Tutorat : concurrent systeme

Question

- transaction 中的读写冲突， ACID 属性，事务的【依赖图】
- concurrent control

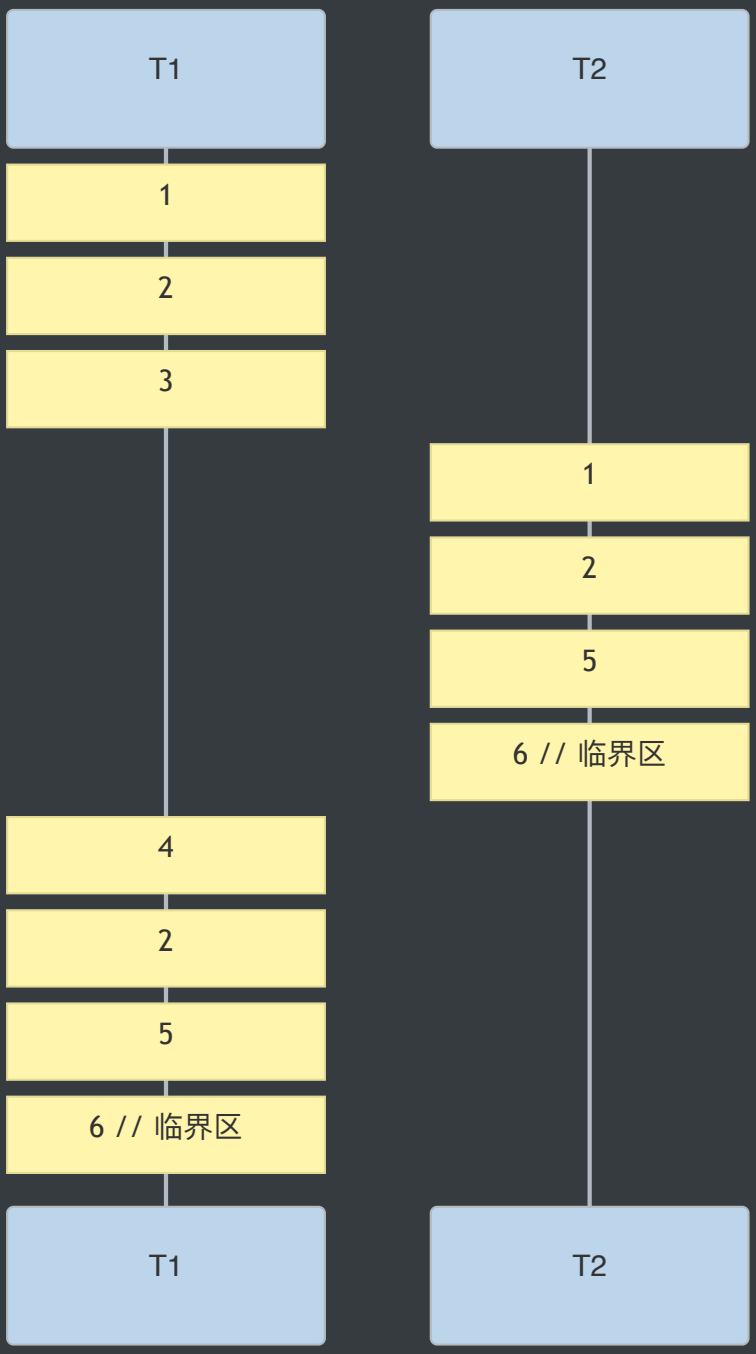
```
n := y +1;
t1.write(t1.read(y));
```

- 多进程的内存共享，并发，互斥

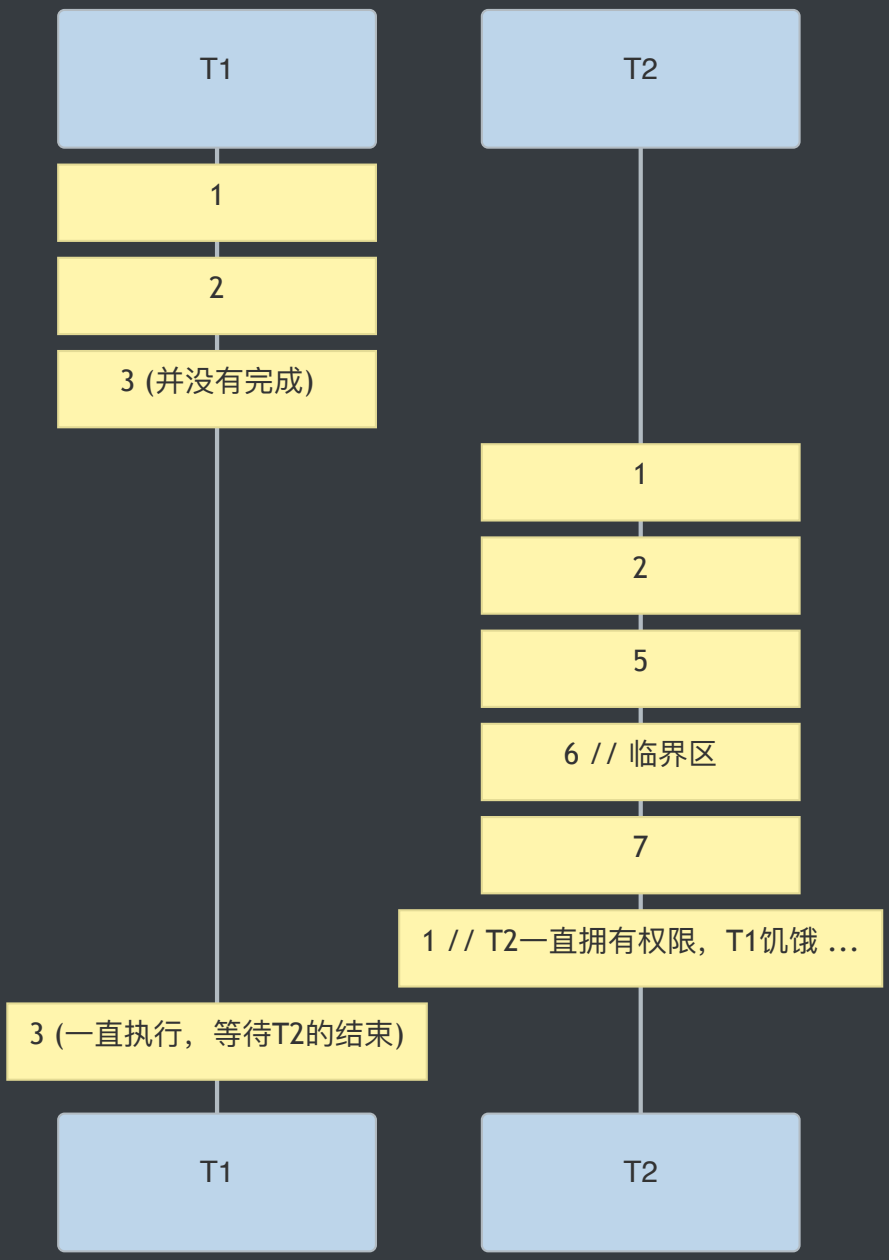
Exercise 1. Concurrent execution

```
demande[i] = true;  // 1
while (tour != i) { // 2
  while (demand[j]) { // 3
    // 阻塞;
  }
  tour <- i // 4
} // 5
// SECTION CRITIQUE // 6
demande[i] = false; // 7
```

1. 描述：算法并不会确保互斥地访问临界区。



2. 描述：算法会导致饥饿



Exercise 2.

同步问题：

	marins	personnals
N	4	2
V	3	3
disponible	3	3
	0	1

```
Semphore PE[category] = new Semphore [0,0];
mutex := 1 // 互斥信号量
```

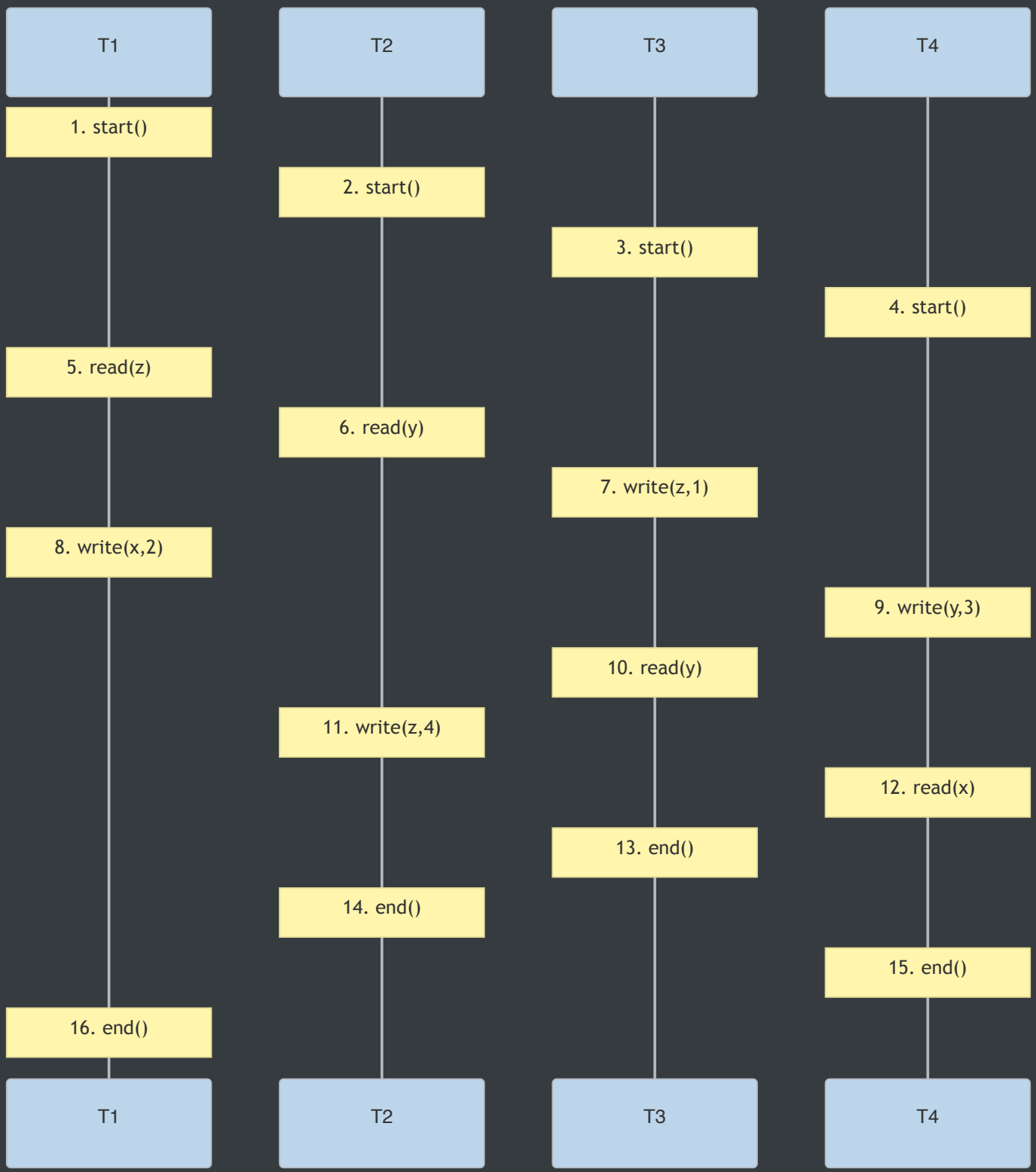
```
lever_l_ancre (t){ // reviel de NB[t,M] marin et NB[t,P]
  for(int i = 1; i < (NB[t,M] + NB[t,P]); i++){
    monter.down();
  }
}
```

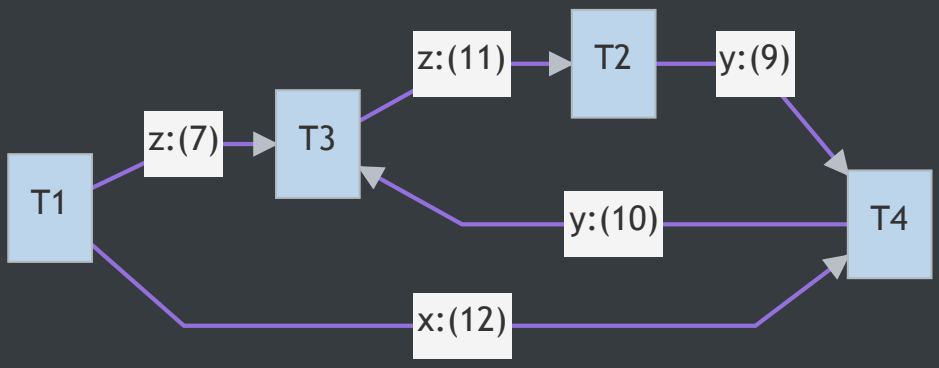
```
embarquer (c) {
  PE(c).down(); // marin可以登船的信号量
  monter.up(); //
}
```

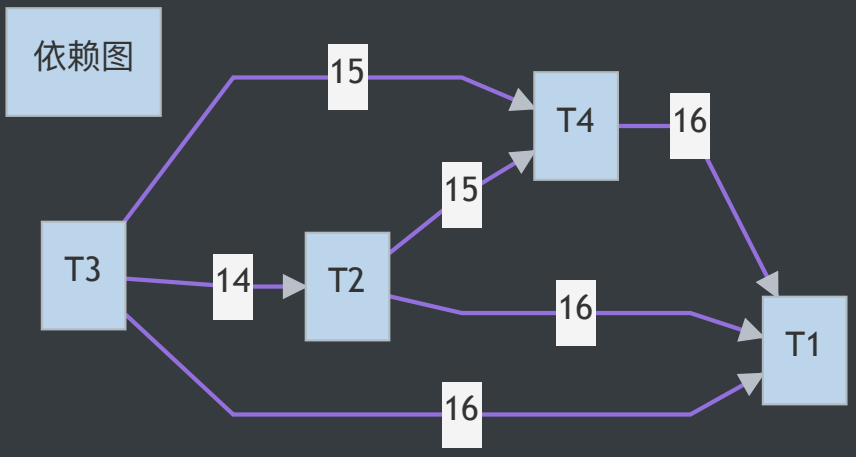
```
appeler_equipage (t) {
  mutex.down();
  if (dispo[-] >= NB[t] ){
    for(int i = 1; i < (NB[t,M]; i++){
      PE.[M].up();
    }
  } else {
    mutex.up(); // why?
    PLA[t].down(); // Peur_Lancer_lAncre
  }
}
```

```
se_presenter(c) {
  mutex.up();
  dispo[c]++;
  if (dispo[-] >= NB[t]){
    PLA[+].down();
  }
}
```

Exercise 3. Transaction


T1
T2
T3
T4
1. start()
2. start()
3. start()
4. start()
5. read(z)
6. read(y)
7. write(z,1)
8. write(x,2)
9. write(y,3)
10. read(y)
11. write(z,4)
12. read(x)
13. end()
14. end()
15. end()
16. end()
T1
T2
T3
T4


T1
z:(7)
T3
z:(11)
T2
y:(9)
T4
y:(10)
x:(12)


依赖图
15
T4
16
15
T3
14
T2
16
T1
16