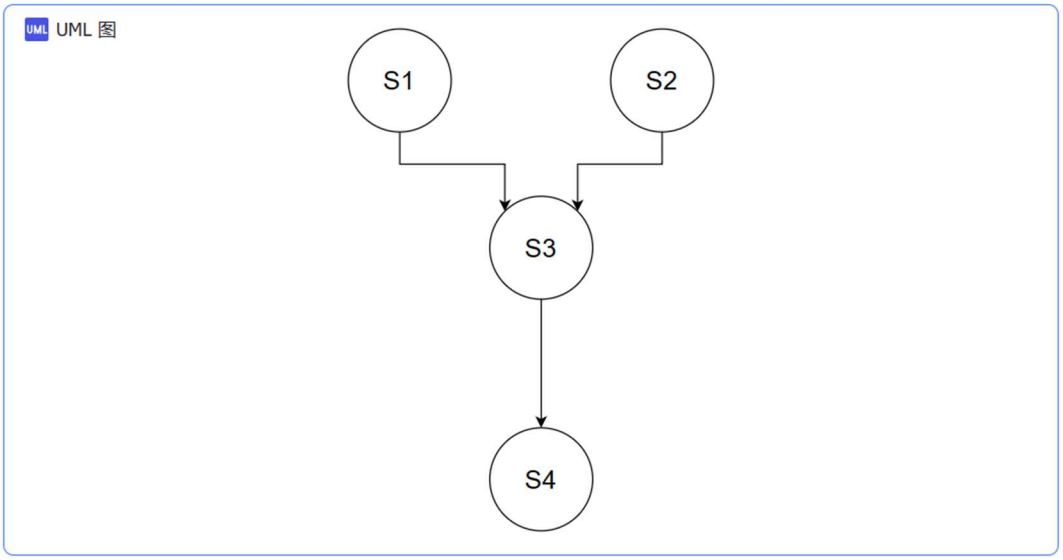
2.




5. 在操作系统中引入进程，是为了实现多个程序的并发执行。进程是程序在一个数据集合上运行的过程，它 是系统进行资源分配和调度的一个独立单位。引入进程概念可以使程序并发执行，并且可以对并发执行的程序加以描述和控制。这样就可以提高系统资源的利用率以及系统的处理能力。

8.（1）进程管理信息：
程序和数据的地址
进程同步和通信机制
资源清单
链接指针
（2）进程调度信息：
进程状态
进程优先级
进程调度所需的其它信息
事件，阻塞原因

9. 三种，线性方式、链接方式和索引方式

11. （1）就绪状态 → 执行状态 ：
当处理机空闲时，将从就绪队列中选择一个进程执行，该选择过程称为进程调度，或将处理机分派给一个进程，该进程状态从就绪转变为执行
（2）执行状态 → 就绪状态 ：
分时系统中，时间片用完，或优先级高的进程到来，将中断较低优先级进程的执行。进程从执行状态转变为就绪状态，等待下一次调度
（3）执行状态 → 阻塞状态 ：
执行进程需要等待某事件发生。通常，会因为进程需要的系统调用不能立即完成，如读文件、共享虚拟内存、等待 I/O 操作、等待另一进程与之通信等事件而阻塞

（4）阻塞状态 → 就绪状态 ：

当阻塞进程等待的事件发生，就转换为就绪状态。进入就绪队列排队，等待被调度执行。

13. 在进行进程切换时，所要保存的处理机状态信息有:

(1)进程当前暂存信息;

(2)下一指令地址信息;

(3)进程状态信息;

(4)过程和系统调用参数及调用地址信息

一. 生产者消费者

如果没有 signal(full)满缓存区数量+1 的话，在生产者执行完毕后，full 依然为 0，执行消费者的时候，因为 full 为 0，判断为现在缓冲池没有满缓存区，而进入等待状态。

如果缺少 signal(empty)，空缓冲区+1 的话，因为 empty 初始值为 n，执行 n 次生产者之后 empty 为 0，之后会让生产者一直处于等待状态

二. 哲学家就餐

```
semaphore chopstick[5] = {1, 1, 1, 1, 1};

void philosopher(int i) {
    while (true) {
        think();
        wait(chopstick[i]); // 拿起左边的筷子
        if (chopstick[(i + 1) % 5].value == 0) { // 检查右边的筷子是否可用
            signal(chopstick[i]); // 放下左边的筷子
            continue;
        }
        wait(chopstick[(i + 1) % 5]); // 拿起右边的筷子
        eat();
        signal(chopstick[(i + 1) % 5]); // 放下右边的筷子
        signal(chopstick[i]); // 放下左边的筷子
    }
}
```