实验1：熟悉Linux系统

熟悉Linux系统的基本操作和开发环境

目的

- 熟悉和掌握**Linux**系统基本命令，熟悉**Linux**编程环境，为以后的实验打下基础。
  - 启动、退出、**ls**（显示目录内容）、**cp**（文件或目录的复制）、**mv**（文件、目录更名或移动）、**rm**（删除文件或目录）、**mkdir**（创建目录）、**rmdir**（删除空目录）、**cd**（改变工作目录）...
  - **C**语言编辑、编译

内容及要求

- 熟练掌握**Linux**基本文件命令；
- 掌握**Linux**编辑程序、对源代码进行编译、连接、运行及调试的过程；
- 认真做好预习，书写预习报告；
- 实验完成后要认真总结、完成实验报告。

预习报告要求

- 题目，目的，要求；
- 对所要熟悉和掌握的命令的理解；
- 在**Linux**环境下编制、调试源程序的过程。

实验报告要求

- 题目，目的，要求；
- 通过实验，掌握**Linux**的命令及其含义；
- 在**Linux**环境下编制、调试源程序的实际过程（每一步的具体说明）。

实验2：进程状态

模拟进程状态转换及其**PCB**的变化

目的

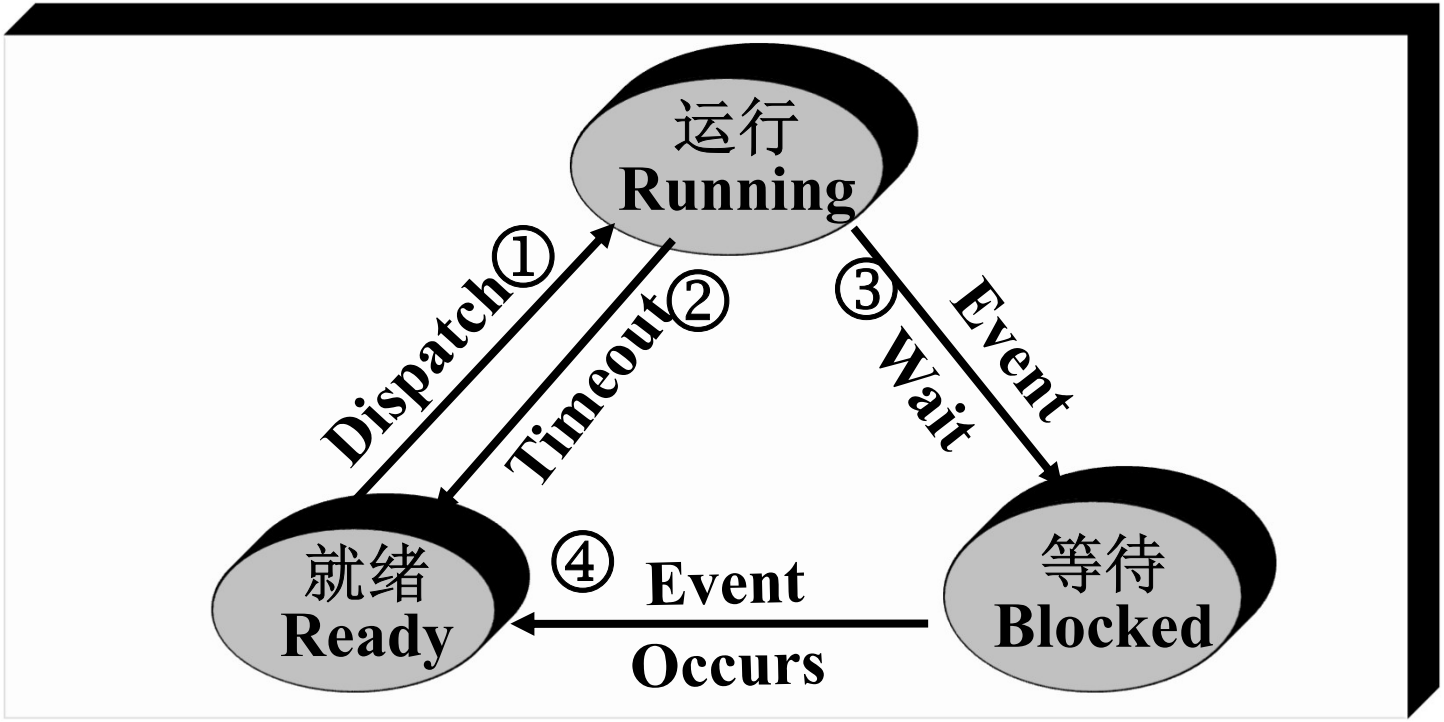
- 自行编制模拟程序，通过形象化的状态显示，使学生理解进程的概念、进程之间的状态转换及其所带来的**PCB**内容、组织的变化，理解进程与其**PCB**间的一一对应关系。

内容及要求

- 设计并实现一个模拟进程状态转换及其相应**PCB**组织结构变化的程序；
- 独立设计、编写、调试程序；
- 程序界面应能反映出在模拟条件下，进程之间状态转换及其对应的**PCB**组织的变化。

内容及要求（续）

- 进程的状态模型（三状态、五状态、七状态或其它）可自行选择



基本状态间的转换

内容及要求（续）

- 代码书写要规范，要适当地加入注释；
- 鼓励在实验中加入新的观点或想法，并加以实现；
- 认真进行预习，完成预习报告；
- 实验完成后，要认真总结，完成实验报告。

预习报告要求

- 题目，目的，要求；
- 初步设计的程序流程图；
- 初步的程序源代码、文档注释及必要的文字说明；
- 预期的程序运行结果。

实验报告要求

- 题目，目的，要求；
- 程序流程图；
- 使用的数据结构及其说明；
- 程序源代码、文档注释及文字说明；
- 运行结果及其说明；
- 程序使用说明。

实验3：进程同步和互斥

生产者和消费者问题模拟

目的

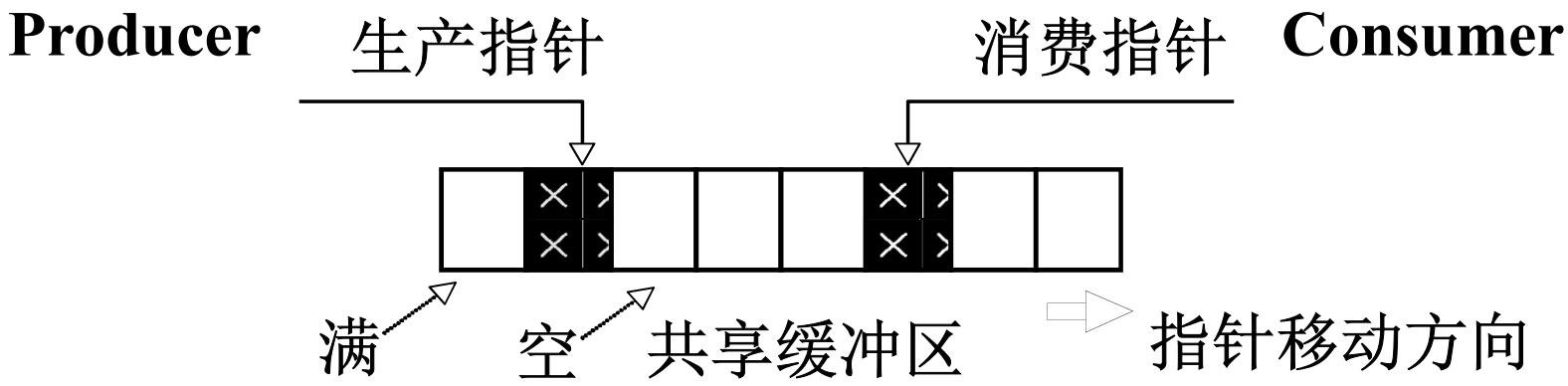
- 调试、修改、运行模拟程序，通过形象化的状态显示，使学生理解进程的概念，了解同步和通信的过程，掌握进程通信和同步的机制，特别是利用缓冲区进行同步和通信的过程。通过补充新功能，使学生能灵活运用相关知识，培养创新能力。

内容及要求

- 调试、运行给出的程序，从操作系统原理的角度验证程序的正确性；
- 发现并修改程序中的原理性错误或不完善的地方；
- 鼓励在程序中增加新的功能。完成基本功能的，得基本分；添加新功能的加分；
- 在程序中适当地加入注释；
- 认真进行预习，阅读原程序，发现其中的原理性错误，完成预习报告；
- 实验完成后，要认真总结，完成实验报告。

程序说明

- 所给程序模拟两个进程，即生产者（**producer**）进程和消费者**(Consumer)**进程工作；
- 生产者每次产生一个数据，送入缓冲区中；
- 消费者每次从缓冲区中取走一个数据。



程序说明

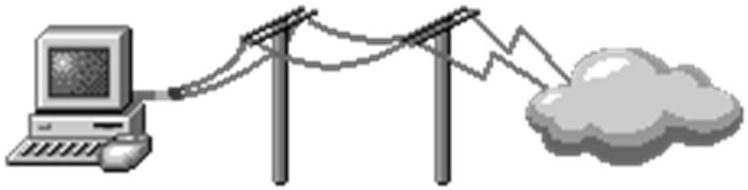
- 缓冲区可以容纳8个数据；
- 因为缓冲区是有限的，因此当其满了时生产者进程应该等待；当消费者取走一个数据后，应唤醒正在等待的生产者进程；
- 当缓冲区空时，消费者进程应该等待；当生产者向缓冲区放入了一个数据时，应唤醒正在等待的消费者进程。
- 这就是生产者和消费者之间的同步

程序说明（续）

- 每次写入和读出数据时，都将读和写指针加一。当指针到达缓冲区尾，重新将指针退回起点；
- 为简单起见，每次产生的数据为**0-99**的整数，从**0**开始，顺序递增；
- 两个进程的调度是通过运行者使用键盘来实现的。

程序使用的数据结构

- 进程控制块：包括进程名，进程状态和执行次数。
- 缓冲区：一个整数数组
- 缓冲区说明块：包括类型，读指针，写指针，读等待指针和写等待指针。



程序使用说明

- 启动程序后，如果使用‘**p**’键则运行一次生产者进程；
- 使用‘**c**’键则运行一次消费者进程；
- 使用‘**e**’键则退出程序；
- 通过屏幕可以观察到两个进程的状态和缓冲区变化的情况。

预习报告要求

- 题目，目的，要求
- 初步理解的程序流程图
- 拟修改、补充的源程序，指出原来程序错误所在，并说明程序中拟加入的功能。
- 新加功能预期运行结果的说明

实验报告要求

- 题目，目的，要求
- 程序流程图
- 最终运行的源程序及修改、补充的说明
- 运行结果及其说明

实验4：进程的管道通信

编程实现进程的管道通信程序

目的

- 加深对进程概念的理解，明确进程和程序的区别；
- 学习进程创建的过程，进一步认识并发执行的实质；
- 分析进程争用资源的现象，学习解决进程互斥的方法；
- 学习解决进程同步的方法；
- 掌握**Linux**系统进程间通过管道通信的具体实现方法。

相关知识

- 基本概念
  - 进程的概念
  - 进程与程序的区别
  - 并发执行的概念
  - 进程互斥的概念
  - 进程通信的基本原理
- 系统调用
  - 设置系统调用号：设置多条系统调用命令，并赋予每条系统调用命令一个唯一的系统调用号

相关知识

- 系统调用
  - 处理系统调用：**OS**中有一张系统调用入口表，表中每个表目对应一条系统调用命令，包含该系统调用自带参数的数目、系统调用命令处理程序的入口地址等。**OS**内核便是根据所输入的系统调用号在该表中查找到相应的系统调用，进而转入它的入口地址去执行系统调用程序。
  - **Linux**的系统调用机制：通过中断机制实现。

相关知识

- 相关函数
  - **Fork()**：创建一个新进程
    - 调用格式　**int fork();**
    - 正确返回：**=0**，创建子进程，从子进程返回的**ID**
      >0，从父进程返回的子进程的**ID**
    - 错误返回：**=-1**，创建失败
  - 由于父进程和子进程分别独立地进入就绪队列等待调度，所以谁首先得到调度是不确定的，因此，谁首先从**fork()**返回、继续执行后面的语句也是不确定的。

相关知识

- 相关函数
  - **wait()**：控制父进程与子进程的同步
    - 父进程调用该函数，则父进程被阻塞，进入等待队列，等待子进程结束。子进程结束时，产生一个终止状态字，系统向父进程发出**SIGCHILD**信号。当接到信号后，父进程提取子进程的终止状态字，从**wait()**函数返回继续执行原来的程序。
    - 调用格式：**#include <sys/type.h>**

      #include <sys/wait.h>

      (pid_t) wait (int *statloc);
    - 正确返回：**>0**：子进程的**ID**；**=0**：其它
    - 错误返回：**=-1**，调用失败

相关知识

- 相关函数
  - **exit()**：结束进程
    - 正常终止时，该函数返回进程结束状态。
    - 调用格式：**#include <stdio.h>**

      void exit (int status);

      其中，**status**为进程结束状态
  - **Kill()**：删除执行中的进程
    - 调用格式：**kill (int PID, int IID);**
    - 其中，**PID**为要被杀死的进程号，**IID**为向被杀死的进程发送的中断号

相关知识

- 相关函数
  - **signal()**：允许调用进程控制软中断信号的处理
    - 调用格式：**#include <signal.h>**

      int sig; …
  - **pipe()**：创建一个管道
    - 调用格式：**#include <unistd.h>**

      pipe (int fp[2]);
    - 其中，**fp[2]**是供进程使用的文件描述符数组，**fp[1]**用于写，**fp[0]**用于读
    - 正确返回：**0**，调用成功
    - 错误返回：**-1**，调用失败

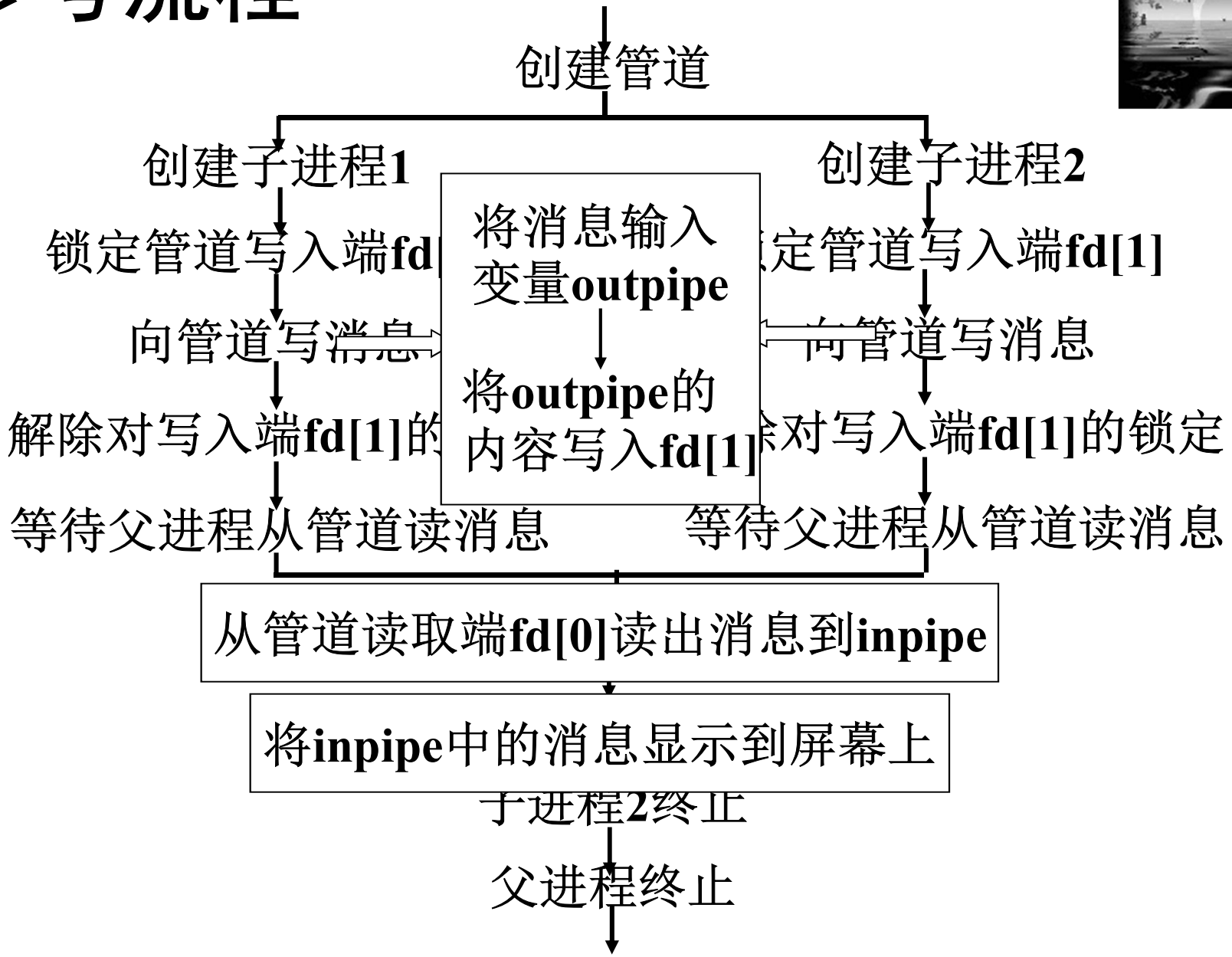
相关知识

- 相关函数
  - **sleep()：睡眠等待**
  - **lockf(管道ID,LID,0)**
    - **LID=1，实现对管道的加锁操作**
    - **LID=0，解除管道的锁定**
  - **read()**
  - **write()**
  - **sprintf()**
  - **……**

实验内容

- 使用系统调用**pipe()**建立一条管道线，两个子进程分别向管道写一句话（写的内容自己定，但要有该进程的一些信息）
- 父进程从管道中读出来自两个子进程的消息，显示在屏幕上。
- 要求：父进程首先接收子进程**p1**发来的消息，然后再接收子进程**p2**发来的消息

参考流程


创建管道
创建子进程1
创建子进程2
锁定管道写入端fd
定管道写入端fd[1]
将消息输入
变量outpipe
将outpipe的
内容写入fd[1]
向管道写消息
向管道写消息
解除对写入端fd[1]的
除对写入端fd[1]的锁定
等待父进程从管道读消息
等待父进程从管道读消息
从管道读取端fd[0]读出消息到inpipe
将inpipe中的消息显示到屏幕上
子进程2终止
父进程终止

实验要求

- 这是一个设计型实验，要求自行、独立编制程序；
- 两个子进程要并发执行；
- 实现管道的*互斥*使用。当一个子进程正在对管道进行写操作时，另一个欲写入管道的子进程必须等待。 使用系统调用**lockf(fd[1],1,0)**实现对管道的加锁操作，用**lockf(fd[1],0,0)**解除对管道的锁定；
- 实现父子进程的*同步*，当父进程试图从一空管道中读取数据时，便进入等待状态，直到子进程将数据写入管道返回后，才将其唤醒。

预习报告要求

- 题目、目的、要求
- 规范的程序流程图
- 编制的源程序
- 拟运行结果及说明
- 查找相关函数的格式、用法

实验报告要求

- 题目、目的、要求
- 修改后的程序流程图
- 调试正确的源程序
- 运行结果及说明
- 回答问题：
  - 进程间的互斥表现在哪里？
  - 进程间的同步表现在哪里？
  - 你的程序采用什么方法实现上述关系？

实验5：页面置换算法

编程实现**FIFO**和**LRU**算法

目的

- 进一步理解父子进程之间的关系
- 理解内存页面调度的机理
- 掌握页面置换算法的实现方法
- 通过实验比较不同调度算法的优劣
- 培养综合运用所学知识的能力

页面置换算法是虚拟存储管理实现的关键，通过本次试验理解内存页面调度的机制，在模拟实现**FIFO**、**LRU**等经典页面置换算法的基础上，比较各种置换算法的效率及优缺点，从而了解虚拟存储实现的过程。将不同的置换算法放在不同的子进程中加以模拟，培养综合运用所学知识的能力。

内容及要求

- 这是一个综合型实验，要求在掌握父子进程并发执行机制和内存页面置换算法的基础上，能综合运用这两方面的知识，自行编制程序
- 程序涉及一个父进程和两个子进程。父进程使用**rand()**函数随机产生若干随机数，经过处理后，存于一数组**Acess_Series[]**中，作为内存页面访问的序列。两个子进程根据这个访问序列，分别采用**FIFO**和**LRU**两种不同的页面置换算法对内存页面进行调度。要求：

内容及要求

- 每个子进程应能反映出页面置换的过程，并统计页面置换算法的命中或缺页情况。
  - 设缺页的次数为**diseffect**。总的页面访问次数为**total_instruction**。
  - 缺页率 = **disaffect/total_instruction**
  - 命中率 = **1- disaffect/total_instruction**
- 将为进程分配的内存页面数**mframe**作为程序的参数，通过多次运行程序，说明**FIFO**算法存在的**Belady**现象。

相关的系统调用或函数

- **fork()** 用于创一个子进程
  - 格式：**int fork()**
  - 返回值：在子进程中返回**0**；在父进程中返回所创建的子进程的**ID**值；当返回**-1**时，创建失败。
- **wait()** 常用来控制父进程与子进程的同步
  - 在父进程中调用**wait()**，则父进程被阻塞，进入等待队列，等待子进程结束。当子进程结束时，父进程从**wait()**返回继续执行原来的程序。
  - 返回值：大于**0**时，为子进程的**ID**值；等于**-1**时，调用失败。

相关的系统调用或函数

- **exit()** 是进程结束时最常调用的
  - 格式：**void exit(int status);** 其中，**status**为进程结束状态。
- **sleep()** 调用进程睡眠若干时间，之后唤醒
  - 格式：**sleep(int t);** 其中**t**为睡眠时间
- **rand()** 返回一个随机整数。需要包含文件 **<stdlib.h>**

主要数据结构（仅供参考）

- 存放页面访问序列的数组**(Acess_Series[total_instruction])**
- 用一个结构数组**M_Frame[]**记录为进程分配的内存页面的使用情况。在**M_Frame[]**中还可记载各页面进入内存或被访问的先后顺序（如可使**M_Frame[0]**总是最先进入或最久未被访问的页面）。

```
struct one_frame {
        int page_no;
        char flag;
};
struct one_frame M_Frame[frame_num];  /
```

这里**frame_num**是给一个进程分配的最大的内存页面数

程序流程（仅供参考）

- 父进程：
  - 随机产生内存访问页面序列，存于数组**Acess_Series[total_instruction]**中；
  - 数据结构**M_Frame**的初始化；
  - 分别创建两个子进程；
  - 等待子进程执行结束，退出。

程序流程

- 子进程：
  - 读页面访问序列**Acess_Series[]**，若序列中已无下一个元素，转**5**）；否则取出序列中的下一个元素作为下次要访问的页面；
  - 如果待访问的页面在内存中（即在**M_Frame[]**中找到），则不发生缺页，命中率加**1**，转**1)**，注意**LRU**算法中要调整该页在数组中的位置；
  - 否则就要将这页调入内存，通过修改相应的数据结构**M_Frame[]**反映出来。首先看**M_Frame[]**中有无空闲页面，如果有，将待访问页面的页号以及被占用的信息写入数组中适当位置，如要统计缺页情况，缺页次数**diseffect**加**1**，返回**1**）；
  - 如果**M_Frame[]**中的所有页面均被占满，则淘汰**M_Frame[0]**，装入待访问页，重新调整各页面在数组中的位置。如要统计缺页情况，缺页次数**diseffect**加**1**，返回**1**）；
  - 所有页面均已访问完成，统计命中率或缺页率；

预习报告要求

- 题目，目的，要求
- 初步的程序流程图
- 初步的程序源代码、文档注释及必要的文字说明
- 预期的程序运行结果

实验报告要求

- 题目，目的，内容，要求
- 程序流程图
- 程序源代码、文档注释及文字说明
- 运行结果及其说明
- 回答以下问题：
  - 父进程空间与子进程空间的关系。
  - 通过完成实验，根据你的体会，阐述虚拟存储器的原理。
  - 写出**FIFO**算法中出现**Belady**现象的内存页面访问序列

Thank You Very Much!