Week11 Assignment

> 班级：212112 学号：21373339 姓名：周星达

阅读教材第七章，回答以下问题。

1. 简述信号量的作用，如何利用信号量实现同步和互斥？

> 信号量的实质是一个等待队列的计数器，可以用一个信号量用一个信号量来表示系统中某种资源的数量。 实现同步：进程获取临界资源之前，要先获取信号量资源；若无信号量资源，则该进程阻塞等待，进入等待队列。若有信号量资源，则对信号量进行P（-1）操作，再获取临界资源。当临界资源+1时，对应的信号量资源则执行V（+1）操作，然后唤醒在等待队列中等待获取临界资 源的进程。 实现互斥：一个进程获取了该临界资源之后，另一个进程无法再访问该临界资源。实现互斥采用一元信号量，即：该信号量的计数器，只能为0或1。一个进程要获取临界资源时，先获取对应的信号量资源；当无信号量资源时，则该进程阻塞等待，进入等待队列。当有信号量资源时，则对该信号量资源进行P（-1）操作，然后获取该临界资源。当该进程使用完临界资源时，将释放信号量资源（对信号量资源进行V（+1）操作），然后唤醒等待队列中的进程。

2.简述共享内存的作用和用法，共享内存为什么需要与信号量一起使用？

> 作用：共享内存是在两个正在运行的进程之间共享和传递数据的一种非常有效的方式。不同进程之间共享的内存通常安排为同一段物理内存。进程可以将同一段共享内存连接到它们自己的地址空间中。 原因：共享内存并未提供同步机制，我们需要通过信号量来同步对共享内存的访问。

3. 有以下代码：

```
int fd1,fd2,fd3,fd4;
fd1=open("a.txt",O_RDONLY);
fd2=open("b.txt",O_WRONLY);
fd3=dup(fd1);
fd4=dup2(fd2,0);
```

请问，最后fd1,fd2,fd3,fd4的值分别是多少？并解释原因。

> 3，4，5，0
>
> 0，1，2分别为标准输入、标准输出，标准错误。fd4为dup2指定的值为0

4. 请查阅资料简述进程间通信的 System V、POSIX 两种标准之间的差异性

> POSIX在无竞争条件下是不会陷入内核的，而SYSTEM V则是无论何时都要陷入内核，因此性能稍差。

5. 编写C程序实现如下功能：创建父子进程，父子进程之间通过管道进行通信，父进程向子进程发送字符串，子进程收到该字符串后，将该字符串的最后5个字符发送给父进程。

```c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
void lstFive(char *str)
{
    char *tmp = str;
    int j = 0;
    int length=strlen(tmp);
    for (int i = length - 5; i < length; i++)
    {
        str[j++] = tmp[i];
    }
    str[j] = '\0';
}
int main()
{
    int pipe1[2], pipe2[2]; // pipe1:parent to child     pipe2 : child to parent
    int pid;
    char str1[40] = "Miss Jiaran,I love you so much!";
    char str2[40];
    char readmessage[40];
    int returnstatus1 = pipe(pipe1);
    if (returnstatus1 == -1)
    {
        perror("Unable to create pipe 1 \n");
        return 1;
    }
    int returnstatus2 = pipe(pipe2);
    if (returnstatus2 == -1)
    {
        perror("Unable to create pipe 2 \n");
        return 1;
    }
    pid = fork();
    if (pid != 0)
    {
        close(pipe1[0]); // 关闭不需要的pipe1读取端
        close(pipe2[1]); // 关闭不需要的pipe2写入端
        printf("Parent Write to pipe 1:%s\n", str1);
        write(pipe1[1], str1, sizeof(str1));
        read(pipe2[0], readmessage, sizeof(readmessage));
        printf("Parent Read from pipe 2:%s\n", readmessage);
    }
    else
    {
        close(pipe1[1]); // 关闭不需要的pipe1写入端
        close(pipe2[0]); // 关闭不需要的pipe2读取端
        read(pipe1[0], str2, sizeof(str2));
        printf("In Child: Reading from pipe 1 Message is %s\n", str2);
        lstFive(str2);
        printf("In Child: Writing to pipe 2 Message is the last five bits%s\n",
str2);
        write(pipe2[1], str2, sizeof(str2));
```

```
    }
    return 0;
}
```