# 实验一 多级队列调度算法

设 RQ 分为 RQ1 和 RQ2，RQ1 采用轮转法，时间 q=7.

RQ1>RQ2,RQ2 采用短进程优先调度算法。

测试数据如下：RQ1: P1-P5, RQ2: P6-P10

| 进程 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 运行时间 | 16 | 11 | 14 | 13 | 15 | 21 | 18 | 10 | 7 | 14 |
| 已等待时间 | 6 | 5 | 4 | 3 | 2 | 1 | 2 | 3 | 4 | 5 |

实现描述：

```
  typedef    struct    tag_pcb
        {    char    name[8];
               int need;//须运行的时间
               int turn;//周转时间
              struct    tag_pcb    *next;
     } PCB;
PCB    * RQ1,*RQ2, *Finish;
int clock=0;    //时钟
```

```
main ( )
{    输入 RQ1;
     输入 RQ2; （最好从文件读入）
     while（ＲＱ１！＝ＮＵＬＬ）
   {     从 RQ1 中选取一进程 Pi 准备运行;
                 计算其运行的时间 t;
                 clock+=t;    //表示 Pi 运行 t;
                 if (Pi 完成）     计算其 turn;
                 否则       Pi 加入到队尾;
      }
     while（ＲＱ2!＝ＮＵＬＬ）
     {          从 RQ2 中选取一进程 Pi 准备运
行;
                     clock+=Pi.need;
                     计算 Pi 的 turn;
      }
   输出进程的周转时间;
}
```

# 实验二 银行家算法

```
#define   n   5   //进程个数
#define    m   3   //资源种类
int     Available[m],Alloc[n][m],Need[n][m];
main( )
{
     int request[m];
     input( );
     while (1)
     {
          read_req( );
          if   (请求结束)      break;
```

(1) if (!($request_i$<=$Need_i$))　　表示非法请求；

(2) if (!($request_i$<=Available)) 则 Pi 阻塞；

(3)　试探性分配

Available=Available - $Request_i$;

$Alloc_i$=$Alloc_i$+$Request_i$;

$Need_i$=$Need_i$-$Request_i$;

(4)若新状态安全，则实际分配资源给 Pi,否则取消试探性分配。

```
   }
}
```

安全状态判别算法：

(1)设置 Finish=(false,...,false)　work=Available

(2)循环查找满足下列条件的进程 pi　　//最多循环 n 次

Finish[i]=false 且 Needi<=work

(3)若找到则　Finish[i]=true;work=work+Alloc$_i$; 转(2)

(4)若 Finish=(true,...,true) 则安全，否则不安全。

测试数据：m=3:种类型的资源（A,B,C,） 进程个数 n=5

Available=(2,3,3);

| | 已分配资源数量 | | | 资源需求量 | | |
|---|---|---|---|---|---|---|
| | A | B | C | A | B | C |
| P1 | 2 | 1 | 2 | 3 | 4 | 7 |
| P2 | 4 | 0 | 2 | 1 | 3 | 4 |
| P3 | 3 | 0 | 5 | 0 | 0 | 3 |
| P4 | 2 | 0 | 4 | 2 | 2 | 1 |
| P5 | 3 | 1 | 4 | 1 | 1 | 0 |

请求序列如下：

a:　进程 P2 请求资源（0，3，4）

b　进程 P4 请求资源（1，0，1）

c. 进程 P1 请求资源（2，0，1）

d. 进程 P3 请求资源（0，0，2）

# 实验三　动态分区式存贮区管理

设计一个动态分区式存贮区管理程序，要求支持不同的放置策略。如首次、最佳、最坏。

说明：

（1）分区描述器 rd 如下：

| flag | size | next |
|---|---|---|

要求自由主存队列按链表组织。

主存大小假设为 maxsize（单位为节=rd 的大小）。

（2）主程序结构如下：

输入放置策略

申请一块内存作为主存

循环处理用户的请求(包括申请、释放）

申请函数 Addr=Request(size)

释放函数 Release(addr)

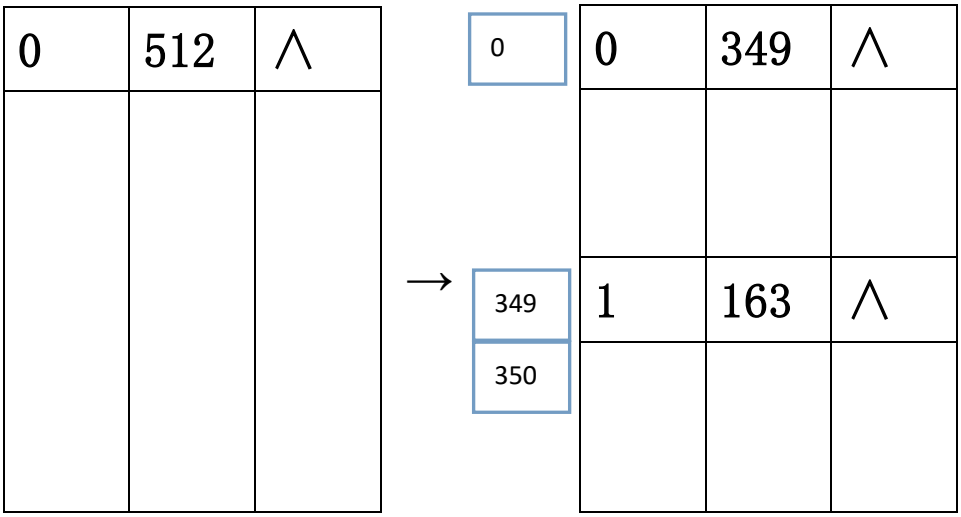
（3）数据实例：maxsize=512

J1 申请 162，J2 申请 64，J3 申请 120，J4 申请 86，J1 完成，J3 完成，J5 申请 72，J6 申请 100，J2 完成，J7 申请 36，J8 申请 60，J4 完成，J9 申请 110，J10 申请 42。备注：

（a）所有大小其单位为节（1 节=rd 的大小）

（b）作业申请 n 节,实际分配的分区大小应为 n+1 节。其中一节作为分区描述器，其他 n 节提供给作业。

（c）已分配区放在高地址处。




开始时　　J1 申请 162 节后 返回地址 350

（d）合并时应考虑四种情况： 假设回收区为 r,上邻为 f1（f1 需搜索自由主存队列）,下邻为 f2（f2 可直接计算）

A)f1 空闲，f2 已分配；

B)f1 已分配，f2 空闲；

C)f1 空闲，f2 空闲；

D)f1 已分配，f2 已分配；

# 实验 1：作业调度

## 1.1 实验目的

1、 对作业调度的相关内容作进一步的理解。

2、 明白作业调度的主要任务。

3、 通过编程掌握作业调度的主要算法。

## 1.2 实验内容

1、假设系统中可同时运行两道作业，给出每道作业的到达时间和运行时间，如下表所示：

| 业名 | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 达时间 | 0 | 2 | 5 | 7 | 12 | 15 | 4 | 6 | 8 | 10 |
| 行时间 | 7 | 10 | 20 | 30 | 40 | 8 | 8 | 20 | 10 | 12 |

2、分别用先来先服务算法、短作业优先和响应比高者优先三种算法给出作业的调度顺序。

3、计算每一种算法的平均周转时间及平均带权周转时间并比较不同算法的优劣。

# 实验 2：磁盘调度

## 2.1 实验目的

1、对磁盘调度的相关知识作进一步的了解，明确磁盘调度的原理。

2、加深理解磁盘调度的主要任务。

3、通过编程，掌握磁盘调度的主要算法。

## 2.2 实验内容

1、对于如下给定的一组磁盘访问进行调度：

| 服务到达 | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 的磁道号 | 30 | 50 | 100 | 180 | 20 | 90 | 150 | 70 | 80 | 10 | 160 | 120 | 40 | 110 |

2、要求分别采用先来先服务、最短寻道优先以及电梯调度算法进行调度。

3、要求给出每种算法中磁盘访问的顺序，计算出平均移动道数。

4、假定当前读写头在 90 号，电梯调度算法向磁道号增加的方向移动。

# 实验 3：熟悉 linux 文件系统调用

## 3.1 实验目的

1. 掌握 linux 提供的文件系统调用的使用方法；
2. 熟悉文件系统的系统调用用户接口；
3. 了解操作系统文件系统的工作原理和工作方式。

## 3.2 实验内容

使用文件系统调用编写一个文件工具 filetools，使其具有以下功能：

1. 创建新文件
2. 写文件
3. 读文件
4. 修改文件权限
5. 查看当前文件权限
0. 退出

提示用户输入功能号，并根据用户输入的功能选择相应的功能。

文件按可变记录文件组织，具体记录内容自行设计。

# 实验 4：进程管理

## 4.1 实验目的

1. 理解进程的概念，明确进程和程序的区别。
2. 理解并发执行的实质。
3. 掌握进程的同步、撤销等进程控制方法。

## 4.2 实验内容

父进程使用系统调用 pipe()建立一个管道，然后使用系统调用 fork()创建两个子进程：子进程 1 和子进程 2

子进程 1 每隔 1 秒通过管道向子进程 2 发送数据：I send message x times.（x 初值为 1，以后发送一次后做加一操作）子进程 2 从管道读出信息，并显示在屏幕上

父进程用系统调用 signal()来捕捉来自键盘的中断信号 SIGINT（即按 Ctrl+C 键,）；当捕捉到中断信号后，父进程用系统调用 kill()向两个子进程发出信号，子进程捕捉到信号后分别输出如下信息后终止： Child Process 1 is killed by Parent!

Child Process 2 is killed by Parent!

父进程等待两个子进程终止后，释放管道并输出如下的信息后终止

Parent Process is Killed!

# 实验 5：请求分页系统中的置换算法

## 5.1 实验目的

1. 了解虚拟存储技术的特点；
2. 掌握请求分页系统的页面置换算法。

## 5.2 实验内容

1.通过如下方法产生一指令序列，共 320 条
指令。

A. 在[1，32k-2]的指令地址之间随机选取一起点,访问 M；
B. 顺序访问 M+1；
C. 在[0，M-1]中随机选取 M1，访问 M1；
D. 顺序访问 M1+1；
E. 在[M1+2，32k-2]中随机选取 M2，访问 M2；
F. 顺序访问 M2+1；
G. 重复 A—F，直到执行 320 次指令。

2. 指令序列变换成页地
址流设（1）页面大小
为1K；
（2） 分配给用户的内存页块个数为 4 页到 32 页,步长为 1 页；
（3）用户虚存容量为 32K。

3. 计算并输出下述各种算法在不同内存页块下的命中率。
A. 先进先出（FIFO）页面置换算法
B. 最近最久未使用（LRU）页面置换算法
C. 最佳（Optimal）页面置换算法

# 实验 6：进程通信

## 6.1 实验目的

1. 理解管道机制、消息缓冲队列、共享存储区机制进行进程间的通信；
2. 理解通信机制。

## 6.2 实验内容

编写一主程序可以由用户选择如下三种进程通信方式：

1. 使用管道来实现父子进程之间的进程通信

子进程向父进程发送自己的进程标识符，以及字符串“is sending a message to parent”。父进程则通过管道读出子进程发来的消息，将消息显示在屏幕上，然后终止。

2. 使用消息缓冲队列来实现 client 进程和server 进程之间的通信

server 进程先建立一个关键字为 SVKEY（如75）的消息队列，然后等待接收类型为 REQ（例如 1）的消息；在收到请求消息后，它便显示字符串“serving for client”和接收到的client 进程的进程标识数，表示正在为 client 进程服务；然后再向 client 进程发送应答消息，该消息的类型是 client 进程的进程标识数，而正文则是 server 进程自己的标识 ID。client 进程则向消息队列发送类型为 REQ 的消息（消息的正文为自己的进程标识 ID） 以取得 sever 进程的服务，并等待server 进程发来的应答；然后显示字符串“receive reply from”和接收到的 server 进程的标识 ID。

3. 使用共享存储区来实现两个进程之间的进程通信

进程 A 创建一个长度为 512 字节的共享内存，并显示写入该共享内存的数据；进程 B 将共享内存附加到自己的地址空间，并向共享内存中写入数据。