入作业的地址空间长度的分区，则把此分区分配给该作业，相应表目的标志改成作业名的标识；若找不到一个这样的空闲分区，则改作业暂时不能装入主存。

主存空间的释放很简单。某作业执行结束后必须归还所占的分区，这时存储管理根据作业名查看主存分配表，找到相应的表目后，把其中的标志位重新置成 0 即可。

二、主存空间的利用率

用固定式分区方式管理主存时，由于很多作业的尺寸所占用的分区尺寸小，因此造成这些分区内部有一部分空间闲置不用，严重影响了主存空间的利用率。这种分区内的空闲部分成为“内部碎片”。

减小“内部碎片”，从而提高主存空间利用率的办法有以下三种，其中第一种做法最为常用。

①系统在对主存从地段到高端进行静态分区时，以此划分出从小到大尺寸渐增的几个分区，并记录在主存分配表中。这样，按顺序分配算法给作业的分区总是一个能满足作业要求的最小的空闲分区，因此，分区内的碎片是最小的，这时的顺序分配算法其实就是“最佳适应分配算法”。

②根据经常出现的作业的大小和频率进行静态分区。

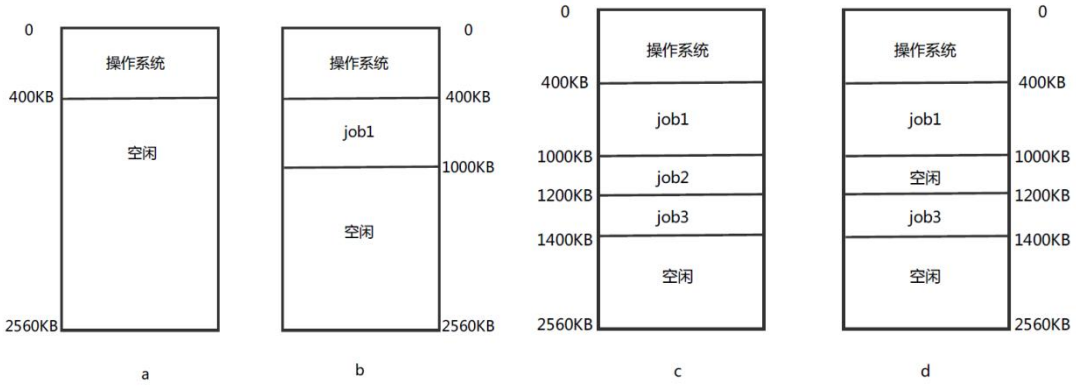
③作业按对主存空间的需求量排成多个队列，规定每个队列中的作业只能装入到对应的指定分区中。这样可防止小作业进入到大分区中，但也会造成某个作

业队列经常为空，因而相应的分区经常闲置的情况发生。

4.2.2 可变分区存储管理

固定分区存储管理方式简单易实现，但最大缺点是“内部碎片”问题严重，主存利用率低。为了克服固定分区管理方式的缺点，一种可以消除“内部碎片”的动态分区技术应运而生，这就是可变分区存储管理方式。这种管理方式不是把作业装入已经分好的分区中，而是在作业要求装入主存时，根据作业需要的主存量和当时主存空间的使用情况决定是否可以装入该作业。当主存有足够的空间能满足作业要求时，就按作业需求量划出一个分区给该作业。由于分区的大小是按照作业的实际需求量来定的，故在作业占用的分区里没有“内部碎片”。这种动态分法使得分区的长度和大小都是可变的。

总体上讲，可变分区管理方式下主存空间的分配与释放跟固定分区管理采用的算法差不多，但由于可变分区管理方式下主存空间在使用过程中会出现划分得比较凌乱的情况，因而带来一些新的问题，因此相应地的主存空间的分配与释放算法也稍微复杂一些。



采用可变分区管理方式的系统中，系统初始化时，主存中除操作系统占用部分外，把整个用户区看成是一个大空闲区，如图 a 所示。当有作业要装入主存时，从空闲去划出一个与作业长度一致的分区来装作业，剩余部分仍为空闲区，如图 b 和 c 所示。当作业需求量超过空闲区长度时，该作业暂时不能装入。当某作业执行结束后，它所占分区必须被收回，成为一个空闲区，如图 d 所示。随着作业不断的进出主存，主存空间被分成了许多区，有的分区被占用，有的分区空闲。特别是，可能在许多被作业占用的分区之间出现了一些无法装入任何作业的小的空闲区，这些小的空闲分区也是主存空间的一种浪费，成为“外部碎片”。为了尽可能简单地及时消除这些外部碎片，主存空间的释放算法中增加了合并相邻空间分区的操作。

由于采用可变分区方式管理主存时，主存中已占分区和空闲分区的数目和大小都是变化的，所以为了便于对主存空间的分配与释放，主存分配表可以用两张表格组成，一张是“已分配区表”，另一张是“空闲区表”。已分配区表记录已

装入主存的作业所占分区的起始地址和长度，并用标志位指出占用分区的作业名。空闲区表记录主存中可供分配的空闲区的起始地址和长度，也用标志位指出该分区是“未分配”的空闲区。由于已占分区和空闲分区的个数不定，所以在已分配区表和空闲区表中都有一定的空表项（其标志位的值为“空”），分别表示相应的分区“已释放”和“已分配或已合并”。其实，空表项是这两种分区管理表格中的分配单位。

采用可变分区方式管理主存时，主存空间的分配与释放都要对已分配区表和空闲区表这两个表进行访问和修改。比如，分配主存时，先查空闲区表，等完成分配后，要修改空闲区表和已分配区表。具体调整过程如下：分配后，把已分配区表中的一个空表项改成一个标志为某作业的相应表项，其起始地址和长度均来自空闲表区；同时，空闲表区仅当被选中分区大小与作业需求相等时才将相应表项状态置成“空”，否则只把相应表项的起始地址和长度改为分割后的值。释放后，将已分配区表中找到插入点，仅当被释放分区与其他空闲分区不相连时，才把空闲区表中的一个空表项状态改为“未分配”的相应表项，否则，要把释放区与相邻的空闲区进行合并，最后把合并结果记录在空闲区表中。

4.2.3 可变分区存储管理的三种算法

为了提高主存分配算法访问空闲分区表的效率，常常对空闲区表的表项按一定顺序进行排列，然后仍按“顺序分配算法”检索空闲表区，进行主存空间的分配。这也导致主存空间的释放算法中必须增加有序插入一个表的操作。由于空闲区表表项的排列顺序有三种，因此就有了由顺序分配算法演化来的以下三种不同的分配算法。这三种分配算法的执行各有利弊，但执行流程是一样的，只不过所用的空闲区表表项的排列顺序不同罢了。

一、首先适应分配算法

这是一种性能一般，但实现比较自然直接，而且易于释放时合并相邻空闲分区的分配算法，因而也是可变分区管理中最常用的分配算法，它所用的空闲分区表的表项是按相应分区的地址大小以递增顺序排列的。分配时顺序查找空闲区表，找到第一个能满足作业要求的空闲区，如果该空闲区比作业长度大，则分割这个空闲区，一部分分配给作业，另一部分仍为空闲区；如果该空闲区与改作业等大小，则直接把它分给作业。调整相应的空闲区表和已分配区表。

优点：释放分区时易于合并相邻的空闲分区，尽可能地保留了高地址端的空闲区。

缺点：完成一次分配平均需要的搜索次数较大，影响了工作效率。

二、最佳适应分配算法

该算法的分配过程同首先适应分配算法，但所用空闲区表的表项按相应分区的容量以递增顺序排列。使用最佳适应分配算法找到的第一个能满足作业要求的

空闲区，一定是一个最小的空闲区，即其大小最接近或最佳适应作业要求的空闲区。这样可保证不去分割更大的区域，更利于今后到来的大作业。

优点：平均只要查找一半便能找到最佳适应的空闲区；如果有一个空闲区的容量正好满足作业要求，则它必被选中；尽可能地保留了较大的空闲区；

缺点：产生非常小的空闲区（“外部碎片”）。

三、最差适应分配算法

该算法的分配过程也与首先适应分配算法相同，但所用空闲区表的表项是按相应分区的容量以递减顺序排列的。使用最差适应分配算法找到的第一个能满足作业要求的空闲区，一定是一个最大的空闲区，即其大小最远离或最差适应作业要求的空闲区。这样可保证每次分割后的剩余部分不至于太小，仍可被分配使用，以减少“外部碎片”。

优点：分割后产生的空闲区一般仍可供以后分配使用。

缺点：工作一段时间后，不能满足大作业对空闲区的请求。

4.3 实验内容

本实验需要你阅读并理解所给源代码，然后补全代码，编译运行它们，体会算法的实现过程。源程序是 partmmuhard.c。

需要你实现的函数分别是 First_fit()、Best_fit()、Worst_fit()和 free_part()四个函数。请你理解算法，先画出各函数的流程图，再分别对这四个函数进行实现，然后编译运行程序验证代码的正确性。