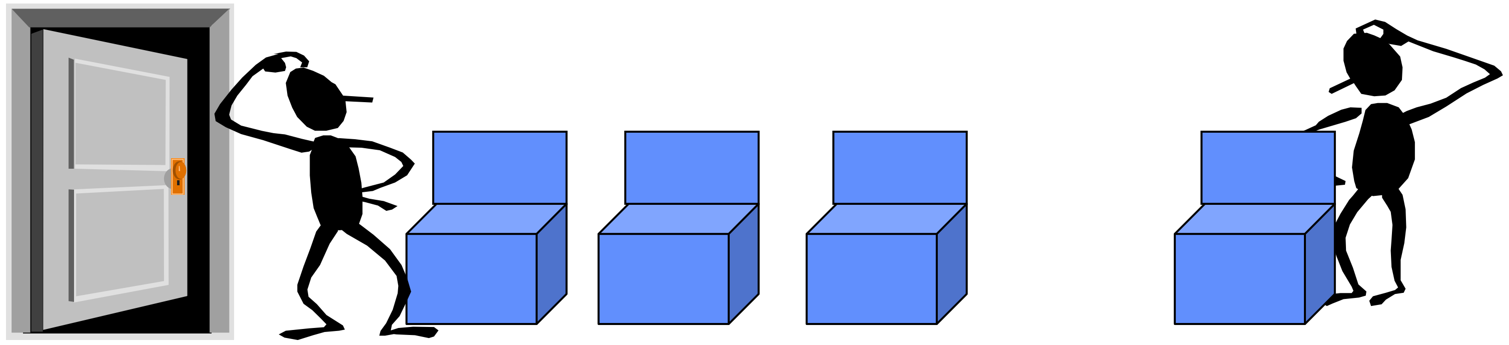
The Sleeping Barber



Customers who need a haircut enter the waiting room. If the room is full, the customer comes back later. If the barber is busy but there is a vacant chair the customer takes a seat. If the waiting room is empty and the barber is daydreaming the customer sits in the barber chair and wakes the barber.

When the barber finishes cutting a customer's hair, the barber fetches another customer from the waiting room. If the waiting room is empty the barber daydreams.

The Sleeping Barber

```
Semaphore mutex=1;
Semaphore customers=0, cutting=0, barber=0;
int waiting=0, numChairs= ..;
```

```
process Customer[i=1..M]
while (true) {
  wait(mutex);
  if (waiting<numChairs){
    waiting++;
    signal(customers);
    signal(mutex);
    wait(barber);
    signal(cutting);
  }
  else up(mutex);
}
end Customer;
```

```
process SleepingBarber
while (true) {
  wait(customers);
  wait(mutex);
    waiting--;
    signal(barber);
  signal(mutex);
  wait(cutting);
}
end SleepingBarber;
```

The Sleeping Barber

This example illustrates the use of semaphores for both mutual exclusion and condition synchronization. The customers and barber use semaphores to control each other's flow through the barber shop. This interaction is an example of a client-server relationship. The customer and barber rendezvous to interact: each waits at a certain point for the other to arrive.

Exercises with semaphores

A process B must do an operation opB() only after a process A has done operation opA().

B: opB() ... A: opA()

How can you guarantee this using semaphores?

Exercises with semaphores

Consider the tree following processes

P1	P2	P3

`Print(R)`
`Print(OK)`
	`Print(I)`
	`Print(OK)`
		`Print(O)`
		`Print(OK)`

Add operations on semaphores such that:

- The result printed is R I O OK OK OK
- The final value of the semaphores is identical to their initial value

Exercises with semaphores

Propose some pseudo-code for processes that have the following behavior: the processes A and B may execute operations opA and opB (in any order) only after a process C has done operation opC.

Do not forget to indicate the initial value of the semaphores.

Exercises with semaphores

Consider the following processes, where initially y = z = 0:

Process 1	Process 2
`// initialization code`	`// initialization code`
`integer x;`	
	`y = 1;`
`x = y + z;`	`z = 2;`
`// other code...`	`// other code...`

1. What are the possible final values for x?

2. Is it possible, using semaphore, to have only two values for x?

Exercises with semaphores

Is this a solution to mutual exclusion?

```
C1, C2 : INTEGER := 1;
```

```
task P1;
task body P1 is
begin
  loop
    NON_CRITICAL_SECTION_1;
    loop
      exit when C2 = 1;
    end loop;
    C1 := 0;
    CRITICAL_SECTION_1;
    C1 := 1;
  end loop;
end P1;
```

```
task P2;
task body P2 is
begin
  loop
    NON_CRITICAL_SECTION_2;
    loop
      exit when C1 = 1;
    end loop;
    C3 := 0;
    CRITICAL_SECTION_2;
    C2 := 1;
  end loop;
end P2;
```

Exercises with semaphores

P1	P2
`print(A);`	`print(E);`
`print(B);`	`print(F);`
`print(C);`	`print(G);`

Insert semaphores to satisfy the properties:

- print A before printing F
- print F before printing C

Don't forget to indicate the initial value of the semaphores.

Exercises with semaphores

P1	P2

`print(C);`
`print(E);`
	`print(A);`
	`print(R);`
	`print(O);`

Insert semaphores such that only ACERO or ACREO is printed.

Don’t forget to indicate the initial value of the semaphores.

Exercises with semaphores

P1	P2	P3

`repeat`
  `print(A);`
  `SC.V;`
  `SA.P;`
`forever`
	`repeat`
	  `print(B);`
	  `SC.V;`
	  `SB.P;`
	`forever`
		`repeat`
		  `SC.P;`
		  `SC.P;`
		  `print(C);`
		  `SA.V;`
		  `SB.V;`
		`forever`

Assuming semaphores SA, SB, and SC are initialized at 0, what strings can be printed?

Exercises with semaphores

```
y = 0; (initial value)
r = 1; (initial value)
x = 2; (initial value)
z = 0; (initial value)
```

```
A:
  r:=r+1
  x:=x+1
  print(x)
  print(r)
```

```
B:
  y:=x
  z:=z+1
  print(z)
```

What semaphores should you add such that the final value of y is 3?

Exercises with semaphores

```
  semaphore mutex = 1;
  semaphore times_a = 2;
  semaphore times_b = 0;
```

```
A:
repeat forever:
  P(times_a)
  P(mutex)
    <A1>
  V(mutex)
  V(times_b)
```

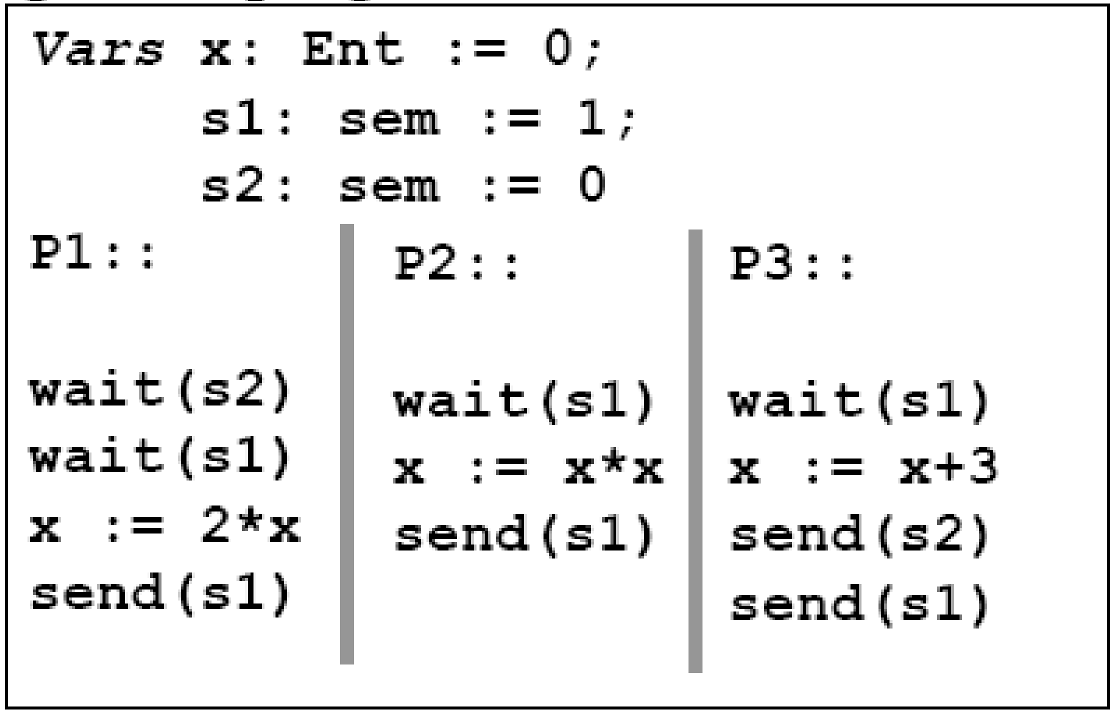
```
B:
repeat forever:
  P(times_b)
  P(mutex)
    <B1>
  V(mutex)
  V(times_a)
```

The concurrent execution of A and B produces an infinite sequence of <A1> and <B1>. Which one is the only possible start of the sequence:

1. A1,A1, B1,A1,A1, B1,A1,A1, B1, ...
2. A1, B1,A1,A1, B1,A1, B1,A1,A1, ...
3. A1, B1,A1, B1,A1, B1,A1, B1,A1, ...
4. A1,A1, B1, B1,A1, B1, B1,A1,A1, ...

Exercises with semaphores

Which are the possible values for X ?
Send() == signal()



```
Vars x: Ent := 0;
     s1: sem := 1;
     s2: sem := 0
P1::          | P2::         | P3::
              |              |
wait(s2)      | wait(s1)     | wait(s1)
wait(s1)      | x := x*x     | x := x+3
x := 2*x      | send(s1)     | send(s2)
send(s1)      |              | send(s1)
```