



## Process คือ โปรแกรมที่กำลัง被执行 .

### ผู้สั่งการของ Process

1. ลำดับความสำคัญของ Process (Priority)

2. อำนาจหน้าที่ของ Process (Authority)

### Process Model

process A, B, C, D ฯลฯ ลับกันทำงาน ก็ต้องแบ่งส่วนๆ ให้ CPU กับ 1 process หรือ 1 Quantum โดยที่  
หักไว้ในต่อจับเวลา ที่เรียกว่า clock ต่อครุ 1 Quantum process A จะต้องป้อน CPU โดยที่มี clock interrupt  
เมื่อเกิด interrupt (มาดูเวลา) CPU จะตัดปลดปล่อย process นั้นๆ (เรียกว่า preempt หรือ)

เมื่อ process A ขาดเวลาการทำงาน OS จะเรียกคืนบริการ CPU และต่อว่า กด interrupt ตัวเอง ว่าต้อง

context-switch = ms save หมายความว่าต้องรีเซ็ต context ของ process ที่เขียนไว้ไปแล้วโดยวิธี และต่อไปต้องรีเซ็ต context ใหม่  
ที่จะทำงาน save process เดิม ผ่านมัน back to process ใหม่ ไม่ overhead มากนัก หมายความว่า context switch ชุบๆ กัน

### Process Creation (EFO)

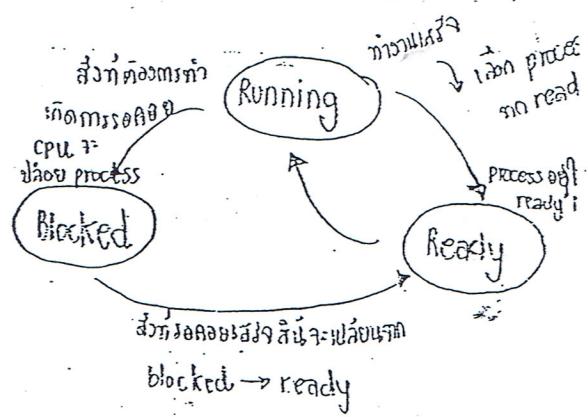
- process ถูกเรียกใช้งานเริ่มต้น
- process เจดจากภายนอก (เช่น system call)
- ผู้ใช้งานเรียกใช้งาน
- งานที่ถูกสร้างขึ้นและเริ่มต้น

### Process Termination (สิ้นสุด EFO)

- สัญญาณการทำงาน ปิดโปรแกรมตามปกติ Error exit
- เมื่อต้องผิดพลาดที่พบได้คร่าวๆ check เผื่อนผู้ใช้ Prog.
- Fatal error ต้องผิดพลาดร้ายแรง
- Killed by another process

### Process States (สถานะของ Process)

- Running = process ใช้อำนาจ CPU อยู่ท่ามกลาง
- Blocked = สถานะที่ process กำลังอยู่บนตู้กรอบมีห้องชั่วๆ ที่เกิดขึ้น  
ก่อนที่จะปลดปล่อย CPU ลงมาต่อไป
- Ready = ผู้ใช้งานเรียกใช้งานแล้วตัด CPU และ Running States ถูก  
preempt state หรือ switch



### ผู้ใช้งาน Process ไม่สามารถรักษาได้

- I/O-bound process คือ Process ที่ไม่สามารถใช้ประโยชน์ของ CPU ได้มาก (ต้องรอ I/O)
- CPU-bound process คือ process ที่ไม่สามารถใช้ประโยชน์ของ I/O ได้มาก (ต้องรอ CPU)

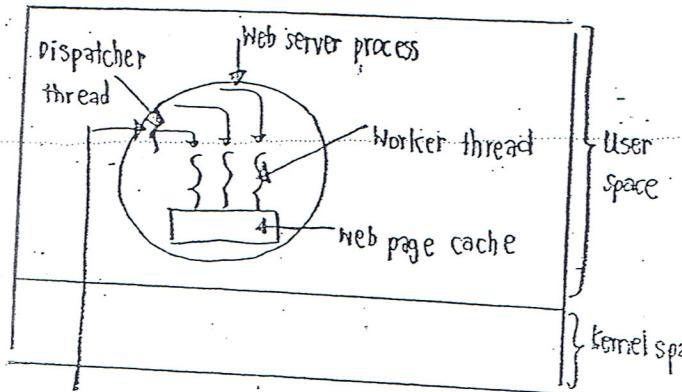
### throughput

คือ ขนาดที่แล้วเสร็จต้องนานเท่าไร ค่าสัมบัติไปทางเดียว ฉะนั้นค่า throughput มากกว่า 1 process เดียว

“...ເສັ້ນການໃຫຍ່ປະມາດພລ ຕົນາກຕຽກກໍາງານ ແລ້ວໄຫຍ່ປະມາດພລ ສອງຕົດຫຼື່ວໂດຍໃຫຍ່ກຳນົດ=Blocked

បានការងារប្រចាំរយៈពេល ដើម្បីបង្កើត Multi-threaded Process (សម្រាប់ប្រភេទ = thread) និងការបង្កើតការងារកំណត់ថ្ងៃ (Scheduled Process)

## Multithreaded Web server



## Dispatcher thread

1. ក្នុង request នៃ client ត្រូវការ request និង save ទៅ buffer
  2. Worker thread នឹងបញ្ចូនការណា request ទៅ Dispatcher ដើម្បីរាយការក្នុង page នៃ Cache
  3. ឯកត្រា នៃការការពារនេះ នឹងត្រួតពិនិត្យនៅលើ disk

ກວດສອບຕາມມາດ thread ສົດສາງໆນັ້ນ shared memory ໂດຍ ຕ່າງໆທີ່ກ່າວຂອງພວກເຮົາ ແລ້ວຢູ່ກ່າວຂອງຈະຫຸ້າສຳຕັ້ງກົງລັບໄປໃຈນັ້ນ

Java: threading multithreaded

1. ความต้องการในคราวเดียว
  2. resource sharing. ผู้คนสามารถ share ร่วมกันใน shared address space ที่มีอยู่หนึ่งเดียว
  3. economy ในการเปลี่ยน process เนื่องจาก switch process จึงต้องทำอย่าง save to สำหรับการ多线程多线程
  4. ไม่ใช้ชุดเดียว แต่ใช้ชุดเดียวกันในคราวเดียว process switch
  5. ไม่ใช้ชุดเดียว multiple processor ได้

## Java Multithreading

## 1. Many-to-one (User-level Threads)

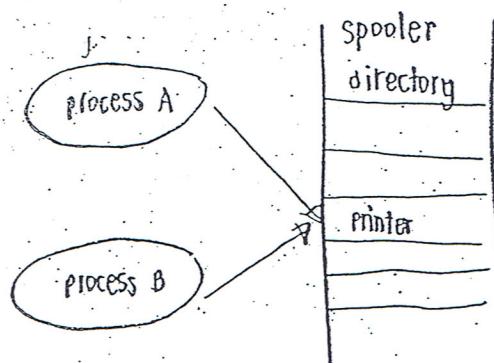
- ក្នុងទេរង់ នឹង thread ដែលរួចរាល់ពី OS  
និង នឹងក្រុមហ៊ុនសម្រេចអាជីវកម្ម
  - សូចសម្រាប់ការឲ្យ 1 process ទិន្នន័យ 1 CPU និង  
OS ត្រូវបានអាចនាំរាយ thread.
  - ក្នុងការឲ្យ thread និងក្រោម នាមរាល់មួយ  
និងក្នុងទេរង់ នឹង thread ទាំងអស់នៃ thread library  
និង thread block → process, block ប្រព័ន្ធ

## 2. One-to-one (kernel-level Threads)

- ມີຮອດການກ່າວຍໃຈ thread ທີ່ຕ້ອງໄປ Kernel ຕ້ອນສົ່ງ  
ກໍາໄຟ overhead ຖຸ້ງ ທ່ານໄດ້ຕ້ອງກ່າວຍ ກັບ OS ສໍາຜົນຫຼຸດນີ້  
ເກົ່ານີ້
  - ຕໍ່ thread : block ສະໜອງ switch ໃປຢາ thread ດັ່ງ  
ໃນ process ເຊິ່ງກັນ ທ່ານໄດ້ ຕໍ່ thread ໃນ block  
process ໄສຕ່ອງ block
  - ຕ້າມ ນລກຍ່າງ CPU ທ່ານໄດ້ເກີດ pseudo-parallism ບຸ້ນ

ມະກີ process ເຊື້ອ shared resource ແລ້ວຜູ້ອໍານວຍທີ່ເກີດຮຽນ ໃປ່ງຍອດຝັບຄ້າລັບໃນການປະກາດຜູ້

กระบวนการ = process ที่เกี่ยวของ

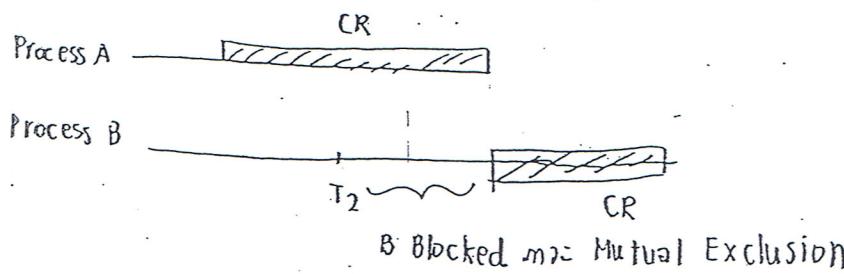


▪ Pro A, B សិរីក្រឡាតាំង printer តែមួយគ្នា និងភ័ព្យលាការកិច្ចសិរីភ្លាម  
នាមតុកំពង់នៃការរាយការណ៍ គឺតុកំពង់នៃការ [spooling] ដែលនឹងកើតឡើង  
ឡើងតាមរបៀបនេះ (write file នៃ local disk នៃការពិនិត្យ update ក្នុង )

## 2. Mutual Exclusion

- មិនការកំណត់បាន Process ដែលទទួលទី resource នៅពេលនេះ Process នៅលាស់ទៀតទៅទី resource នៅលើ  
ប្រវត្តិថាទីដែរឱ្យបានដោយការរាយការណ៍. Process A នូវយុទ្ធសាស្ត្រ resource នៅលើ ឬការណា Critical Region (CR)  
ឬការងារមិនបាន Mutual Exclusion

1. ຕ່ອງໄຟເລື່ອ process 2, process ອໍານີ້ໃຈ CR ມີຄວາມ
  2. ຕ່ອງໄຟເລື່ອສາມາຕົ້ງກຸາແລະ ບ້ອຈາກດັກເບິຍກັບຄວາມປັບປຸງ ຖລ = 1 ນ. CPU ລາກທີ່ປັບປຸງ
  3. ຕ້ອງໄຟເລື່ອ process 1 ທີ່ ສະໜອດ CR ທີ່ block ມີກ່າວກ່າວຂອງ process ຢື່ນ
  4. ຕ້ອງໄຟເລື່ອ process 2 ທີ່ກ່າວມາໃຫຍ່ CR ຕາມຄວາມ



## Mutual Exclusion with busy waiting

- សម្រេចការងារ CR តាមលក្ខណនាលើ 2 process នៅក្នុងការងារប្រចាំថ្ងៃ និងការងារក្នុងការងារប្រចាំថ្ងៃ

1. clock interrupt : ของ CPU ว่า วน ครอนวัล \_QUantum ปลด
  2. I/O interrupt : ของ CPU ว่า process ที่ถูก block วนกาวันกับ I/O เหรอ แล้ว CPU กับจะรีคิว process นี้ให้เสร็จวะ และ นำมุ่งมาเป็น ready

Ակտորների միջև Mutual Exclusion

1. Software
  2. Hardware
  3. OS / Programming Languages . Support

## 1. Disable interrupt

- ออกเด้ง interrupt
- CPU ห้ามที่จะรับข้อมูลใดๆ ไม่ได้ interrupt
- Process ที่อยู่ใน CR ห้าม execute งานต่อไปเกิด Race condition

- ไม่คำนึง Disable interrupt ต้องรอน CR:

Able interrupt ไม่ห้ามก่อตัว CR

### ข้อดี

- ไม่ต้องเปลี่ยน context ให้กลับมา Kernel mode

- ตัวเดียว Able interrupt ทำให้ process ที่ใช้ร่วมกันเดียวกัน

- คำนึง Disable interrupt สามารถ CPU หล่อเลี้ยง

## 2. Lock Variable

- If Software สามารถจัดการได้ Race condition
- ไม่หันมาก Race Condition ผ่านทางปุ่ม shared ของลูก

## 3. Strict Alternation

- ใช้การสลับturn เท่า CR ทั้ง Process 2 Process (0,1,0,1)
- กำหนดตัวแปรที่ทั้ง 4 บิต เก็บผลการสลับturn เท่า CR ทั้ง Process หักกันๆ จนกว่าจะเท่ากัน

### ข้อดี

- ใช้ได้เฉพาะ 2 process เท่านั้น

- เกิด busy waiting ซึ่งค้างใน CPU อญ แต่ไม่เกิดปะทะกัน

- ถ้าเราลากไว้ในกราฟผ่านๆ ผลลัพธ์ process ทั้ง 2 ต่อตัวมาก ทำให้ process ที่ไม่ใช้งานทำงานของ process ที่ใช้

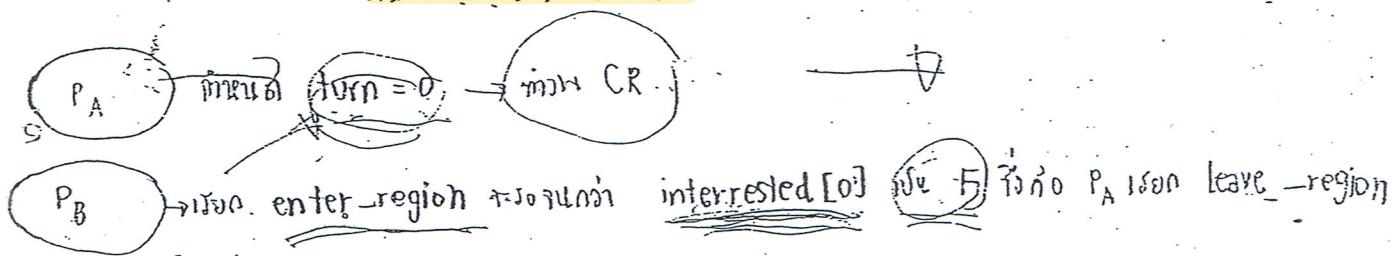
## Peterson's Solution

- ใช้ shared ตัวแปร (turn) และ 2 procedure คือ

1. enter\_region = ทำการเก็บturn ของตัวเอง แล้วดูดูว่าturn ของตัวอื่น

2. leave\_region = แสดงตัวเองเป็นเจ้าของ turn แล้ว leave\_region ให้ตัวเองเป็นเจ้าของturn แล้ว

ขึ้นใน Process ที่ไม่ใช้ตัวเอง



- ร่วมกันป้องกัน lock variable โดยที่ Test set lock ในรูปแบบเดียวกัน
  - ต่อไปนี้จะอธิบายว่า มาก่อนหน้า register และ set ที่ไหน ก็ตามที่ = 1
  - กรณีที่ CPU ไม่ได้เป็นผู้ใช้งานตัวที่ 0 ความจริงที่ processor จัดการกับ锁ในรูปแบบเดียวกันที่ต้องหันไปที่จุดก่อตัวของตัวเองแล้ว
  - CPU ที่ TSL ไม่สามารถเข้าสู่ CPU อื่น หรือ ค่าต่อไปนี้
  - ค่าที่ TSL จะต้อง share ล็อก
  - ตัวอย่าง lock = 0 process ถูกตั้งค่า = 1
  - ค่าที่ TSL = 0 แต่หู 1 ที่จะห้ามตัวนั้นเข้าไปใช้锁 = ค่าต่อไปนี้  
 $MOVE = \text{การนำมาร์กพอยต์ไปที่ } 0$
  - ANSI-Process ต้องร่างเข้าไป CR\_1 = 1 ตอน enter\_region หลัง copy ต่อมาจะ Lock ที่ Register  
และ set ที่ Lock = 1

## Sleep and Wakeup

- sleep នឹង system call នៅ block និង process ឱ្យបានរាយការការងារ ទាំងអស់ នៃ process ដើម្បី
  - Wakeup កំណត់ឡើង process ការងារវិញ
  - Sleep និង Wakeup ការងារនេះត្រូវបានផ្តល់បញ្ជី ចំណាំការងារនៃ parameter និង address ដើម្បី ការស្នើសុំការងារវិញ

Program Producer-Consumer with monitor Sleep-Wakeup (fatal race condition (ก่อไข้ใน sleep-wake))

- producer = ເພີ້ມໂຄງການໃນ buffer ໂດຍເຫັນອະນຸຍາດການເລືອດໃນ buffer ໂດຍ  $\Rightarrow$  sleep ໂດຍ  $\Rightarrow$  Sleep ໂດຍ  $\Rightarrow$
  - Wakeup ມີ consumer ທີ່ສູ່ມີການເລືອດໃນ buffer ແລ້ວ  $\Rightarrow$  ພົບການເລືອດໃນ buffer
  - consumer = ທີ່ຮັບການເລືອດໃນ buffer  $\Rightarrow$  sleep ໂດຍເຫັນອະນຸຍາດການເລືອດໃນ buffer ໂດຍ  $\Rightarrow$  Wakeup  $\Rightarrow$  producer

```

graph LR
    sleep --> Running
    Running --> Blocked
    Blocked --> Ready
    Ready --> wakeup

```

ກວດສອບ Producer ຂະໜາ ຖືກາງ ສົງລັບ count = N ພຽບໃໝ່ delta = N ຮັງ Sleep ກ່ອດປູນຈຳກຳ ນຳກຳກາມຜົດໄກ

ເນື້ອງກໍາລົງປະກາດ. ອັນດີເມືອນບັນຍາ. Wake up

ກ່ອນມີຄວາມເປົ້າໃຫຍ້ 0 ລາຍລະອຽດເກົ່າງໆ Wakeup - ດີຈາກ ລາຍລະອຽດ Wakeup

- Semaphore ແມ່ປັບປຸງຕາມເລີຍກໍາໃນ process ສະໜັກການດີ access Semaphore ໄດ້ຈະກຳໄວ

ເສົ່າມະນຸຍາມໃຫຍ້ block ກໍາໄຟຮ້າງໄຕຣກໍາປູ້ທາງ synchronize ນີ້ແລ້ວເລີ່ມຕົວເລີ່ມຕົວ race condition

Mutex

- 例4 Semaphore օգնութեան հիմքում Mutual Exclusion նւ resource ն պահպան

- **ANSWER:**  $\text{unlocked} = 0$ ,  $\text{lock} = \text{large}$

- កំពុងការធ្វើ, CR ទទួលឯក mutex\_bck និង Mutex នូវការដែល unlock នាមឈរក្នុងតាមរយៈការបញ្ចូន CR ត្រូវ

- මෙයි Locked process සඳහා block කළ තුළු ප්‍රෝසේസ් නිස් නිස් නිස් CR සඳහා නිස් නිස් නිස් නිස් නිස් mutex-unlocked

\* mutex-locked នៅពេល enter-region និង enter-region ជាន់ CR ត្រូវបាន busy waiting

នៅក្នុងការបញ្ចូលឈើក្នុង mutex-locked នៃការសម្រាប់ការបញ្ចូលឈើក្នុង CALL thread-yield នៅក្នុង CPU តែនៅ thread ឬឱ្យការ

Monitor

- ក្នុងពិធីរាយ Down 2 ក្នុងរាយទាំងនេះ ដើម្បីជួយ Producer ការរំលែកសំណង់

- Muñeca ទេសការិយេ empty និងអ្នកទេសការិយេស្ថាប់រាយការ

- ຕາ buffer ເພີ້ນ ແລ້ວ Producer ຈະ ດັບອຸປະກອດ ໂດຍ Mutex ຈະໃຫຍ່ກຳ

- (ii) Consumer માણસુધી એક્સેસ બફર રૂપી રીતે કરી શકતું હૈ.

- Process 2가 block 상태로 N을 만들었을 때

\* Monitor է պահպան մօթելի ըստ Dead lock

\* Monitor នៃ ក្បាល់ procedure , ចំណាំ , ក្រោមធនធានមុនការណ៍បានដូចណាន់ដែល packet និងខ្សោយការណ៍នេះ

- Process នៃការគាំទ្រនូវផលិតផលផ្លូវការ

Process និង Monitor នៅក្នុង shared memory ទៅអនុវត្តន៍ និងបានស្វែងរក shared memory នៅក្នុង message passing

\* Mr. Send, receives msg at producer & consumer wills synchronize msg b/w.

1. Batch
2. Interactive
3. Real time.

### Scheduling in Batch System

- 1) First-come first served (FCFS): តើ អាក់សំបុរីនេះ មែន សំរាប់ long term និង ខ្លួនឯកសារ ភាព  
  - មួយចាប់ Waiting time គឺ ពីកូលិនិមួយៗ Arrival time ទៅពីលិខិត Process ចូល ឱ្យ
  - ភាព ~~Convo~~ effect តើ អីដឹង Process ចូលសម្រាប់ការបង្កើត និង ដឹង Process ដូច
  - ឯកសារទាំងអស់ តើ សំរាប់ និង បង្កើត

### 2) Shortest job first (SJF)

- ឯកសារ នៃ Burst time និង ពីកូលិនិមួយៗ Arrival time ឱ្យ
- ភាព Waiting time តារាង និង Average turnaround time ឱ្យ
- អំពី ទៅ Long-time scheduling
- ឯកសារទាំងអស់ និង (nonpreemptive)
- ស្របតាម ការបង្កើត និង Process ចូល និង ការបង្កើត គិតជាដំឡើង

### 3) Shortest-remaining-time-first (SRTF)

- នៅ SJF ការបង្កើត និង ការបង្កើត (preemptive)



### Scheduling in interactive Systems

#### 1) Round-robin scheduling (RR)

- ប្រព័ន្ធគារបង្កើត quantum តាមបន្ទាត់ជាប្រព័ន្ធដែល និង
- ពី quantum និង រាយសង្គម និង Context switch, throughput ឱ្យ
- ពីនូវ 1 quantum ឬ process ទូទៅ និង ក្នុងការបង្កើត និង ឯកសារ queue និង ឯកសារ queue
- ចំណាំ preemptive

#### 2) Priority Scheduling

- នៅ Priority គូរការពីកូលិនិមួយៗ និង FCFS
- និង ពីកូលិនិមួយៗ និង nonpreemptive
- នាយកសារ ឥវឌីនី Stratification នៃ process នៃ priority និង សំណង់ និង ផ្ទេរ ភាព កំណែ
  - 1) នៅ Priority និង
  - 2) នៅលើ "Aging" និង Upgrade និង Priority

## The Dining Philosophers Problem

- ห้องอาหาร 5 โต๊ะ มีคนนั่งกินอาหารรอบโต๊ะ 5 ตัว
- แต่ละโต๊ะ ก็ต้องใช้ช้อน 2 กรร (ซ้าย-ขวา)
- เมื่อ กินเสร็จ ไม่สามารถกลับไปกินปูต่อได้ ต้องรอ
- ถ้าไม่ปักหมุดที่ตัว กิน ก่อน กินปู ก็จะเกิด Deadlock
- ถ้าปักหมุด ร้าบ-ร้าว สลับกัน กิน ทุกคน ก็ไม่ได้กิน เนื่องจาก Strayation

\*Process ที่ต้องรับ resource ไม่ครบ จะไม่กินต่อไป จนกว่า resource จะครบ

การให้สิทธิ์ resource hold and wait คือ ต้องการถือ resource ก็ต้องใช้ก่อน ไม่ว่าจะมีคนอื่นอยู่ หรือไม่

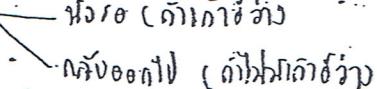
ถ้าให้ลง mutex ก็ไม่เกิด deadlock

- ถ้า ก่อนเข้ามานั่ง ก็ Down ที่ mutex
- พอเขียนต่อ ก็ Up

## (2) The Readers and Writers Problem

- กรณี Reader database ที่ต้องการเข้ามา อ่าน ไม่ต้อง block แต่ต้อง block ตัวผู้อ่าน
- กรณี Writer db ต้องการเข้ามา อ่าน ต้อง block ตัวผู้อ่าน
- กรณี Writer กด lock ที่ database ต้องการเข้ามา อ่าน ต้อง block ตัวผู้อ่าน
- แต่ reader คนที่ 2 ที่ต้องเข้ามา อ่าน ก็ต้อง block ตัวผู้อ่าน ก่อน ไม่ได้ใช้ semaphore mutex แต่ใช้ db update db ที่หักกันก็ต้องการเข้ามา อ่าน ก็ต้อง block หักกัน

## (3) The Sleeping Barber Problem

- ผู้ตัดผม 1 คน , ผู้ตัดผม 1 ตัว กลางคืนร้อนร้อนมาก
- สนใจร้านค้า ที่ว่างเปล่า
- ไม่ยอมร้านค้า กินแยกร้านค้า ไม่เป็นไปได้
- ล้างน้ำหัวน้ำเงี้ยว  ผู้คน ล้างหัว ล้างหน้า
- ปัญหาง่ายๆ คือ ล้างหัวแล้วล้างหน้า ไม่เกิด Race condition

