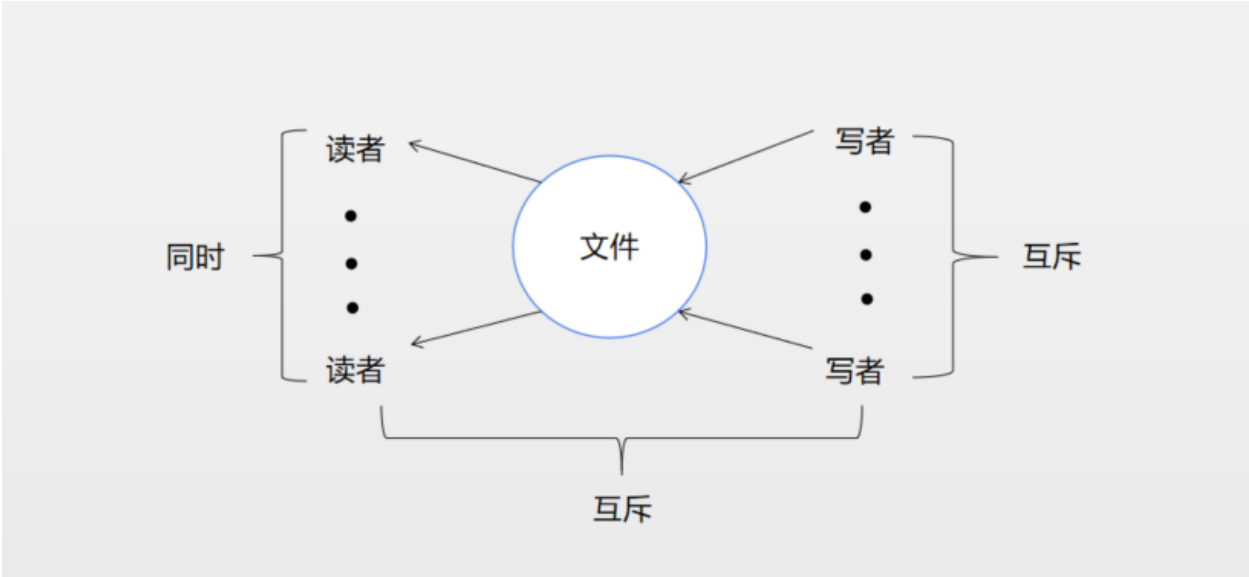
读者与写者的问题

- 例：有读写两组进程，共享一个文件。多个读者可同时访问文件，但多个写者不能同时访问文件，写者和读者也不能同时访问文件。



  - 特征：资源被谁占用，读者，写者；读者会形成一个读者团。
  - 分析：

    `写者` — `S` —文件是否被占用；

    > 读者团
    >
    > > 第一个读者：文件是否被占有
    > >
    > > 中间读者：只增加读者团人数
    > >
    > > 最后一个读者：释放文件

    互斥信号量`mutex` = 1；

    `S` = 1;

    `count`：读者团数量；
  - 伪代码

```
writer1()
{
    while(1)
    {
        P(S);
        写;
        V(S);
    }
}
```

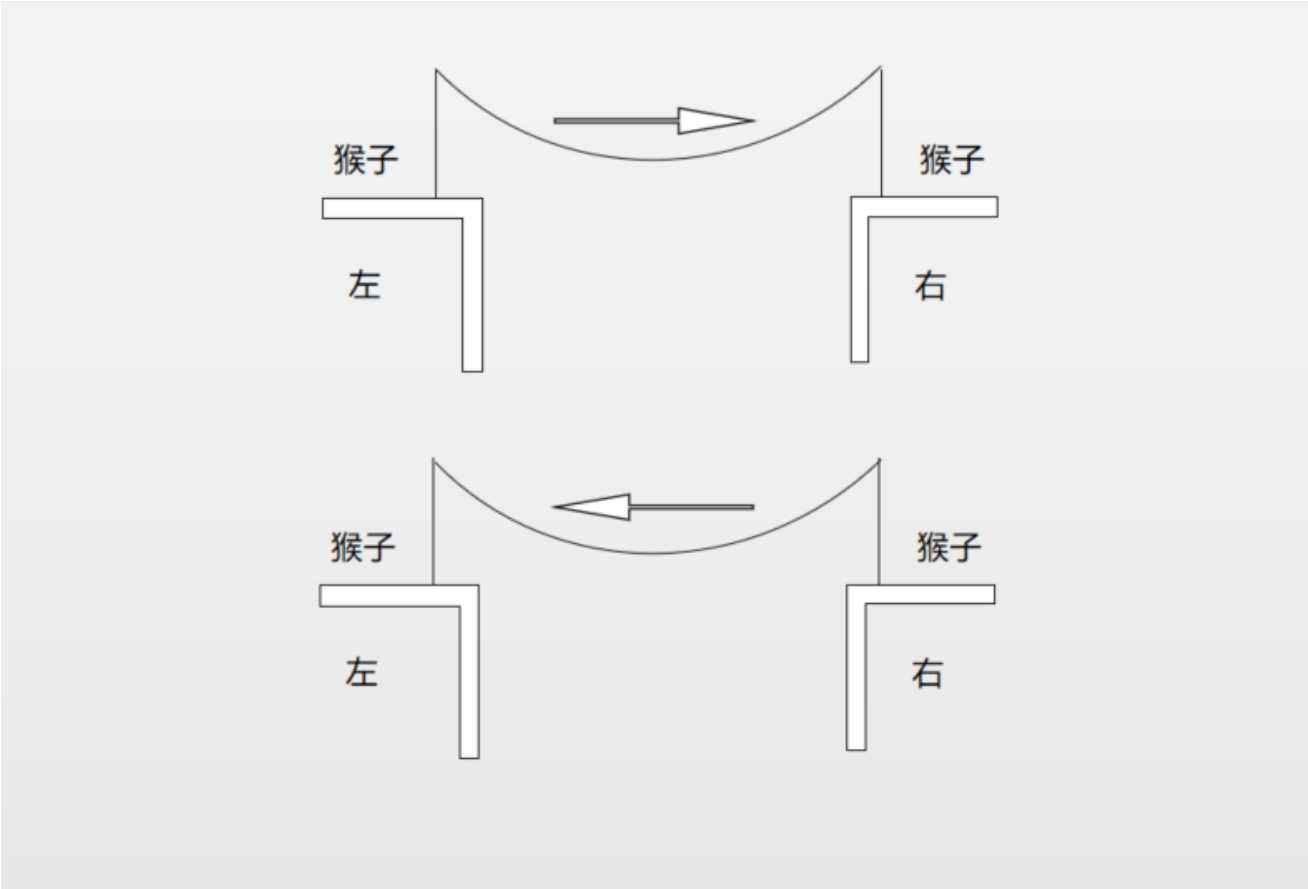
```
writer2()
{
    while(1)
    {
        P(S);
        写;
        V(S);
    }
}
```

```
reader1()
{
    while(1)
    {
        P(mutex);
        if(count == 0)//如果是第一个读者，判断文件是否被占有，中间读者不会去判断文件是否被占有
        {
            P(S);
        }
        count++;
        V(mutex);
        读;
        P(mutex);
        count--;
        if(count == 0)//如果是最后一个读者，释放文件
        {
            V(S);
        }
        V(mutex);
    }
}
```

```
reader2()
{
    while(1)
    {
        P(mutex);
        if(count == 0)
        {
            P(S);
        }
        count++;
        V(mutex);
        读;
        P(mutex);
        count--;
        if(count == 0)
        {
            V(S);
        }
```

```
        V(mutex);
    }
}
```

    - 上述代码也可以合并。
- 例：横跨峡谷有一根绳索，猴子通过绳索过峡谷。只要他们朝着相同的方向，同一时刻可以有多只猴子通过。但如果相反方向同时有猴子通过则会产生死锁(假设一直猴子无法从另一只钩子身上翻过去)。如果一只猴子想过峡谷，必须看是否有相反方向的猴子通过。请用P，V操作解决问题。



- 分析：只有读者

> 左猴团
>
> > 第一只：桥占用了吗?
> >
> > 中间猴子：只增加数量
> >
> > 最后一只：释放桥资源
>
> 右猴团
>
> > 第一只：桥占用了吗?
> >
> > 中间猴子：只增加数量
> >
> > 最后一只：释放桥资源

- 初始值：`count_L`：左边猴子数量

  `count_R`：右边猴子数量

互斥信号量：`S`

`mutex_L`：左边猴子信号量

`mutex_R`：右边猴子信号量

- 伪代码

```
left()
{
    while(1)
    {
        P(mutex_L);
        if(count_L == 0)
        {
            P(S);
        }
        count_L++;
        V(mutex_L);
        过桥;
        P(mutex_L);
        count_L--;
        if(count_L == 0)
        {
            V(S);
        }
        V(mutex_L);
    }
}
```

```
right()
{
    while(1)
    {
        P(mutex_R);
        if(count_R == 0)
        {
            P(S);
        }
        count_R++;
        V(mutex_R);
        过桥;
        P(mutex_R);
        count_R--;
        if(count_R == 0)
        {
            V(S);
        }
        V(mutex_R);
    }
}
```