文件读写编程题目

myecho.c

- myecho.c的功能与系统echo程序相同
- 接受命令行参数，并将参数打印出来，例子如下：

```
$ ./myecho x
x
$ ./myecho a b c
a b c
```

mycat.c

- mycat.c的功能与系统cat程序相同
- mycat将指定的文件内容输出到屏幕，例子如下：
- 要求使用系统调用open/read/write/close实现

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
$ ./mycat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```

mycp.c

- mycp.c的功能与系统cp程序相同
- 将源文件复制到目标文件，例子如下：
- 要求使用系统调用open/read/write/close实现

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
$ ./mycp /etc/passwd passwd.bak
$ cat passwd.bak
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
```

多进程题目

mysys.c: 实现函数mysys，用于执行一个系统命令，要求如下

- mysys的功能与系统函数system相同，要求用进程管理相关系统调用自己实现一遍
- **使用fork/exec/wait系统调用实现mysys**
- 不能通过调用系统函数system实现mysys

- 测试程序

```
#include <stdio.h>

void mysys(char *command)
{
    实现该函数，该函数执行一条命令，并等待该命令执行结束
}

int main()
{
    printf("--------------------------------------------------\n");
    mysys("echo HELLO WORLD");
    printf("--------------------------------------------------\n");
    mysys("ls /");
    printf("--------------------------------------------------\n");
    return 0;
}
```

- 测试程序的输出结果

```
--------------------------------------------------
HELLO WORLD
--------------------------------------------------
bin    core  home            lib         mnt   root  snap  tmp  vmlinuz
boot   dev   initrd.img      lost+found  opt   run   srv   usr  vmlinuz.old
cdrom  etc   initrd.img.old  media       proc  sbin  sys   var
--------------------------------------------------
```

sh1.c

- 该程序读取用户输入的命令，调用函数mysys(上一个作业)执行用户的命令，示例如下

```
# 编译sh1.c
$ cc -o sh1 sh1.c

# 执行sh1
$ ./sh

# sh1打印提示符>，同时读取用户输入的命令echo，并执行输出结果
> echo a b c
a b c

# sh1打印提示符>，同时读取用户输入的命令cat，并执行输出结果
> cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

- 请考虑如何实现内置命令cd、pwd、exit

sh2.c: 实现shell程序，要求在第1版的基础上，添加如下功能

- 实现文件重定向

```
# 执行sh2
$ ./sh2

# 执行命令echo，并将输出保存到文件log中
> echo hello >log

# 打印cat命令的输出结果
> cat log
hello
```

sh3.c: 实现shell程序，要求在第2版的基础上，添加如下功能

- 实现管道
- 只要求连接两个命令，不要求连接多个命令
- 不要求同时处理管道和重定向

```
# 执行sh3
$ ./sh3

# 执行命令cat和wc，使用管道连接cat和wc
> cat /etc/passwd | wc -l
```

- 考虑如何实现管道和文件重定向，暂不做强制要求

```
$ cat input.txt
3
2
1
3
2
1
$ cat <input.txt | sort | uniq | cat >output.txt
$ cat output.txt
1
2
3
```

多线程题目

pi1.c: 使用2个线程根据莱布尼兹级数计算PI

- 莱布尼兹级数公式: 1 - 1/3 + 1/5 - 1/7 + 1/9 - ... = PI/4
- 主线程创建1个辅助线程
- 主线程计算级数的前半部分
- 辅助线程计算级数的后半部分
- 主线程等待辅助线程运行結束后,将前半部分和后半部分相加

pi2.c: 使用N个线程根据莱布尼兹级数计算PI

- 与上一题类似，但本题更加通用化，能适应N个核心
- 主线程创建N个辅助线程
- 每个辅助线程计算一部分任务，并将结果返回
- 主线程等待N个辅助线程运行结束，将所有辅助线程的结果累加
- 本题要求 1: 使用线程参数，消除程序中的代码重复
- 本题要求 2: 不能使用全局变量存储线程返回值

sort.c: 多线程排序

- 主线程创建两个辅助线程
- 辅助线程1使用选择排序算法对数组的前半部分排序
- 辅助线程2使用选择排序算法对数组的后半部分排序
- 主线程等待辅助线程运行結束后,使用归并排序算法归并子线程的计算结果
- 本题要求 1: 使用线程参数，消除程序中的代码重复

pc1.c: 使用条件变量解决生产者、计算者、消费者问题

- 系统中有3个线程：生产者、计算者、消费者
- 系统中有2个容量为4的缓冲区：buffer1、buffer2
- 生产者生产'a'、'b'、'c'、'd'、'e'、'f'、'g'、'h'八个字符，放入到buffer1
- 计算者从buffer1取出字符，将小写字符转换为大写字符，放入到buffer2
- 消费者从buffer2取出字符，将其打印到屏幕上

pc2.c: 使用信号量解决生产者、计算者、消费者问题

- 功能和前面的实验相同，使用信号量解决