Semaphore

- Synchronization
  - Critical Section
  - Mutual Exclusion
  - Fairness

Processes

Spring 2018

CSE 330: Operating Systems

Class: 15

Date: 2/27
\[ \{ \alpha \} \rightarrow \{ \gamma \} \]

Definition:
initial \( s \) = value

Another \( \gamma \)

Example:
if \( s = 0 \) then \( \gamma \)
not good

1. Blooding Somophore
2. Busy most somophore

Impromutability somophore
\( p(s) \)  

\[
\begin{aligned}
\text{if } s > 0 & \rightarrow x - - \text{ Enabled} \\
\text{else} & \rightarrow \text{ Enabled} \\
\end{aligned}
\]
\{ s. l \leq 0 \}
\{ \text{count} = 1 \}
\text{until} \; \text{some} (s', a) \;
\text{next} \;
\text{init} \;
\text{assert} \;
\text{modelling} \text{ Someplace} \n\text{Processor}
\{ w \geq 0 \} \land \{ v = 0 \} \land \{ \text{lock}(i) \} \land \{ \text{lock}(i+1) \} \land \{ \text{lock}(i+2) \} \land \ldots
\[ P(s) \Rightarrow \text{lock}(s, L) \]

\[
\text{if } (s\text{.count} > 0) \quad \begin{cases} 
\text{unlock}(s, L) \\
\text{else} \quad \{ \text{unlock}(s, L) \}
\end{cases}
\]

\[ \text{goto} \]
typedef Semaphore
{
    int count;
    int l;
} PCB queue;
Call to routine in `scheduler`

Dependent code

```
if (scheduler (S.L) <= L Lock (S.L))
```
which (S, L) 3

\[
\begin{align*}
\text{Read 3} \\
\text{S, L and add a} \\
\text{blank} & \text{ if low row} \\
(0 = \geq \text{count}) \quad \text{if} \\
\text{S, count ++} \\
\text{look (S, L)} \\
V(S) \leq 16k \text{(S, L)} \\
\end{align*}
\]
\[ \text{limit}(s, a) \]
\[ \text{in specifications}(s, a) \]
\[ \text{seen} \rightarrow \text{count}, \text{a} \]
{ }  

swapPrevAt (cur, 

addG (s, g, cur) 

ext = G 

curr = d = G (R, G) 

} if (s <= s <= 0) 

} s. lower --
Add up runs, only one
\[
\{ f_i(s) \text{ count } + \}
\]
Vanderbilt that has no value
as many of a class
class or description does this

Conditionals \[ \text{not} \text{ condition} \]
something \[ \text{not} \text{ something} \]
Memes
if do not
3 

3 

3 

matched

matched

matched

natzahl n_i

\text{condición } (i < 2, \text{cond})

x . m_2 (i)

x . m_1 (i)

\exists x . m_1 (x)
Monitors Semantics

Mutiple threads can call

execute in one critical section

will not exceed
Condition variables support 2 methods

- **wait** (cond) → step out of monitor
  - block
  - when awakened, re-enter monitor

- **signal** (cond) → wake one process up
  - only used in monitor method
  - if there are any blocked on the cond.
or 6 Watts
The process is as follows:

1. Draw process steps
2. Make process safe

Option 2

Option 1

\[ \text{Option 1} \]

\[ \text{Option 2} \]

Worst case can be due to:

- Process Exxon or APEC
- West, all surrounding
- West, process, less by

\[ \text{Worst (C)} \]

\[ \text{Sign} \]

\[ (C) \]

\[ \text{Worst (C)} \]

\[ (C) \]

\[ 3 \]