论 文（设 计）任 务 书

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 课程编号 | j4011720y0 | 课程名称 | | 操作系统课程设计 | | | 周数 | 1 |
| 实施地点 | 厚为楼801 | | | 班级 |  | | 人数 | 72 |
| 起止时间 |  | | 形式 | √集中 □分散 | | 指导教师 |  | |
| 论文  （设计）  进度  安排 | 3月24日：课程设计动员，设计题目及小组确定。  3月24日—6月9日（暂定）：按要求完成课程设计全部内容，并于校友邦提交课程设计报告书。  说明：学期中布置课程设计任务，提前准备、选题、编程、调试，保障课程设计顺利完成。 | | | | | | | |
| 论文  （设计）  内容 | 根据课程设计目的和要求，由教师指定题目，题目的难度和份量要适中，使大多数学生通过努力能按时完成。  使用所学操作系统知识点进行题目分析、算法及最终的编码设计，然后上机调试通过，并撰写课程设计报告。 | | | | | | | |
| 要求  (包括纪律要求和报告书要求) | （1）按照授课时间按时到实验室进行课程设计，不迟到，不早退。  （2）要求用操作系统的编程思想来完成系统的设计；系统设计要具有实用性。  （3）编程要简单清楚，实用，功能较全。  （4）课程设计完成后，每人上交一份设计报告，必须用专用的课程设计报告纸书写，要求具有以下内容：题目，原理叙述、设计有关的基本规则，流程图，源程序清单，运行结果，调试情况分析，设计总结体会。  （5）课程设计报告严禁抄袭，否则按不及格论。  （6）采用报告评阅、代码程序运行和回答问题的方式进行考核，结合课程设计报告进行成绩评定。  在课程设计周的最后一天按规范的格式要求撰写和提交《数据结构课程设计报告》。 | | | | | | | |

注：本表按自然班填写。于动员时发给学生。不够纸请另附页。

**附：**

**《操作系统课程设计》**

**课程设计内容及考核要求、评分标准**

**一、选题**

在给定的选题中，选择一个题目作为课程设计内容进行实现：

①WFS文件系统；②筛素数进程版；③筛素数线程版。

**二、课程设计内容及考核要求**

(1)要求用操作系统编程思想来完成系统的设计；系统设计要具有实用性。

(2)编程要简单清楚，实用，功能较全。

(3)课程设计完成后，每人上交一份设计报告，必须用专用的课程设计报告纸书写，要求具有以下内容：题目，原理叙述、设计有关的基本规则，流程图，源程序清单，运行结果，调试情况分析，设计总结体会。

(4)课程设计报告严禁抄袭，否则按不及格论。

(5)采用报告评阅、代码程序运行和回答问题的方式进行考核，结合课程设计报告进行成绩评定。

**三、评分标准**

本课程设计主要考核综合程序开发和课程设计报告。具体评分标准如下：

1．文档结构合理（10%）

2. 综合程序开发（60%）

（1）需求分析（5）

（2）界面设计（10）

（3）程序功能及实现（30）

（4）程序代码及理解、设计与开发（5）

3．论述详细（20%）

4．格式规范（10%）

**附2：**

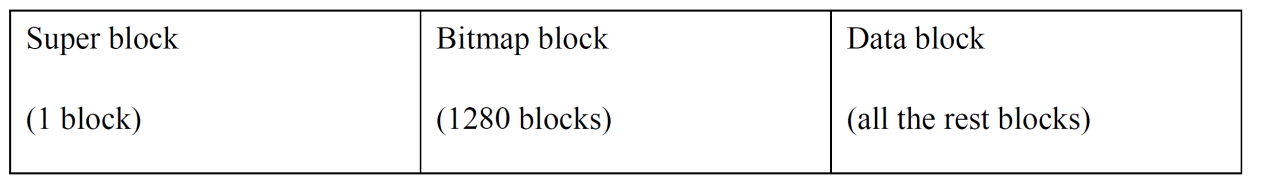
**设计题目选题表**

**一、WFS文件系统**

【问题描述】

WFS 文件系统是一个 FUSE 应用，也就是说WFS是利用FUSE框架创建的一个可以在linux上运行的真正的文件系统，下面是对WFS的基本介绍：

WFS 文件系统实现在一个大小为5M的磁盘映像文件diskimg上，即这个diskimg文件作为磁盘设备承载WFS。将这个文件放在/tmp目录下。WFS 文件系统的布局如下。假定磁盘的每块大小为 512 字节。所以平均每 block 大小为 512 bytes，然后整个5MB文件共 10240 块：



* 超级块是文件系统的第一块，描述整个文件系统。其结构如下：

struct super\_block {

long fs\_size; //size of file system, in blocks（以块为单位）

long first\_blk; //first block of root directory（根目录的起始块位置，以块为单位）

long bitmap; //size of bitmap, in blocks（以块为单位）

};

* 目录

目录也被同时处理为文件。每个目录项格式如下：

struct file\_directory {

char fname[MAX\_FILENAME + 1]; //文件名 (plus space for nul)

char fext[MAX\_EXTENSION + 1]; //扩展名 (plus space for nul)

time\_t atime; /\* 上次访问时间 \*/

time\_t mtime; /\*上次修改时间 \*/

int uid; //

int mode; //

size\_t fsize; //文件大小（file size）

long nStartBlock; //目录开始块位置（where the first block is on disk）

int flag; //indicate type of file. 0:for unused; 1:for file; 2:for directory

};

* 文件名格式为8.3。所谓8.3格式短文件名规范，就是型如 ABCDEFGH（目录）或者ABCDEFGH.exe（文件）这样的名称——“8”是指文件名或目录名的主体部分小于等于8个字节；“3”是指文件名的扩展名部分小于等于3个字节。另外还有一点，就是8.3文件名的有效字符不包括空格等特殊字符。
* WFS文件由有一系列链接的块组成，每一块对应于一个磁盘块，其格式如下：

struct data\_block {

size\_t size; //文件使用了这个块里面的多少Bytes

long nNextBlock; //（该文件太大了，一块装不下，所以要有下一块的地址） long的大小为4Byte

char data[MAX\_DATA\_IN\_BLOCK];// And all the rest of the space in the block can be used for actual data storage.

};

由此可以看出，这个WFS的结构更像FAT，与ext2文件系统的结构有一定的差距。

* WFS实现的函数（系统调用）如下表所示：

|  |  |  |
| --- | --- | --- |
| WFS\_getattr | Description: | This function should look up the input path to determine if it is a directory or a file. If it is a directory, return the appropriate permissions. If it is a file, return the appropriate permissions as well as the actual size. This size must be accurate since it is used to determine EOF and thus read may not be called. |
| UNIX  Equivalent: | man -s 2 stat |
| 返回值: | 0 on success, with a correctly set structure  -ENOENT if the file is not found |
| WFS\_readdir | Description: | This function should look up the input path, ensuring that it is a directory, and then list the contents.  To list the contents, you need to use the filler() function. For example: filler(buf, ".", NULL, 0); adds the current directory to the listing generated by ls -a In general, you will only need to change the second parameter to be the name of the file or directory you want to add to the listing. |
| UNIX  Equivalent: | man -s 2 readdir  However it’s not exactly equivalent |
| 返回值: | 0 on success  -ENOENT if the directory is not valid or found |
| WFS\_mkdir | Description: | This function should add the new directory to the root level, and should update the .目录 file |
| UNIX  Equivalent: | man -s 2 mkdir |
|  | 0 on success  -ENAMETOOLONG if the name is beyond 8 chars -EPERM if the directory is not under the root dir only  -EEXIST if the directory already exists |
| WFS\_rmdir |  | Deletes an empty directory |
| UNIX  Equivalent: | man -s 2 rmdir |
| 返回值: | 0 read on success  -ENOTEMPTY if the directory is not empty  -ENOENT if the directory is not found  -ENOTDIR if the path is not a directory |
| WFS\_mknod | Description: | This function should add a new file to a subdirectory, and should update the .目录 file appropriately with the modified directory entry structure. |
| UNIX  Equivalent: | man -s 2 mknod |
| 返回值: | 0 on success  -ENAMETOOLONG if the name is beyond 8.3 chars  -EPERM if the file is trying to be created in the root dir  -EEXIST if the file already exists |
| WFS\_write | Description: | This function should write the data in buf into the file denoted by path, starting at offset. |
| UNIX  Equivalent: | man -s 2 write |
| 返回值: | size on success  -EFBIG if the offset is beyond the file size (but handle appends) |
| WFS\_read | Description: | This function should read the data in the file denoted by path into buf, starting at offset. |
| UNIX  Equivalent: | man -s 2 read |
| 返回值: | size read on success  -EISDIR if the path is a directory |
| WFS\_unlink | Description: | Delete a file |
| UNIX  Equivalent: | man -s 2 unlink |
| 返回值: | 0 read on success  -EISDIR if the path is a directory  -ENOENT if the file is not found |

【实现要求】

WFS目前已经实现了最简单最基础的功能，但仍有许多不足之处有待于你的改进：

1. 文件目录里缺乏文件权限、文件主、时间日期等信息，请予以补全。
2. 文件的大小目前最大只有一个磁盘块大小，请实现文件的大小可以是多块。
3. WFS目前只支持一级目录，即所以有的文件均在根目录下，请在WFS下实现多级目录。
4. 按照目前WFS目前的设计，文件的访问是一种顺序访问的方式，请改进WFS的数据访问方式为随机访问。

**注意：**关于更多Fuse框架以及WFS的相关介绍，请查看对应如下链接文档资料



**二、筛素数进程版**

【问题描述】

寻找大素数在信息安全领域有着广泛的应用，但这也是一个计算密集型的任务。本课程设计就是利用并行的方法，充分利用多核CPU的能力，快速筛选素数。该任务需要分别使用**传统算法，多进程模型，进程池模型**计算求得30000000到30000200之间的所有素数，输出这些素数，记录计算时间，比较不同模型之间的性能差异，并加以解释。

【实现要求】

1. **实现传统方法求素数：**编写一个程序primer\_pro1.c，使用暴力计算法计算30000000到30000200之间的所有素数，并输出这些素数。运行10次primer\_pro1，记录每次运行的时间，并计算平均运行时间。

提示：

* 记录运行时间时，需要把你的标准输出重定向到/dev/null设备；
* 查看程序运行时间可用time命令。

1. **实现多进程计算素数：**编写一个程序primer\_pro2.c，程序运行开始时，创建200个子进程，父进程等待这200个子进程结束；200个子进程分别判断30000000到30000200之间的200个数是否为素数，如果是素数则输出这个数。运行10次primer\_pro2，记录每次运行的时间，并计算平均运行时间。
2. **实现进程池计算素数：**编写一个程序primer\_pro3.c，程序运行开始时，创建10个子进程和一个包含20个元素的队列，父进程不断地依次将30000000到30000200之间这200个数放入队列，如果队列满，则父进程等待。

* 队列的定义可参考如下：

#define MAXSIZE 20

struct quequ\_st{

int finished; //父进程是否已将所有200个整数放入了队列

    int head; //队列头

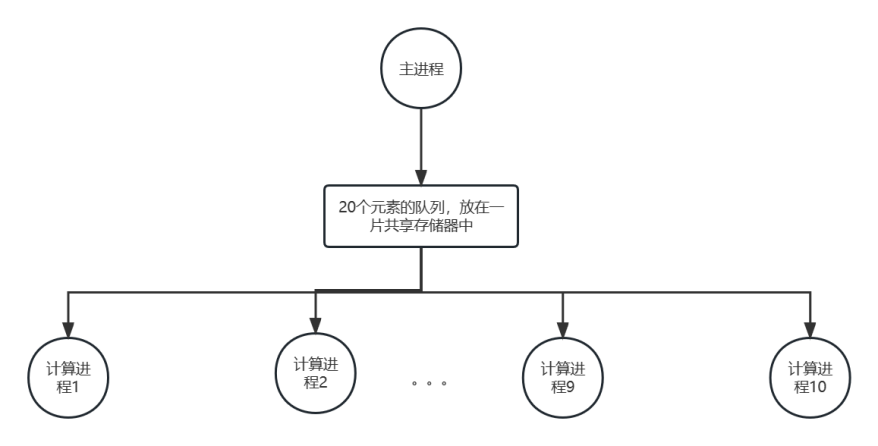
    int tail; //队列尾

    char data[MAXSIZE]; //队列数据

    int count; //当前队列中数据个数

};

* 队列成员finished初始值为0，当父进程将全部200个整数全部放入队列后，将finished置为1，并等待10个子进程结束。
* 子进程依次不断地从队列中取出一个整数，并判断这个整数是不是素数，如果是素数，则输出这个进程的进程号以及这个整数。当然，每次从队列中取数之前，首先需要判断finished是否为1并且队列是否为空，如果是，则子进程结束。
* 运行模型可参考如下图所示：



由于父子进程各自有自己独立的地址空间，我们需要将队列数据结构放在一个父子进程都能够访问到的地方，为此，父进程需要首先创建一块共享存储器，将其附接在自己的地址空间中，将队列数据结构定义于其中（可用指针的强制类型装换）。子进程首先取得父进程创建的共享存储器，并将其附接在自己的地址空间中。当然这件事情需要有一个确定的先后顺序，即父进程需要先创建共享存储器并附接，子进程才能取得这个共享存储器并附接。请自行使用合适的进程同步互斥工具保证这一顺序的正确性。

所有这些事情完成后，父子进程各自开始工作。当然，父子进程对队列的访问应当是互斥的，请使用**信号量**保证这一互斥。本课程设计中，所有的同步互斥工具**推荐**使用System V IPC。

* 运行10次primer\_pro3，记录每次运行的时间，并计算平均运行时间。

1. **数据收集整理：**填写下面两个表，并要求课程设计报告中也需要包括上述两个表

表1 原始数据表

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 平均 |
| primer\_pro1 |  |  |  |  |  |  |  |  |  |  |  |
| primer\_pro2 |  |  |  |  |  |  |  |  |  |  |  |
| primer\_pro3 |  |  |  |  |  |  |  |  |  |  |  |

表中1,2，...，10分别代表第1次，...，第10次运行时间的测量值。

表2 性能比较表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| primer\_pro1平均运行时间 | primer\_pro2平均运行时间 | primer\_pro3平均运行时间 | CPU | 核心数（虚拟机） | 内存大小 |
|  |  |  |  |  |  |

分析你的性能比较表，在课程报告中回答以下问题：

1. 哪一个程序运行最快？为什么？
2. 增加核心数是否有助于提高程序的运行速度？
3. 增加内存的大小是否有助于提高程序的运行速度？
4. primer\_pro3中增加队列的大小是否有有助于提高程序的运行速度？

**三、筛素数进程版**

【问题描述】

寻找大素数在信息安全领域有着广泛的应用，但这也是一个计算密集型的任务。本课程设计就是利用并行的方法，充分利用多核CPU的能力，快速筛选素数。该任务需要分别使用**传统算法，多线程模型，线程池模型**计算求得30000000到30000200之间的所有素数，输出这些素数，记录计算时间，比较不同模型之间的性能差异，并加以解释。

【实现要求】

1. **传统方法求素数：**编写一个程序primer\_thr1.c，使用暴力计算法计算30000000到30000200之间的所有素数，并输出这些素数。运行10次primer\_thr1，记录每次运行的时间，并计算平均运行时间。

提示：

* 记录运行时间时，需要把你的标准输出重定向到/dev/null设备；
* 查看程序运行时间可用time命令。

1. **多线程计算素数：**编写一个程序primer\_thr2.c，程序运行开始时，创建200个子线程，主线程等待这200个子线程结束；200个子线程分别判断30000000到30000200之间的200个数是否为素数，如果是素数则输出这个数。运行10次primer\_thr2，记录每次运行的时间，并计算平均运行时间。
2. 线程池计算素数：编写一个程序primer\_thr3.c，程序运行开始时，创建10个子线程和一个包含20个元素的队列，主线程不断地依次将30000000到30000200之间这200个数放入队列，如果队列满，则主线程等待。

* 队列的定义可参考如下：

#define MAXSIZE 20

struct quequ\_st{

int finished; //主线程是否已将所有200个整数放入了队列

    int head; //队列头

    int tail; //队列尾

    char data[MAXSIZE]; //队列数据

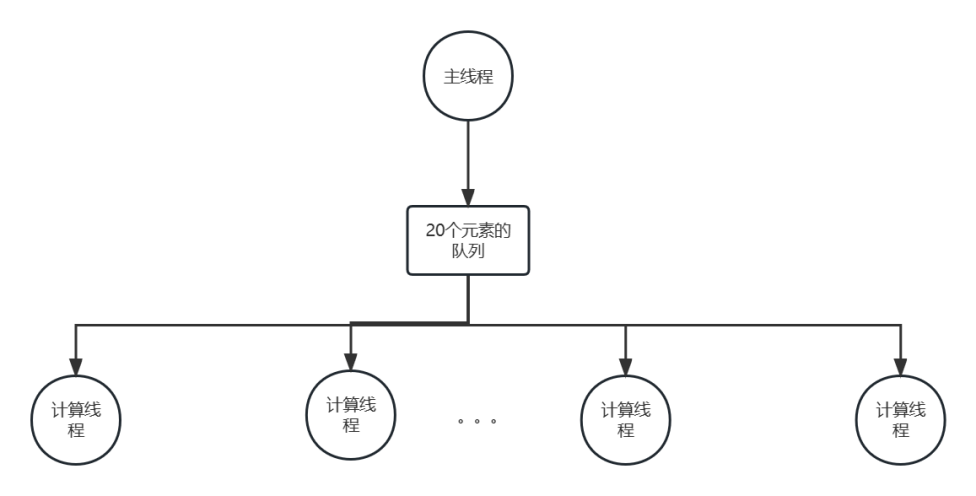
    int count; //当前队列中数据个数

    pthread\_mutex\_t mutex; //

    pthread\_cond\_t cond; //

};

* 队列的实现可以采用数组的形式，也可以采用链表的形式，上述代码采用的是数组的形式。队列成员finished初始值为0，当主线程将全部200个整数全部放入队列后，将finished置为1，并等待10个子线程结束。子线程依次不断地从队列中取出一个整数，并判断这个整数是不是素数，如果是素数，则输出这个线程的线程号以及这个整数。当然，每次从队列中取数之前，首先需要判断finished是否为1并且队列是否为空，如果是，则子线程结束。
* 运行模型可参考如下图所示：



所有这些事情完成后，父子线程各自开始工作。当然，父子线程对队列的访问应当是互斥的，请使用线程同步互斥手段保证这一互斥。本课程设计中，所有的同步互斥工具**不推荐**使用System V IPC。

* 运行10次primer\_thr3，记录每次运行的时间，并计算平均运行时间。

1. **数据收集整理：**填写下面两个表，并要求课程设计报告中也需要包括上述两个表

表1 原始数据表

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 平均 |
| primer\_thr1 |  |  |  |  |  |  |  |  |  |  |  |
| primer\_thr2 |  |  |  |  |  |  |  |  |  |  |  |
| primer\_thr3 |  |  |  |  |  |  |  |  |  |  |  |

表中1,2，...，10分别代表第1次，...，第10次运行时间的测量值。

表2 性能比较表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| primer\_thr1平均运行时间 | primer\_thr2平均运行时间 | primer\_thr3平均运行时间 | CPU | 核心数（虚拟机） | 内存大小 |
|  |  |  |  |  |  |

分析你的性能比较表，在课程报告中回答以下问题：

1. 哪一个程序运行最快？为什么？
2. 增加核心数是否有助于提高程序的运行速度？
3. 增加内存的大小是否有助于提高程序的运行速度？
4. primer\_thr3中增加队列的大小是否有有助于提高程序的运行速度？