文章编号:1005-6033(2005)15-0252-02 收稿日期:2005-07-04

浅析伙伴系统的分配与回收

焦莉娟

(忻州师范学院计算机系,山西忻州,034000)

摘 要:分配效率、碎片问题是操作系统中内存分配的两大问题。一个好的分配器应该能够快速地满足各种大小的分配要求,同时不能产生大量的碎片浪费空间。基于数据结构中的伙伴系统的分配与回收思想给出了一个有效的算法。

关键词:伙伴系统;存储管理;空闲块;操作系统

中图分类号:TP316 文献标识码:A

内存管理系统是操作系统中最为重要的部分,从管理与使用角度考虑,可以分为静态存储管理和动态存储管理。其中动态存储管理的基本问题是系统如何应用户请求分配内存,又如何回收那些用户不再使用而释放了的内存,以备新的请求产生是重新进行分配。我们这里讨论的就是动态存储管理系统。在不同的动态存储系统中,每次分配给用户的内存区域大小不一。所以,对于不同的内存管理算法,造成的内存利用率也不尽相同。其中伙伴系统就是操作系统中常用到的一种动态存储管理方法。为了方便起见,在下面的讨论中,将系统已分配给用户使用的地址连续的内存区称为"占用块",称未曾分配的地址连续的内存区为"空闲块"。

1 动态存储管理的方法

动态存储管理系统进行内存管理通常采用以下 3 种方法:

(1) 系统运行期间所有用户请求分配的内存量大小相同。其实现方法是,在系统开始运行时将内存区按所需大小分割成若干大小相同的块,然后用指针链接成一个可利用的空间表。采用这种方法分配和回收都比较简单,但它牺牲的是内存利用率。

(2) 系统运行期间用户请求分配的存储量有若干种大小的规格。这种管理策略下的系统需要建立多个可利用空间表,同一链表中节点大小相同。这种方法的算法复杂性和系统利用率介于二者之间。

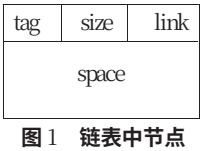

图 1 链表中节点的结构

(3) 系统在运行期间分配给用户的内存块的大小不固定,可以随请求而变。因此,可利用空间表中的节点,即空闲块的大小也是随意的。这种情况下,链表中节点的结构与前两种情况有所不同,节点中除标志域 tag 外,尚需一个标明节点大小的域 size,以指示空闲块的存储量。其结构如图 1 所示。

2 伙伴系统

2.1 伙伴系统概述

伙伴系统是操作系统中用到的一种动态存储管理方法。它属于上述介绍的第三种情况,即在用户提出申请时,分配一块大小恰当的内存区给用户,在用户释放内存区时即回收。在伙伴系统中,无论是占用块还是空闲块,其大小均为 2 的 k 次幂(k 为某个正整数)。当用户申请 n 个字的内存区时,分配的占用块大小为 2^k 个字($2^{k-1}<n\leq 2^k$)。由下面将要讨论到的分配回收算法可以看到,在分配时经常需要将一个空间块分裂成两个大小相等的存储区。这两个由同一个大块分裂出来的小块就称为"互为伙伴"。

系统将所有空闲块链接在一个双重循环链表结构的可利用空间表中,如图 2 所示。假设系统的可利用内存空间为 2^m 个字,则在开始运行时内存区是一个大小为 2^m 的空闲块。运行一段时间后,被分成若干占用块和空闲块。为了便于查找,将大小相同的块建立在一张子表中。每个子表又是一个双重链表,这样的子链表最多可有 m+1 个。将着 m+1 个表头指针用向量结构组织成一个表,就是伙伴系统中的可利用空间表,如图 3 所示。

2.2 分配与回收算法

在研究系统的分配与回收方法之前,我们有必要先定义其可利用空间表的数据类型如下。

```
typedef struct word{
word * llink;//指向前驱节点
```

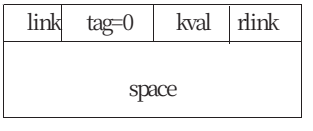

图 2 空闲块的节点结构

7 网络系统的经济效益

减少手工操作,提高工作效率,确保数据安全可靠,数据交换加速,数据存储容量大,方便实用,切实使住房补贴工作更加规范化、科学化。

(实习编辑:王永胜)

第一作者简介:吴丽芳,女,1973 年 8 月生,1994 年毕业于广西桂林航天高等专科学院计算机应用专业,助理工程师,山西省晋城煤业集团长平公司物业管理中心房屋财产办,山西省晋城市,048006.

Compilation and Implementation of the Information Networking Management System for Employees' House Allowrance

WU Li-fang

ABSTRACT: This paper introduces in detail the compilation and implementation of the information networking management system for employees' house allowranc, which include the construction of the database, the principles of the data integration, the supporting tools of the data integration and development, the remote network technique, the evaluation of the data quality, the technical difficult points of the system, and the economic benefit of the network system, etc.

KEY WORDS: database of house aloowrance; networking management; network structure

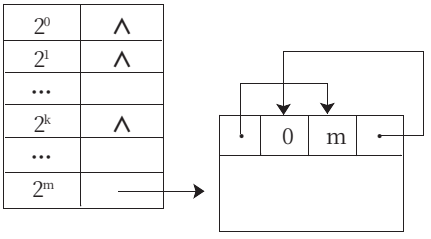

图 3　表的初始状态

```
int      tag;  //块标志,0:空闲,1:占用
int      kval;  //块大小,值为 2 的 k次幂
word   * rlink;  //头部域,指向后继节点
OtherType  Other;//字的其他部分
} word,head;      //word:内存字类型,节点的第一个字也称为 head
typedef  struct  HeadNode {
int   nodesize;  //链接表空闲块的大小
word  * frist;   //链接表的表头指针
} FirstList[m+1];   //表头向量类型
```

(1) 分配算法。当用户提出大小为 n 的内存请求时,首先在可利用表上寻找节点大小与 n 相匹配的子表,若此子表非空,则将子表中任一个节点分配之即可;若此子表为空,则需从节点更大的非空子表中去查找,直到找到一个空闲块,将其中一部分分配给用户,将剩余部分插入相应的子表中。算法描述如下:

```
Space AllocBuddy(FreeList &avail,int n){
for(k=0;k<=m&&(avail[k].nodesize<n+1||!avail[k].first)++k);//查找满足分配要求的子表
If(k>m)return NULL;//分配失败返回空
else{//进行分配
pa=avail[k].first;//指向可分配子表的第一个节点
  pre=pa(llink;suc(rlink;//分别指向前驱和后继
if(pa==suc)avail[k].first=NULL;//分配后该子表变成空
else{//从子表删去* pa 节点
pre(rlink=suc;suc(llink=pre;avail[k].first=suc;
}
for(I=1;avail[k-I].nodesize>=n+1;++I){
pi=pa+2k^i;pi(rlink=pi;pi(llink=pi;
pi(tag=0;pi(kvail=k-I;avail[k-I].first=pi;
}//将剩余块插入相应子表
pa(tag=1;pa(kval=k-(--I);
}
return pa;
}//AllocBuddy
```

(2) 回收算法。当用户释放不再使用的占用时,系统将这新的空闲块插入到可利用空间表中去。在伙伴系统中系统仅考虑互为伙伴的两个空闲块的归并。所以首先应判别其伙伴是否为空闲块,若否,则只要将释放的空间块简单插入在响应的子表中即可;若是,则需要在响应子表中找到其伙伴并删除之,然后再判别合并后的伙伴是否为空闲块。依此重复,直到归并所得空闲块的伙伴不是空闲块时再插入到相应的子表中。其递归算法如下:

```
void Free_BS (freelist &avail,char * addr,int n)//伙伴系统的空闲块回收算法
{
buddy=addr%(2* n)?(addr- n):(addr+n);//求回收块的伙伴地址
addr- >tag=0;
addr- >kval=n;
for(i=0;avail[i].nodesize<n;i++);//找到这一大小的空闲块链
if(!avail[i].first)//尚没有该大小的空闲块
{
addr- >llink=addr;
addr- >rlink=addr;
avail[i].first=addr;//作为唯一一个该大小的空闲块
}
else
{
for(p=avail[i].first;p!=buddy&&p!=avail[i].first;p=p- >rlink);//寻找伙伴
if(p==buddy)//伙伴为空闲块,此时进行合并
{
if(p- >rlink==p)avail[i].first=NULL;//伙伴是此大小的唯一空闲块
else
p- >llink- >rlink=p- >rlink;
p- >rlink- >llink=p- >llink;
}//从空闲块链中删去伙伴
new=addr>p?p:addr;//合并后的新块首址
Free_BS(avail,new,2* n);//递归地回收新块
}//if
else//伙伴为占用块,此时插入空闲块链头部
{
q=p- >rlink;
p- >rlink=addr;addr- >llink=p;
q- >llink=addr;addr- >rlink=q;
}
}//else
}//Free_BS
```

3　结语

无论是分配还是回收算法,伙伴系统都相应比较简单,而且执行速度较快,但由于它只归并互为伙伴的空闲块,所以容易产生碎片。

参考文献

[1]　严蔚敏,吴伟民.数据结构[M].北京:清华大学出版社,1999.
[2]　汤子瀛,哲凤屏,汤小丹.计算机操作系统[M].西安:西安电子科技大学出版社,1998:257;390.
[3]　李学干,苏东庄.计算机系统结构[M].第 2 版.西安:西安电子科技大学出版社,1998:136.

(责任编辑:邱娅男)

第一作者简介:焦莉娟,女,1978 年 2 月生,2000 年毕业于山西大学计算机科学系,现为山西大学计算机科学系 2004 级在读研究生,忻州师范学院计算机系,山西省忻州市,034000.

Allocation and Reclaim Algorithm of Buddy System

JIAO Li-juan

ABSTRACT: The allocation efficiency and piece problemare two of the important problems in operating system. An effective divider should meet all kinds of demand quickly and do not generate too many pieces. Buddy systemis a very good one. This paper, based on the thought of allocation and reclaimof the buddy systemin data structure, provides an effective algorithm.
KEY WORDS: buddy system; storage management; free block; operating system