第六章 运行时的存储空间

1. 写一个表区管理算法。总的有一个表区，其中将存放各种结构的表。我们假设表分为以 BYTE 为单位的和以 BYTE)为单位的两种。凡是创建新表(空表)，填表，删表和改表等操作都必须通过这个系统。其中内部应有回收能，以便在填表时如果没有够大的碎片，应能回收废区。表元素的存放将按数组方式而不是链表方式。假设系统级语言写。

(答案)

单位标志	---标识以 BYTE 为单位还是以 4BYTE 为单位
引用计数	---记录有多少指向该块的指针
SIZE	---内容区域有多少单位
内 容	---内容
连 接	---连接下一部分的指针

创建表 Create Table (flag: integer)

1. 从总的表区中寻找最大的空闲区域的第一个起始地址记 addr. (空闲区域要大于 3BYTE)
2. IF (flag =1) THEN
   addr 单元写入 1;
   ELSE
   addr 单元写入 4;
3. adrr+1 处，引用计数：=1;
4. adrr+2 处，SIZE：=0;
5. adrr+3 处，写入连接标记：=0；表示不需要续接表。

填表 WriteTable(addr: ↑Table; content: ContentType)

1. 从 addr 处读入单位标志，并送入 flag;
2. 从 addr+2 处读入大小到 SIZE;
3. IF (addr+3+size*flag 到 addr+3+(size+1)*flag 之间全是空闲单元) THEN
   content 写入这段区域; size:=size+1;
4. IF(addr+4+size*flag≠0) THEN
   { addr:= addr+4+size*flag 中的那个地址;
   size:= addr+2 处的大小;
   goto 3.
   }
   ELSE
   {从 addr+4+size*flag 向后找空闲区域大于（3+flag）的块
   addr：=这个块地址头;
   addr 单元写入 flag;
   addr+1 处写入 1;
   addr+2 处写入 1;
   goto 3.
   }

```
改表 ModifyTable (addr:↑Table, content:ContentType)
1. 从 addr 处读入 flag;
2. 从 addr+2 处读入大小到 SIZE;
3. IF(addr+3 到 addr+3+size*flag 之间有要修改的东西) THEN
   写入 content;
    ELSE
   addr:= addr+4+size*flag
   goto 2.
```

(关闭)

2. 试写一个堆区管理中的边界融合算法。

(答案)

```
堆块节点:
Heap=RECORD
        length    :INTEGER;
        Next      :↑Heap
   END
融合算法:
   tp1,tp2: ↑Heap;
   tp1=Head;
   while(tp1!=NULL)
   {
     tp2=tp1↑.next;
     if((tp2!=NULL)&&(tp2-tp1=tp1↑.length))
              { tp1↑.length=tp1↑.length+tp1↑.length;
                tp2=tp2↑.next;
              }

     else     { tp1=tp2; }

   }
```

(关闭)

3. 一个活动记录包含哪些信息？各信息的作用？何时填写它们？

(答案)

结 构	功 能	填写
→top		
临时变量区	---涉及编译器产生的变量，而不是用户程序中的变量。	编译
局部变量区	---涉及用户程序中过程的局部变量	编译
形参变量区	---涉及用户程序的过程中的形参变量	编译
返回值	---涉及用户程序的过程中的返回值	运行
全局变量环境	---包括有关访问非局部变量的信息	运行
机器状态	---包括寄存器状态等过程中断时机器状态	运行
过程层数	---相应过程的层数，用于非正常出口情形	运行
返回地址	---返回值的地址	运行
动态链指针	---用于释放当前 AR，是因为 AR 长度不尽相同而需要的。	运行
sp→		

(关闭)

4. 活动记录 AR 中为什么需要动态链。

(答案)

调用过程就要开辟一个新的 AR(一个新的执行环境)，而且新 AR 将成为当前 AR，故栈指针被修改。但当调用需要释放当前活动记录 NewAR,回到上一个过程的执行环境，所以每个 AR 中都需要保存前一个 AR 的首地址就是用来实现这个功能的。

(关闭)

5. 证明每个过程的活跃 AR 唯一。

(答案)

证明：栈区是由 AR 组成的一个链。过程的每个 AR（即一个过程有 0...n 个 AR 相对应），运行中多个过程的 A 个链。相邻 AR 之间有动态链指针相连。
假设一个过程有多于一个活跃 AR 相对应，则当运算转至该过程时，需从该过程的活跃 AR 中返回机器状态，1 个 AR。因此，从哪个 AR 中返回机器状态就产生了矛盾。所以，一个过程只对应唯一活跃 AR。

(关闭)

6．Display 表、静态链和 Display 寄存器的作用是什么？试举或找一个程序例子，并考查其 Display 表、静态链寄存器表的内容。

(答案)

Display 表方法是用表结构来表示变量访问环境；静态链是用链表表示变量访问环境；寄存器方法是用寄存器表问环境.

例: Procedure P;

Procedure Q; begin R end;

Procedure R;

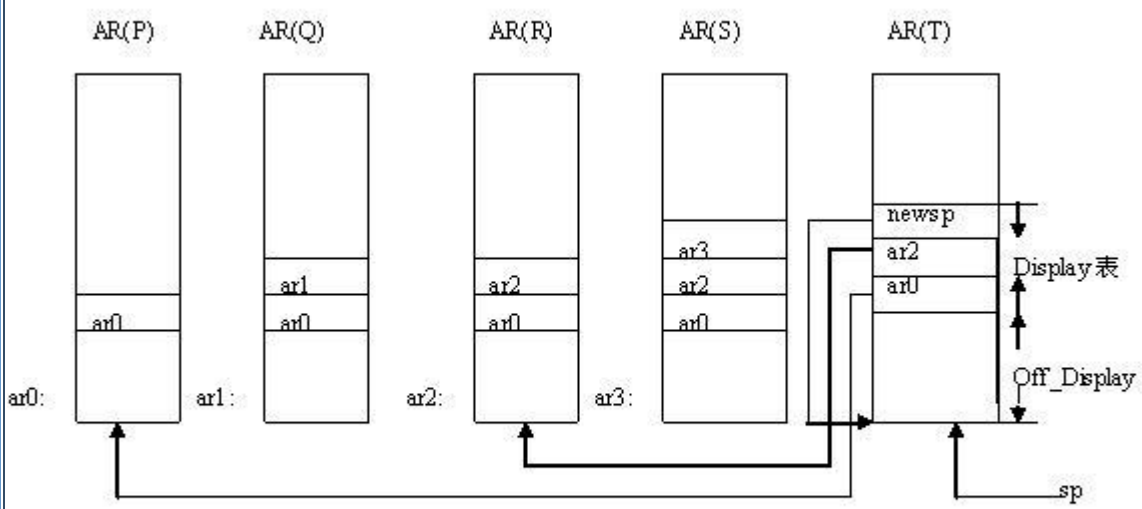
Procedure S ; begin T end;

Procedure T ; begin end;

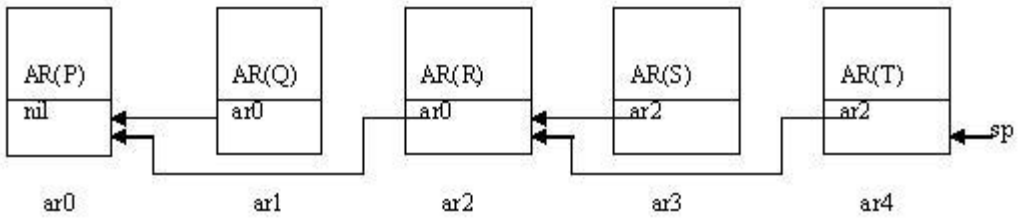
begin S end

begin Q end

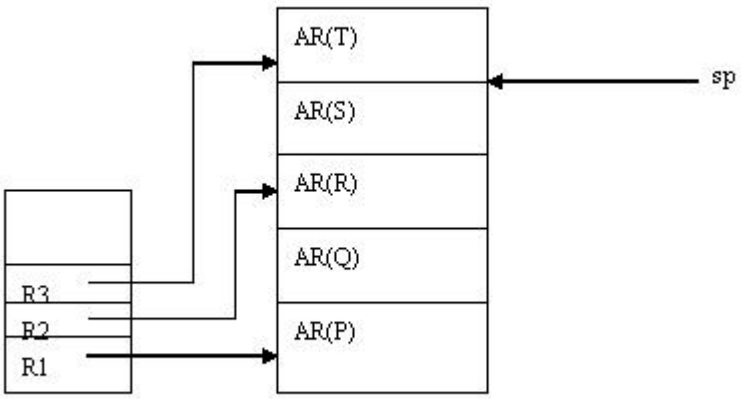
(1)Display 表:



(2)静态链方法；



(3)寄存器方法：



(关闭)

7. 栈区中的每个 AR 对应一个过程，试证明不可能有不同的 AR1 和 AR2 使得其对应过程名不相同而层数相同

(答案)

证明：栈区中每个 AR 对应一个过程，故没有同一过程对应一个以上 AR。AR1，AR2 对应过程名不同，所以栈区中。因为以 AR1，AR2 对应的过程必有 P1 中有 P2 的调用或 P2 中有 P1 的调用。若 P1，P2 调用是平行的在谁包含谁的问题。则 AR1，AR2 不可能同时在栈中。这就证明了 AR1，AR2 对应的过程必有 P1 中有 P2 的中有 P1 的调用。因此，P1，P2 的层数肯定不同

(关闭)

<<上一章◎ ◎回页首◎ ◎下一章>>