Quiz 4

```c
int Quizzes

for (i = 1; i <= 6; i++)
    if (func(i) == x)
        printf("x%d", i);
```

---

Threads : Processes
1: 1: 1 old machine [MS-DOS]
2: M: 1 Unix
3: 1: M Thread migration
4: M: N Combine (2 + 3)

Read about microkernels

Symmetric Multiprocessing (SMP)

Parallel Processing
Multiple Instruction
Multiple Data

Single Instruction SIMD
Multiple Data

MIMD
MIMD

Shared Memory
Tightly Coupled

Distributed Memory
Loosely Coupled

Master/Slave
Symmetric

CPU1
CPU
SMP

RAM

Read: Inter Process Communication

Next chapter
Deadlocks

\[ P_1 \]
\[ \rightarrow \text{wait} (S_1) \]
\[ \rightarrow \text{wait} (S_2) \]
\[ \rightarrow \text{signal} (S_2) \]
\[ \rightarrow \text{signal} (S_1) \]

\[ P_2 \]
\[ \rightarrow \text{wait} (S_2) \]
\[ \rightarrow \text{wait} (S_1) \]
\[ \rightarrow \text{signal} (S_1) \]
\[ \rightarrow \text{signal} (S_2) \]

Deadlock handling:

A. Deadlock Prevention

B. Deadlock Avoidance / Detection

C. Recovery

Four necessary conditions for deadlocks:

1. Mutual Exclusion of Resources
2. Hold and Wait of Resources
3. No Preemption of Resources
4. Circular Wait
No circular wait.

1. Statically order all resources; Common to all processes.
   Example: \( R_1 < R_2 < R_3 \ldots < R_m \)

2. Processes can request resources that are (ordered) strictly higher than what they hold.

**Claim:** If processes follow these 2 rules, there is no circular wait.

**Holt Graph**

- \( P \) for process
- \( R \) for resource

\( P \) is waiting for \( R \)

\( R' \) is held by \( P' \)
Proof: By contradiction. Assume \( \exists \) a circular work involving \( P_1, P_2, \ldots, P_n \).

Consider \( P_2 : R_1 < R_2 \) and \( P_3 : R_2 < R_3 \).

By transitivity, \( R_1 < R_n \).

Consider \( P_1 : R_n < R_1 \). Contradiction.

\( \exists \) a cycle.
## Banker's Strategy

<table>
<thead>
<tr>
<th>Customers</th>
<th>Max-Needs</th>
<th>Current Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>7 1</td>
<td>0 5 0</td>
</tr>
</tbody>
</table>

**Current Availability**

\[
\begin{bmatrix}
25 \\
15 16 8 1
\end{bmatrix}
\]

*5 is Safe state if starting from 5, there exists at least one sequence in which all customers can complete multiple resource types & multiple identical units in each resource type.*

**Example:**  \( m = 3 \) resource types & system capacity: \((10, 15, 12)\)
CAR - Currently Available Resources
= \begin{pmatrix} 10 & 10 & 10 \end{pmatrix} \Rightarrow m = 3

n \text{ processes.}

\text{MAX-Request}_i = \text{Max resource req. of process } P_i.

\text{Curr-Allocation}_i = \text{what is currently allocated to } P_i.

\text{if } x \leq y \text{ each component of } x \text{ is less than or equal to corresponding component of } y.

\text{Algorithm to check if a given state is safe:}
\text{input: } \text{CAR, MAX-Request}, \text{ CURR-Allocation, } \forall i
\text{done} \leftarrow \text{false}; \quad \text{for } P = \{P_1 \ldots P_n\}.
\text{while } \text{not done} \text{ do }
\text{if } P = \emptyset \text{ then } \{\text{done } \leftarrow \text{true; goodbye}\}
\text{else if } \exists P_i \in P \text{ s.t. } \text{MAX-Request}_i \geq \text{Current-Allocation}_i \leq \text{CAR}
\{\text{CAR} += \text{Curr-Allocation}_i; \text{ delete } P_i \text{ from } P;\}
\text{else } \{\text{done } \leftarrow \text{true; // bad state}\}
\text{if } P = \emptyset \text{ good state else bad state}
Running time?

$O(n \cdot n \cdot m)\uparrow$

$\#$ of iterations of loop

**Example:** $CAR = (2,1,3)$  System-Capacity = (6,3,8)

$n = 4$

<table>
<thead>
<tr>
<th>$i$</th>
<th>Max-Request</th>
<th>Current-Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,3,1)</td>
<td>(1,0,1) (0,3,0)</td>
</tr>
<tr>
<td>2</td>
<td>(5,2,4)</td>
<td>(0,1,2) (5,1,2)</td>
</tr>
<tr>
<td>3</td>
<td>(5,3,2)</td>
<td>(1,0,2) (4,3,2)</td>
</tr>
<tr>
<td>4</td>
<td>(3,2,2)</td>
<td>(2,1,0) (1,1,2)</td>
</tr>
</tbody>
</table>

$P > \{ P_1, P_2, P_3, P_4 \}$

Delete $P_4$

$CAR = (2,1,3)$

$+ (2,1,0)$

$CAR = (4,2,3) \quad p = (P_1, P_2, P_3)$

Unsafe.

**Example:** $CAR = (2,0,1)$

<table>
<thead>
<tr>
<th>$i$</th>
<th>Max-Request</th>
<th>Current-Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,2,3)</td>
<td>(0,1,2)</td>
</tr>
<tr>
<td>2</td>
<td>(5,2,2)</td>
<td>(2,0,1)</td>
</tr>
<tr>
<td>3</td>
<td>(2,1,1)</td>
<td>(1,1,0)</td>
</tr>
<tr>
<td>4</td>
<td>(4,3,3)</td>
<td>(0,1,0)</td>
</tr>
</tbody>
</table>
Deadlock Avoidance:

& request by a process, say P_i 
for additional resources: ≤

S ≤ current state:

Assume request by P_i is granted and go to state S':

if S' is P safe, 
& allocate resources to P_i

current state = S';

else block P_i;

you go from one safe state to another safe state.