2、内存管理

1．

将逻辑地址化分成“页号，偏移量”的形式；

方法一：将210010转化成二进制，从低位截取10位为页内偏移量（页大小为1024字节），剩余的高位为页号；

方法二：21001024=2…52，即页号为2，页内偏移量为52；

查页表，2号页面在第6号块中，因此逻辑地址2100对应的物理地址为6x1024+52=6196;

3100 1024＝3…28，即页号为3，页内偏移量为28；

逻辑地址3100对应的物理地址为8x1024+28=8220;

画出地址变换图；

2．答： 85%\*1+15%\*2\*1=1．15

3、（1）答： S：表示售票厅可容纳的购票者的人数（ S的最大值为20）

S=0 ：表示售票厅已满

0<S <=20：表示售票厅还可容纳S人

S<0：：表示售票厅已满，且外面等待|S|人。

（2）　PROBEGIN　PROCESS　PI(I=1，2，……)

　　　　　 begin

　P（S）　　；

　　　　　　　　　进入售票厅；

　　　　　　　　　购票；

　　　　　　　　　退出；

　 V（S）　　；

　　　　　 end；

　END (2分)

（3）（-∞ ，20）

3、答：1）

2） 最短作业优先算法：ADBC

高响应比优先调度算法：ACBD

3）最短作业优先算法的平均周转时间，平均带权周转时间 ：7，8．9

高响应比优先调度算法的平均周转时间，平均带权周转时间：6．7，9．1

4、答： 1）将所采用的页面置换算法的缺页中断率与最佳页面置换算法的缺页中断率进行比较，两者之差越小则所采用的页面置换算法性能就越好。

2）LRU置换算法基本思想： 淘汰的页面是选择距现在最久未使用的页面。

3）页面淘汰的顺序： 1，2，6，4，7，3，2，1，4，7

缺页中断次数： 14次

最后留驻存4页的顺序： 1，2，5，6

5、物理地址＝219+430=649

物理地址＝2300+10=2310

因为段内偏移量500>段长100，地址越界；

物理地址＝1327+400=1727

因为段内偏移量112>段长96，地址越界；

6、采用首次适应算法

块1：首址290KB，长度10KB

块2：首址400KB，长度112KB

采用最佳适应算法

块1：首址240KB，长度60KB

块2：首址450KB，长度62KB

随后又要申请80KB

对于首次适应算法，分配成功；

对于最佳适应算法，分配失败；

7、　解：

　　根据已知条件页面大小是100字，将页面访问序列简化为：

　　0，0，1，1，0，3，1，2，2，4，4，3

　　又因为该程序基本可用内存是200字，可知内存块数为2。

　　采用先进先出置换算法（FIFO），总共有6次缺页，缺页率为6/12=50%，具体算法如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 页面走向 | 0 | 0 | 1 | 1 | 0 | 3 | 1 | 2 | 2 | 4 | 4 | 3 |
| 块1 | 0 |  | 0 |  |  | 3 |  | 3 |  | 4 |  | 4 |
| 块2 |  |  | 1 |  |  | 1 |  | 2 |  | 2 |  | 3 |
| 缺页 | 缺 |  | 缺 |  |  | 缺 |  | 缺 |  | 缺 |  | 缺 |

　　采用最近最少使用置换算法（LRU），总共有6次缺页，缺页率为6/12=50%，具体算法如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 页面走向 | 0 | 0 | 1 | 1 | 0 | 3 | 1 | 2 | 2 | 4 | 4 | 3 |
| 块1 | 0 |  | 0 |  |  | 0 | 1 | 1 |  | 4 |  | 4 |
| 块2 |  |  | 1 |  |  | 3 | 3 | 2 |  | 2 |  | 3 |
| 缺页 | 缺 |  | 缺 |  |  | 缺 | 缺 | 缺 |  | 缺 |  | 缺 |

　　采用最佳置换算法（OPT），总共有5次缺页，缺页率为5/12=41.6%，具体算法如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 页面走向 | 0 | 0 | 1 | 1 | 0 | 3 | 1 | 2 | 2 | 4 | 4 | 3 |
| 块1 | 0 |  | 0 |  |  | 3 |  | 3 |  | 3 |  |  |
| 块2 |  |  | 1 |  |  | 1 |  | 2 |  | 4 |  |  |
| 缺页 | 缺 |  | 缺 |  |  | 缺 |  | 缺 |  | 缺 |  |  |

8、　页式存储管理的逻辑地址分为两部分：页号和页内地址。由已知条件“用户编程空间共32个页面”，可知页号部分占5位；由“每页为1KB”，1K=210，可知内页地址占10位。由“内存为16KB”，可知有16块，块号为4位。

　　逻辑地址0A5C（H）所对应的二进制表示形式是：000 1010 0101 1100 ，根据上面的分析，下划线部分为页内地址，编码“000 10”为页号，表示该逻辑地址对应的页号为2。查页表，得到物理块号是4（十进制），即物理块地址为：01 00 ，拼接块内地址10 0101 1100，得01 0010 0101 1100，即125C（H）。

9、

|  |  |  |  |
| --- | --- | --- | --- |
| 内存块数 | 淘 汰 算 法 | | |
| LRU | FIFO | OPT |
| 3 | 15 | 16 | 11 |
| 5 | 8 | 10 | 7 |

10、　（1）采用先来先服务磁盘调度算法FCFS，进行调度的情况为：从143道开始

|  |  |
| --- | --- |
| 下一磁道 | 移动磁道数 |
| 86  147  91  177  94  150  102  175  130 | 57  61  56  86  83  56  48  73  45 |

磁头移动总量为565。

　　（2）采用最短寻道时间优先磁盘调度算法SSTF，进行调度的情况为：从143道开始

|  |  |
| --- | --- |
| 下一磁道 | 移动磁道数 |
| 147  150  130  102  94  91  86  175  177 | 4  3  20  28  8  3  5  89  2 |

磁头移动总量为162。

　　（3）采用电梯磁盘调度算法，进行调度的情况为：从143道开始

|  |  |
| --- | --- |
| 下一磁道 | 移动磁道数 |
| 147  150  175  177  130  102  94  91  86 | 4  3  25  2  47  28  8  3  5 |

磁头移动总量为125。

11、① 先来先服务法（FCFS）

作业1 作业2 作业3 作业4 作业5

0 10 11 13 14 19 t

时间片轮转法（RR）

作业 1 2 1 3 4 1 5 3 1 5 1 5 1 5 1 5 1 1 1

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 t

非抢占式优先级:

作业1 作业4 作业3 作业5 作业2

0 10 11 13 18 19 t

②和 ③

先来先服务法（FCFS）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 作业 | 到达时间 | 运行时间 | 完成时间 | 周转时间 | 带权周转时间 |
| 1 | 0 | 10 | 10 | 10 | 1.0 |
| 2 | 1 | 1 | 11 | 10 | 10.0 |
| 3 | 2 | 2 | 13 | 11 | 5.5 |
| 4 | 3 | 1 | 14 | 11 | 11.0 |
| 5 | 4 | 5 | 19 | 15 | 3.0 |
| 平均周转时间 | | 11.4 | | | |
| 平均带权周转时间 | | 6.1 | | | |

时间片轮转法（RR）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 作业 | 到达时间 | 运行时间 | 完成时间 | 周转时间 | 带权周转时间 |
| 1 | 0 | 10 | 19 | 19 | 1.9 |
| 2 | 1 | 1 | 2 | 1 | 1.0 |
| 3 | 2 | 2 | 8 | 6 | 3.0 |
| 4 | 3 | 1 | 5 | 2 | 2.0 |
| 5 | 4 | 5 | 16 | 12 | 2.4 |
| 平均周转时间 | | 8.0 | | | |
| 平均带权周转时间 | | 2.06 | | | |

非抢占式优先级

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 作业 | 到达时间 | 运行时间 | 完成时间 | 周转时间 | 带权周转时间 |
| 1 | 0 | 10 | 10 | 10 | 1.0 |
| 2 | 1 | 1 | 19 | 18 | 18.0 |
| 3 | 2 | 2 | 13 | 11 | 5.5 |
| 4 | 3 | 1 | 11 | 8 | 8.0 |
| 5 | 4 | 5 | 18 | 14 | 2.8 |
| 平均周转时间 | | 12.2 | | | |
| 平均带权周转时间 | | 7.06 | | | |

12、磁盘总盘字节数：16\* 210 \* 210 \*210 =234字节

磁盘总块数： 234 /210 块= 224 块

存储位示图需要的位数： 224 位

存储位示图需要的字节数：224 /23 字节=221字节

位示图所占盘块数： 221 /210 块=211块

**13**、1、

磁盘总字节数：16\*230=234字节

磁盘总块数：234/210 块= 224 块

FAT表所占空间（字节数）： 224 \*4 字节=226字节

采用FAT所占盘块数为： ： 226 /210 块=216块

2、

磁盘总盘字节数： 234字节

磁盘的总块数上限： 216块

每个磁盘块的大小最小应该为： 234/ 216=218字节，即256KB

**16**、解析：（1）（求安全系列：说的通俗点就是把所有剩下的资源分给进程，但是要看剩下的资源数是否能满足进程的需求量，满足了就分给它，等它结束后释放它所拥有的所有资源，再继续往下分，直到所有的进程都运行成功。）

1>先求剩余量**A(17)、B(5)、C(20) ————**三个资源的各自的总量

**Work=2 3 3————**分给每个进程后三个资源所剩下的量{这样子算的：

A（17）-2-4-4-2-3=2，即总量减去已分配里边竖起来相加的和等于剩余}

2>再求每个进程T0时刻的所需量：Need=最大雪球—已分配

综上可得到如图：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 最大需求 | 已分配 | **Need** |
| P1 | 5 5 9 | 2 1 2 | **3 4 7** |
| P2 | 5 3 6 | 4 0 2 | **1 3 4** |
| P3 | 4 0 11 | 4 0 5 | **0 0 6** |
| P4 | 4 2 5 | 2 0 4 | **2 2 1** |
| P5 | 4 2 4 | 3 1 4 | **1 1 0** |

观察Need的那些值，**Work=2 3 3，**work能够need的有P4跟P5，二者选哪个都可以，比如就选择P4，即我们把work的值分给P4，让它运行完成，P4释放所有资源，则work的值就变为了 work=4 3 7（P4的所有资源都要释放哦），再继续找满足条件的进程以此类推，可得到安全系列:{P4,P2,P5,P3,P1}(**第一问在答题时只需写出一个安全系列就行了，且安全系列并不确定)**

（2）在(2)的基础上P2: Request(0,3,4)（P2的need（=1 3 4））

第一次比较：Request(0,3,4)<= need（=1 3 4）

第二次比较：Request(0,3,4) >=Available (=2 3 3) 因为

Request(0,3,4) >=Available (=2 3 3)

所以不能分配。{ Available (=2 3 3)就是work =2 3 3，不同的表达而已，但是第二问就是要这样子写才规范，Request(0,3,4)表示对三种资源的需求，所剩资源必须完全满足需求才可以分配，否则不行}

（3）P4:Request(2,0,1)

因为P2不能分得资源，推迟了，则现在所剩仍是Available (=2 3 3)，

Request(2,0,1)<=Need（=2 2 1）(第一条件满足了)

Request(2,0,1)<=Available (=2 3 3);（第二条件也满足），接下来还得进行安全性算法检查，先假定可以分配，若分配后得到我work（0 3 2），看这时的（0 3 2 ）是否能够让新的形成的序列达到安全，若是安全就分配，不安全就推迟，经计算是可以形成安全系列{P4，P5，P3，P2，P1}则可以分配。

（4）在(3)的基础上P1:Request(0,2,0)

在(3)的基础上，即P4运行完成释放了所有资源，这时候work的值发生了变化

Work=4 3 7（它是原来的work加上已分配得到的），

Request(0,2,0)<= Need（=3 4 7）;

Request(0,2,0) <= Available (=4 3 7),都满足，但是此时的work是上次分配后剩下的，即work（=0 3 2）,若再分配（0 2 0 ）出去，剩下的work（0 1 2）就不能再满足任何的进程需要了，所以不可以可以分配。