implement function
no system call
user level

spawns threads
memory

user level threads

Synchronization & mutual exclusion
By system call

- system read -> pare ce/bleuer
  - spinning in key
  - corner changes of
    - small with desk screen
    - less之城

[system call]

- spin reader -> no system call

Remed least thrive
Let's no spin back

Lose money→ Scheer

Keenly read

Let's spin back

Lose money → no Scheer

Spin + read back
\[ A \subseteq B \]

\[ \forall (x) \exists (y) 
\]

\[ 1 \leq 2 
\]

\[ s = 0 \]

Semantics
2. math

\[ \text{Sign (G)} \]
\[ \text{Write (G)} \]
\[ \text{Write (A)} \]
\[ \text{Write (C)} \]

\text{Convert C1, C2}
before

wait (c1)

signal (c2)

signal (c1)

wait (c2)

after
```c
before (c) { if (a == 0) { sig(c) + b++ } for (i = 0; i < 3; ++i) { count(c) } } if (a != 0) {
```
Barrier Synch

\[ \begin{align*}
T_1 & \quad T_2 & \quad T_3 & \quad T_4 \\
A & \quad C & \quad E & \quad G \\
B & \quad D & \quad F & \quad H \\
\end{align*} \]

\[ [A, C, E, G] \rightarrow [B, D, F, H] \]
Solution

1. `barrier(n)`
2. `if (count % n == 0) {
   signal(c);
   count++
   n++
}
3. `wait(c)`
4. `signal(c)`

Using

monitor

A C E H
else count -- \lor \text{ Wait}
\end{enumerate}
\begin{enumerate}
\item \text{(count < n)}
\item \text{(count > n)}
\item \text{(count = n)}
\end{enumerate}
Process

Base

Process

Process

Process

Process

Process

Process

Process

Process
elaborated / supplemented

perceived benefits

useful and unnecessary

case by case

effective

Barric