操作系统上的程序

核心观点：程序是状态机

- 状态 = stack frame 的列表 (每个 frame 有 PC) + 全局变量
- 初始状态 = main(argc, argv), 全局变量初始化
- 迁移 = 执行 top stack frame PC 的语句; PC++
  - 函数调用 = push frame (frame.PC = 入口)
  - 函数返回 = pop frame

> 每一次函数调用都会产生一个新的栈
>
> 每个栈里面存有参数，以及**pc**
>
> 函数返回就是把顶上的栈删掉

程序=计算+syscall

syscall：把进程的所有东西交还给操作系统

编译器：源代码 **S** (状态机) → 二进制代码 **C** (状态机)

$$compile(C) = compile(S)$$

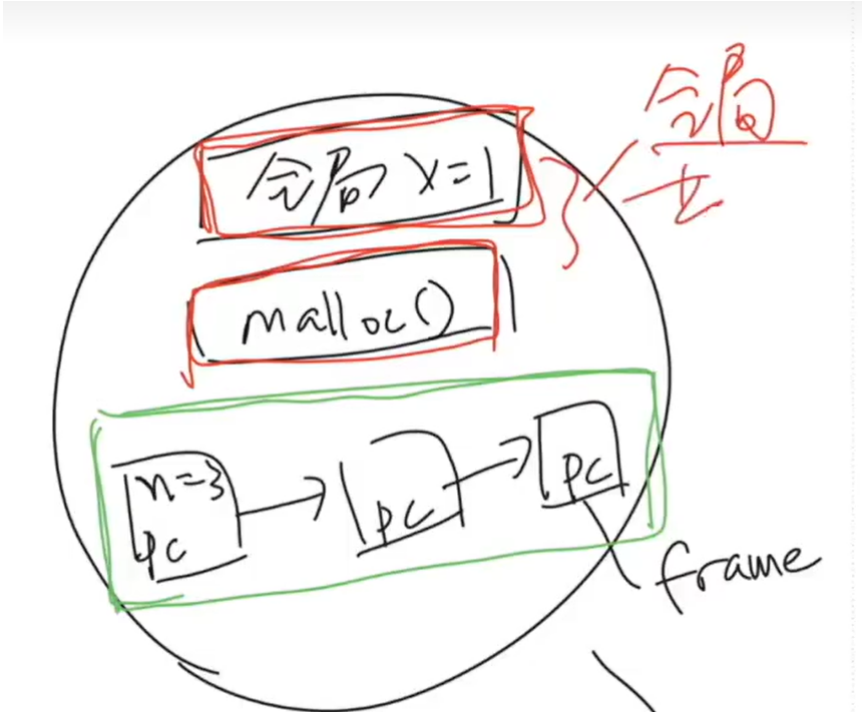
多处理器编程

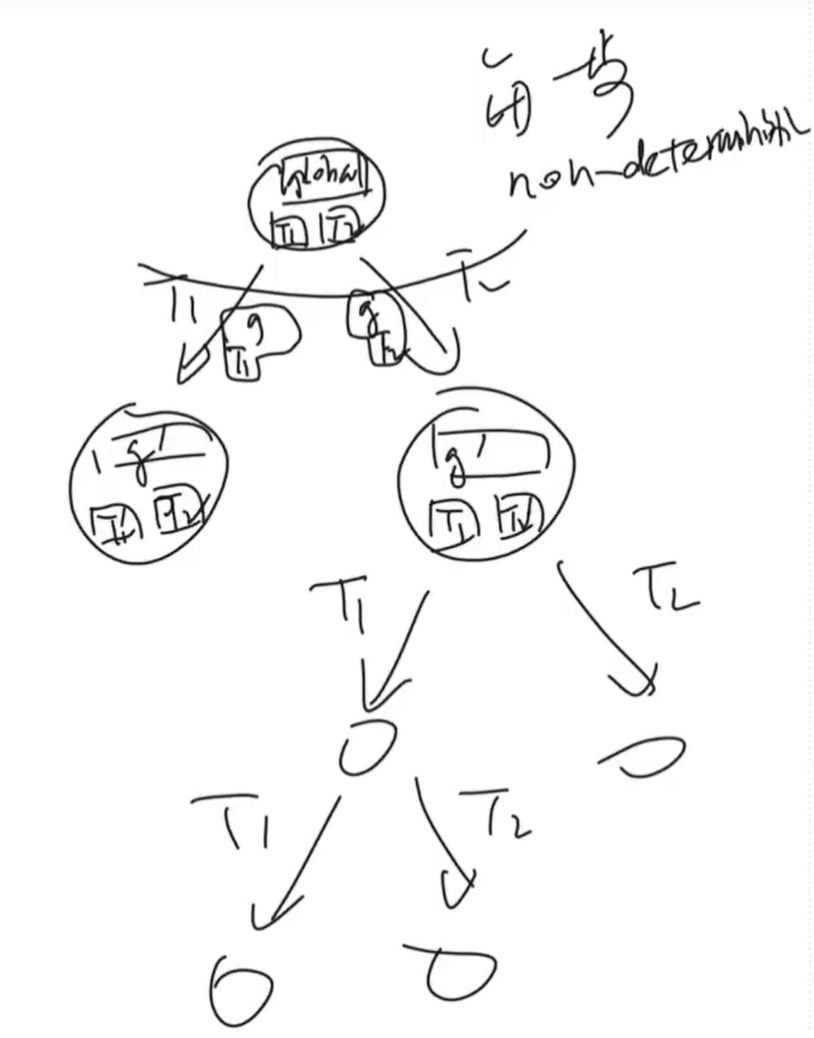
并发（Concurrency)

定义：一个程序，算法或者问题的分成多个部分，乱序或部分按序执行，不影响结果

基本单位：线程

状态机视角：





并发拥有多个“局部”进程，每次随机选择一个线程执行

```
asm volatile ("Instruction List" : Output : Input : Clobber/Modify);
//  内嵌汇编
```

在代码中插入 “优化不能穿越” 的 barrier

- ```
  asm volatile ("" ::: "memory");
  ```

  Barrier 的含义是 “可以读写任何内存”
- 使用 volatile 变量，保持 C 语义和汇编语义一致

  ```
  extern int volatile done;
  while (!done) ;
  // 否则会被优化为 if (!done) while (1);
  ```

单个处理器把汇编代码 (用电路) “编译” 成更小的 μops

在保证编译正确性的前提下，尽可能多地发射指令，乱序执行，按序提交
  ```