第19 - 20讲 进程的并发控制概述及软件实现方法

内　　容（2.6 - 2.7）

1 Mutual Exclusion and Synchronization

2 Deadlock And Starvation

Learning objectives (2.6 - 2.7)

By the end of this lecture you should be able to:

- Explain what's Concurrency, Synchronization ,Mutual exclusion, Deadlock, Starvation, Critical sections
- 掌握 Requirements for Mutual Exclusion
- 掌握 Approaches of Mutual Exclusion: Software Approaches & Hardware Support—Semaphores 、 Monitors 、 Message Passing
- 区别掌握 Types and meanings of Semaphores

Learning objectives (2.6 - 2.7) （ continue ）

By the end of this lecture you should be able to:

- 掌握 3 个经典问题的解决方法： Producer/Consumer Problem 、 Readers/Writers Problem 、 Dining Philosophers Problem
- 理解 Conditions for Deadlock 、 Deadlock Prevention 、 Deadlock Avoidance 、 Deadlock Detection 、 Strategies once Deadlock Detected, Banker's Algorithm (Safe State *vs.* Unsafe State)

§2.6 Concurrency ：
Mutual Exclusion and Synchronization

电子科技大学
University of Electronic Science and Technology of China

Design Issues of Concurrency

- Communication among processes
- Sharing /Competing of resources
- Synchronization of multiple processes
- Allocation of processor time

Difficulties with Concurrency

- Sharing global resources
- Management of allocation of resources
- Programming errors difficult to locate

A Simple Example

```
void echo()
{
    chin = getchar();
    chout = chin;
    putchar(chout);
}
```

A Simple Example

Process P1

....

in = getchar();

....

chout = in;

....

putchar(chout);

....

Process P2

....

in = getchar();

....

chout = in;

....

putchar(chout);

Operating System Concerns

- Keep track of active processes: PCB
- Allocate and deallocate resources
  - Processor time: scheduling
  - Memory: virtual memory
  - Files
  - I/O devices
- Protect data and resources
- Result of process must be independent of the speed of execution of other concurrent processes.

Process Interaction

- **Processes unaware of each other**
  - Competition
  - Mutual exclusion, Deadlock, Starvation
- **Processes indirectly aware of each other**
  - Cooperation by sharing
  - Mutual exclusion, Deadlock, Starvation, Data coherence (一致性)
- **Process directly aware of each other**
  - Cooperation by communication
  - Deadlock, Starvation

Competition Among Processes for Resources

- **Mutual Exclusion(互斥)**
  - **Critical sections （临界区）**
    - **Only one program at a time is allowed in its critical section.**
    - **Eg. Only one process at a time is allowed to send command to the printer （*critical resource*）.**
- **Deadlock （死锁）**
- **Starvation**

Competition Among Processes for Resources

Deadlock

例如，有两个进程 P1 、 P2 ，竞争两个资源 A 、 B 。假设

占用： P1 （ B ） and P2 （ A ）

申请： P1 （ A ） and P2 （ B ）

结果： P1 、 P2 永久等待（死锁）

P1 & P2 - Deadlock

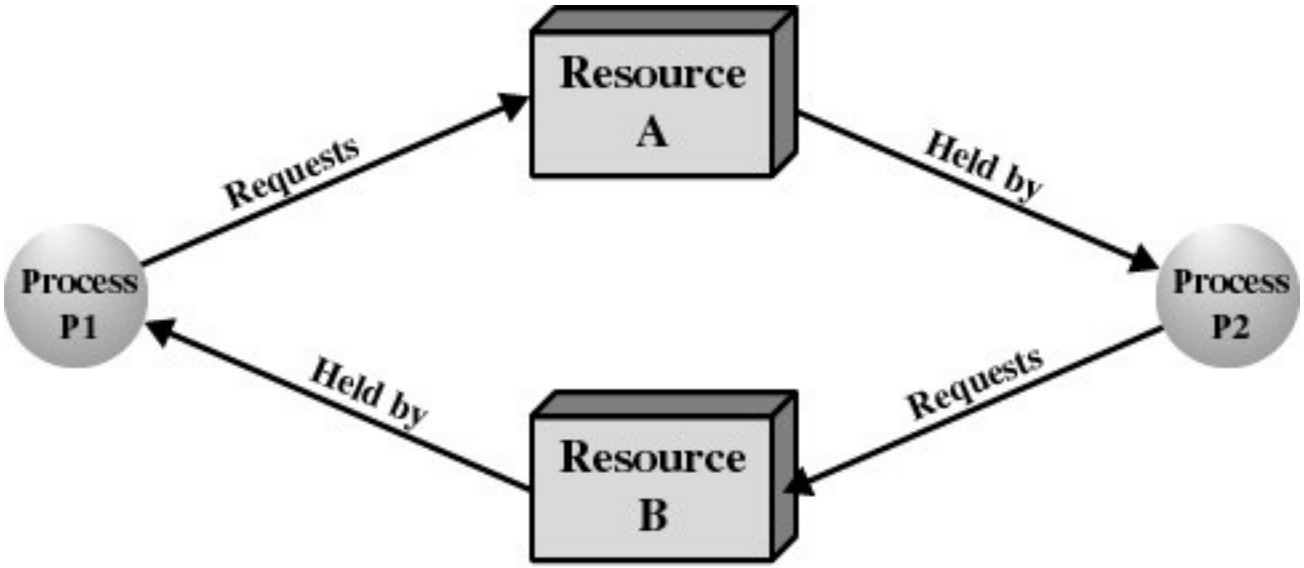

Figure 6.5 Circular Wait

Mutual Exclusion Mechanism

entry section

critical section

exit section

Cooperation Among Processes by Sharing

- Writing must be mutually exclusive.
- Critical sections are used to provide data integrity （数据完整性） .

Cooperation Among Processes by Communication

- **Messages are passed**
  - Mutual exclusion is not a control requirement.
- **Possible to have deadlock**
  - Each process waiting for a message from the other process.
- **Possible to have starvation**
  - Two processes sending message to each other while another process waits for a message.

Requirements for Mutual Exclusion

- Only one process at a time is allowed in the critical section for a resource .
- A process that halts in its non-critical section must do so without interfering with other processes.
- No deadlock or starvation.

Requirements for Mutual Exclusion

- A process must not be delayed access to a critical section when there is no other process using it.
- No assumptions are made about relative process speeds or number of processes.
- A process remains inside its critical section for a finite time only.

Requirements for Mutual Exclusion

- *空闲让进*
- *忙则等待*
- *有限等待*
- *让权等待*

Approaches of Mutual Exclusion

- ***Software Approaches***
- **Hardware Support**
- **Semaphores**
- **Monitors**
- **Message Passing**

Software Approaches

- Memory access level
- Access to the same location in main memory are serialized by some sort of memory arbiter .
- Dekker's Algorithm
- Peterson's Algorithm

Software Approaches

- To mutual exclusion for two processes.
- To illustrate most of the common bugs encountered in developing concurrent programs.
- Is likely to have high processing overhead.
- The risk of logical errors is significant.