Process Synchronization -2

CS3600 Spring 2022

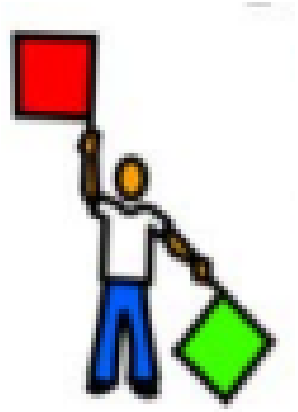
Semaphore

- ***Semaphore (S)*** is a type of generalized lock

  Defined by Dijkstra in the last 60s

  Main synchronization primitives used in UNIX

  Two types **Binary semaphore and Continuous Semaphore.**



  - Two operations
    - ***P()***: an atomic operation that waits for semaphore to become a positive value, then decrement it by 1, else wait until s is a positive number. **[STOP]**
    - ***V()***: an atomic operation that increments semaphore by 1 and wakes up a waiting thread at P(), if any. **[GO]**

Note: P and V from the Dutch words Probeer (try) and Verhoog (increment)

Semaphore functions

- Can only be accessed via two indivisible (atomic) operations **`wait()`** and **`signal()`** originally called **`P()`** and **`V()`**

- Definition of the **`wait()`** operation P(s) :
  - **if s > 0, decrement s by 1, otherwise wait until s > 0**

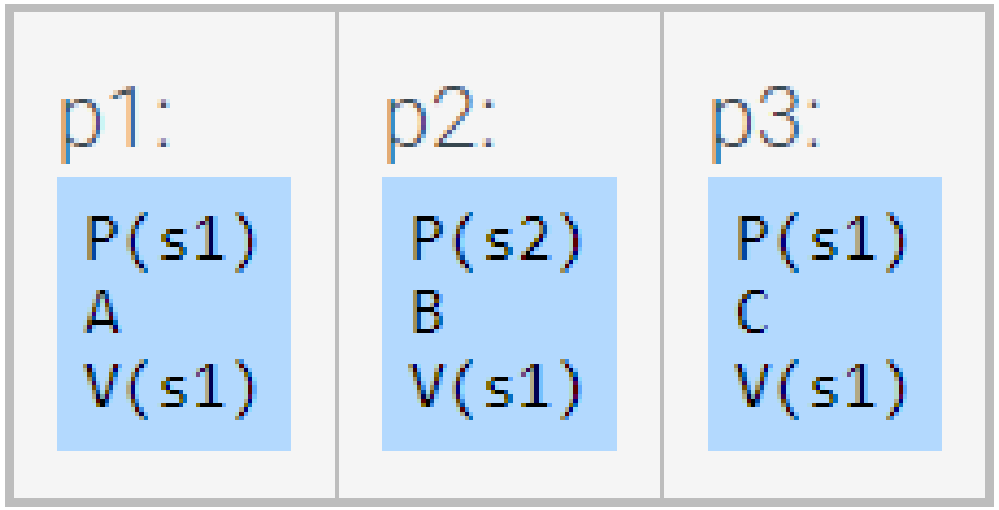
```
wait(S) {
    while (S <= 0)
        ; // busy wait
    S--;
}
```

- Definition of the **`signal()`** operation V(s): increment s by 1

```
signal(S) {
    S++;
}
```

Example 1

- If s1 = 0 and s2 = 1, in which order can the statements A, B, and C execute.



Example - precedence

Process P1	Process P2
A	C
B	D

- A precedes D
- C precedes B

Semaphore for Resource Allocation

- Let the number of resources is R.
- Objective : sharing R resources without conflict

- Semaphore solution:

  In the shared memory semaphore S = R //R resources
- Using Resources

  wait(S) //allocate resource

  …….. // using resource

  signal(S) //return resource

  //each time semaphore will be updated with resource let in the pool

Bounded buffer Problem -Semaphore

Buffer

If consumer reads before producer writes ?

Producer

Consumer

Bounded buffer Problem –Semaphore Cont.

1. One to maintain number of items in buffer so consumer don't read from an empty buffer.
   - Every time producer writes it signals

Full = 0

Producer
Produce Item
V(Full)

Consumer
P (Full)
Consume Item

Bounded buffer Problem –Semaphore Cont.

Now consider buffer overflow where buffer is not overwritten with a 9th element in the without the consumer being read.

Full = 0

Empty = 8

Producer
P(Empty)
Produce Item
V(Full)

Consumer
P (Full)
Consume Item
V(Empty)

Bounded buffer Problem –Semaphore Cont.

The buffer space should be protected from other process to access, so we add a mutual exclusion lock for that.

Buffer

Mutex =1

Full 0

Empty 8

Producer

P (Empty)
P(Mutex)
Produce Item
V(Mutex)
V(Full)

Consumer

P (Full)
P(Mutex)
Consume Item
V(Mutex)
V(Empty)

Semaphore in Pthread

```
#include <semaphore.h>

    sem_t  m; // declare semaphore
    sem_init(&m,0,<initial count>)  //initialize with initial count
    sem_wait(&m);   // wait()
    //Critical section
    sem_post(&m);  // signal()
```

Compile the program using

gcc filename.c -o objectname -lpthread

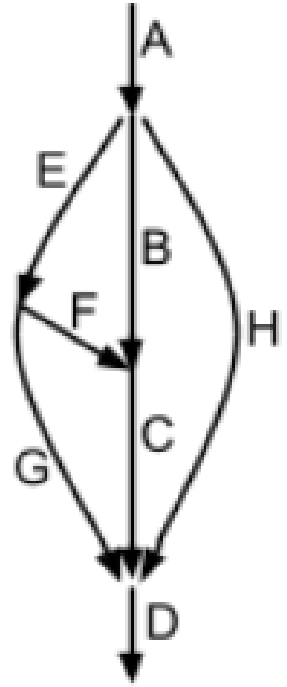
Classwork- Semaphores

- Classwork can be done as Team of 3 and only one member must submit with the members name in comments.

Classwork 01 -Semaphores

- Four Process are executing the computations A – H. In the given order.

p1:	p2:	p3:	p4:
A B C D	E F	G	H



- Use 5 semaphores and enforce the precedence.

- Submit as a pdf

Classwork 02 – **Lab 5**

- Use Program sem1.c to fix the program to get an expected result so that the program always produces the expected output

  (the value 2*NUM)

- Submit as a c file

Announcements on 02/24/2022

- Solutions for Classwork-semaphores will be posted tomorrow at 4 pm.

- Homework on Semaphores will be posted tomorrow at 4pm

Next class - Basic solution R/W problem

```
wait ( mutex)
Write Here
signal (mutex)
```

```
wait ( mutex)
Read here
signal (mutex)
```