CPT104

> 内容完全来自于**LearningMall**的PPT。仅供学习和交流使用。任何机构或个人不得将其用于商业用途或未经授权的传播。如需引用或分享，请注明出处并保留此声明。
>
> GITHUB REPOSITORY
>
> VISIT NOTES AUTHOR'S PAGE CLICK ME

CPT104 **Operating Systems Concepts** 操作系统概念

■ [Week 1 Lecture 0]

课程信息

学分：**5** credits

项目	占比	其他信息
Final Exam	80%	2 小时。
CW1	20%	Week 7
CW2	20%	Week 12

模块负责人及联系方式

- **姓名**: Gabriela Mogos
- **电子邮件**: Gabriela.Mogos@xjtlu.edu.cn
- **办公室电话**: 88161515
- **办公室地点及办公时间**: SD547；周一 13:00-14:00，周二 15:00-16:00，或预约

[Module Handbook] 学生须达到不低于80%的出勤率。未能遵守此要求可能导致考试失败或被禁止参加补考。

> Failure to observe this requirement may lead to failure or exclusion from resit examinations or retake examinations in the following year.

内容

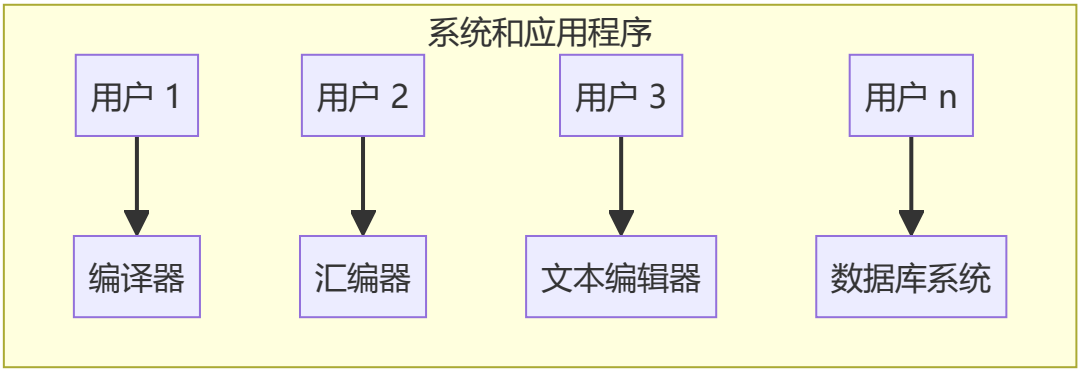
1. 操作系统（Operating Systems concept）的概念
2. **进程**（Process Concept）概念
3. **进程调度**（Process Scheduling）
4. 进程操作（Operations on Processes）
5. 进程间通信（Inter-process Communication）

[1] 操作系统的概念

操作系统可以理解为：**用户和硬件之间的接口。**或者说，一种"**架构（architecture）**"环境。

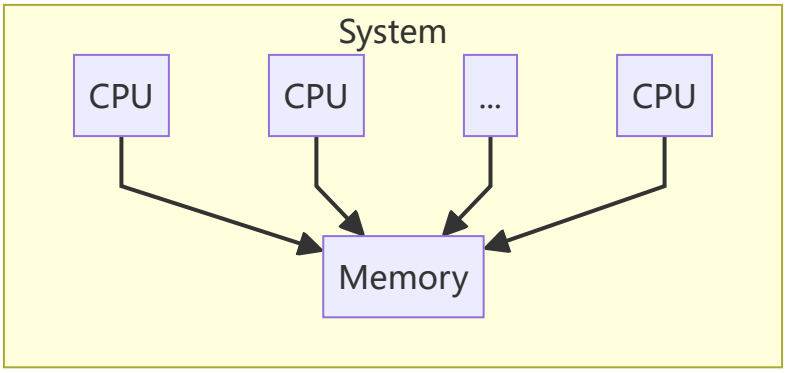
- 允许方便的使用；隐藏繁琐的部分
- 允许高效的使用；**并行活动（parallel activity）**，避免**浪费的周期**
- 提供信息保护
- 为每个用户提供一个**资源片段（a slice of the resources）**
- 作为控制程序发挥作用

> **并行活动**：指的是操作系统能够同时执行多个任务或进程，让多个程序或用户共享系统资源，从而提高系统的效率和响应速度。
>
> 避免**浪费的周期**：指的是操作系统通过合理分配和利用计算资源，避免资源闲置或浪费，确保每个资源（如CPU、内存等）都被充分利用。

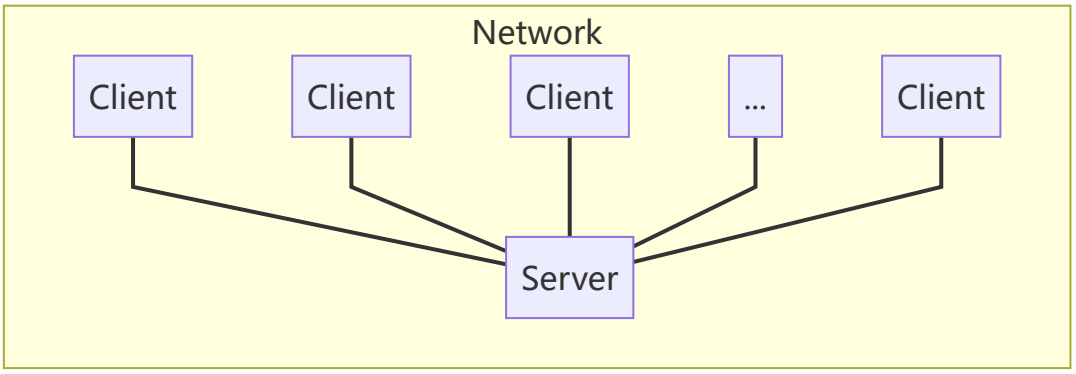


操作系统的特征：

1. **时间共享（Time Sharing）** - 允许多个用户**同时**使用一台计算机系统。
2. **多处理（Multiprocessing）** - 通过**共享内存**实现紧密联系的系统。通过将多个现成的处理器组合在一起，**提升计算速度。**



3. **分布式系统（Distributed Systems）** - 通过**消息传递**实现松散联系的系统。其优点包括**资源共享、加快处理速度、提高可靠性**以及**增强通信能力。**



操作系统中最常见的操作：

1. 进程管理（Process Management）
2. 内存管理（Memory management）
3. 文件系统管理（File System Management）
4. 输入/输出系统管理（I/O System Management）
5. 保护与安全（Protection and Security）

[2] 进程概念

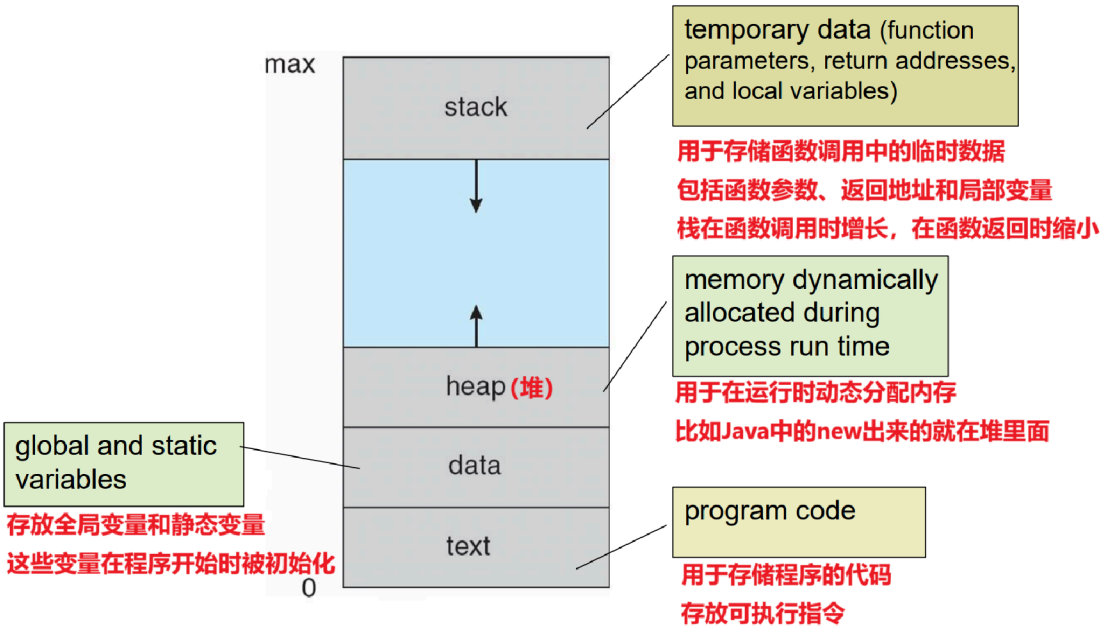
进程 = 正在执行的程序 （Process = a program in execution）

进程的执行必须**按顺序进行(In sequential fashion)。**

进程被视为"**活动（active）**"实体。

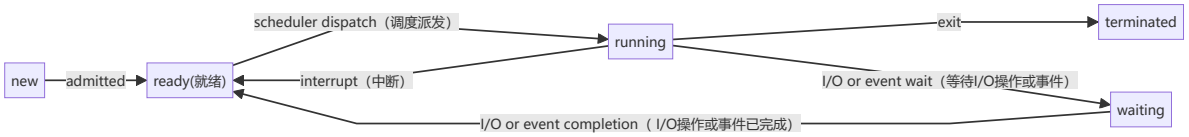
程序被视为"**被动**（Passive）"实体（存储在磁盘上的可执行文件（executable file））

程序在**可执行文件加载到内存时**成为进程。（Program becomes process when **executable file loaded into memory**）



一个进程执行时改变状态，进程的状态在某种程度上由该进程**当前的活动**来定义。

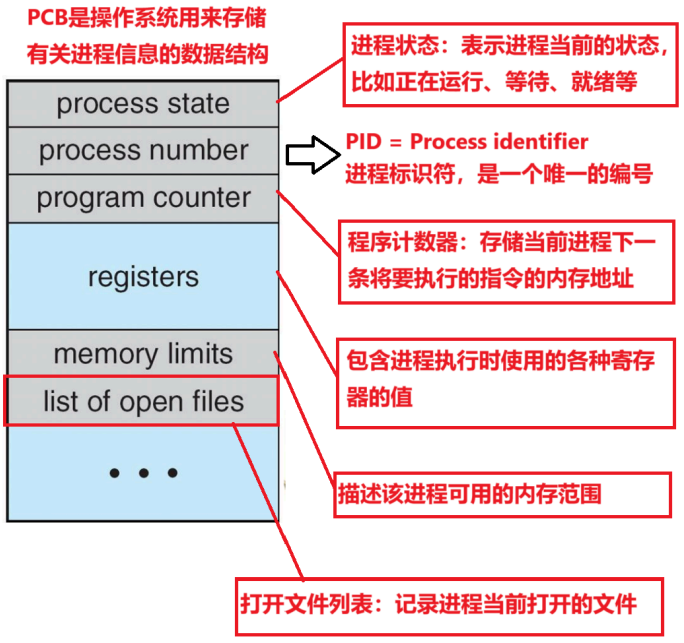
- **new（新建）**：进程正在被创建
- **running（运行）**：指令正在被执行
- **waiting（等待）**：进程正在等待某个事件发生
- **ready（就绪）**：进程正在等待被分配给处理器
- **terminated（终止）**：进程已完成执行



进程控制块（Process Control Block, PCB）是一种用于存储进程相关信息的数据结构。

- 每个进程都有其自身的PCB。
- PCB也被称为进程的**上下文（context）**。
- 每个进程的PCB都存储在主内存中。
- 所有进程的PCB都存在于一个链表中。
- 在**多道程序**(Multiprogramming)设计环境中，PCB至关重要，因为它包含与同时运行的多个进程相关的信息。

> **多道程序**：一种计算机操作系统技术，允许多个程序（程序1、程序2、程序3等）同时存在于内存中，并**轮流使用CPU**。虽然CPU同一时间只能执行一个程序，但操作系统通过快速切换，让用户感觉多个程序在同时运行。



---

[3] 进程调度

进程的执行由**CPU执行**和**I/O等待**的**交替序列**组成。

> **CPU执行**：进程在运行时，CPU会执行程序中的指令
>
> **I/O等待**：当进程需要进行 **输入/输出** 操作（如从硬盘读取文件、向显示器输出数据）时，CPU会切换到等待状态，因为这些操作需要外部设备的响应。
>
> **交替序列**：进程的运行不是一直占用CPU，而是时而工作（CPU执行），时而等待（I/O等待）

进程调度器(Process scheduler) 从内存中选择已准备好执行的进程，并将CPU分配给其中一个进程。

进程的调度队列：

- **作业队列（Job queue）**：系统中所有进程的集合。
- **就绪队列（Ready queue）**：所有已准备好并且等待执行的进程的集合。
- **设备队列（Device queues）**：等待某个I/O设备（如硬盘、打印机等）的进程的集合。

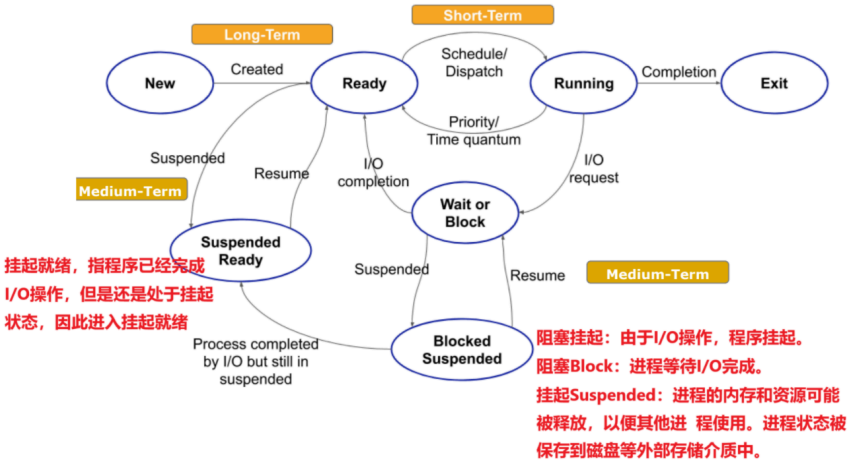
调度器种类

长期调度器(Long-Term Scheduler)：也被称为**作业调度器(Job Scheduler)**。控制系统的多道程序设计程度，管理处于**就绪（Ready）**状态的进程数量。

短程调度器（Short-Term Scheduler）：也被称为**CPU调度器(CPU scheduler)**：负责从就绪队列中**选择**一个进程，并将其调度到**运行（Running）**状态（CPU执行）。

中期调度器（Medium-term scheduler）：负责进程的内存与外存交换（如将进程从主内存换出到磁盘，或从磁盘换入主内存），对系统性能产生中期影响

> 上学期期末考有一题就是页面切换（内存到硬盘之前切换），算时间的，就是中期调度器做的事情。



上下文切换

当CPU切换到另一个进程时，系统必须保存旧进程的状态并通过**上下文切换（CONTEXT SWITCH）**加载新进程的保存状态。进程的上下文信息在**进程控制块（PCB）**中表示。

[4] 进程操作

操作系统必须提供必要的进程操作：**进程创建（Process creation）**、进程终止（Process termination）。

进程创建

父进程创建子进程，子进程又创建其他进程，形成一棵**进程树。**

资源共享有三种选项

1. 父进程和子进程共享所有资源（类似`Public`）
2. 子进程共享父进程资源的一个子集（类似`Protected`）
3. 父进程和子进程不共享资源（类似`Private`）

执行选项

1. 父进程和子进程**并发执行（execute concurrently）**
2. 父进程等待直到子进程终止

进程终止

进程执行最后一条语句，然后使用`exit()`系统调用请求操作系统删除它

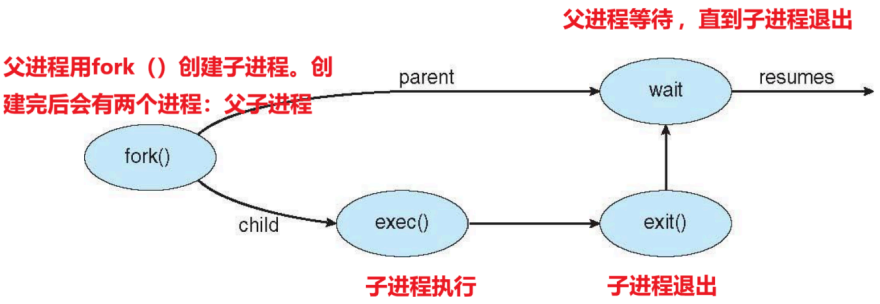
> 子进程主动终结

- 将状态数据（status data）从子进程**返回**给父进程
- 操作系统释放进程的资源

父进程可以等待子进程终止执行

> 这是指父进程的行为。父进程可以使用诸如 wait() 或 waitpid() 的系统调用来等待其子进程的终止。在这个过程中，父进程可能会暂停执行，直到子进程完成并退出。这样做的好处是，父进程可以在子进程结束后获取子进程的退出状态信息，进行资源清理或执行其他逻辑。

- 子进程超出分配的资源



---

[5] 进程间通信

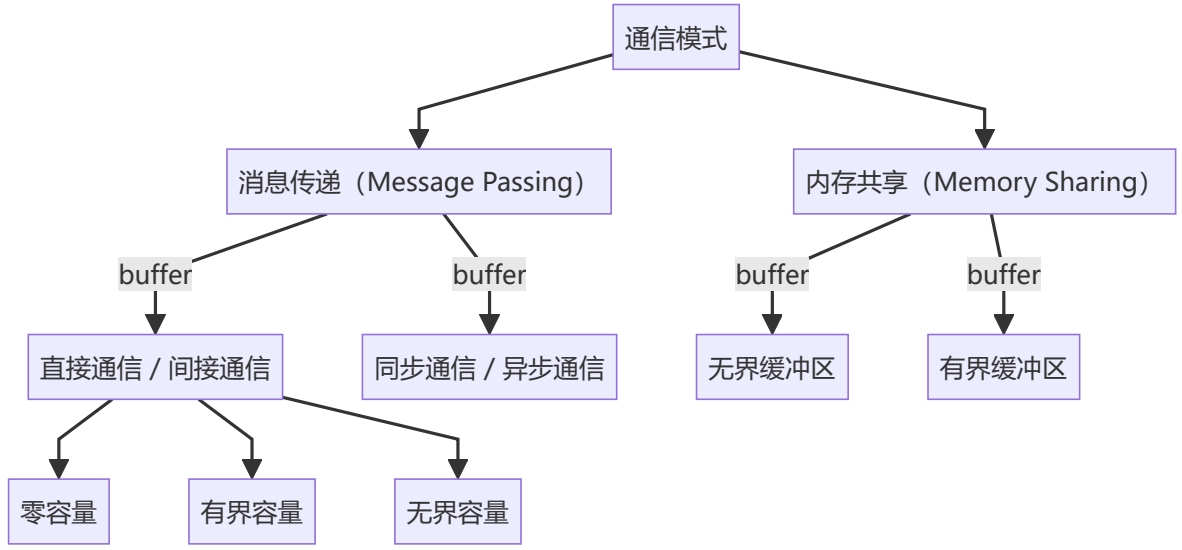
独立进程（Independent Process） - 既不影响其他进程，也不受其他进程影响。

协作进程（Cooperating Process） - 可以影响或被其他进程影响。

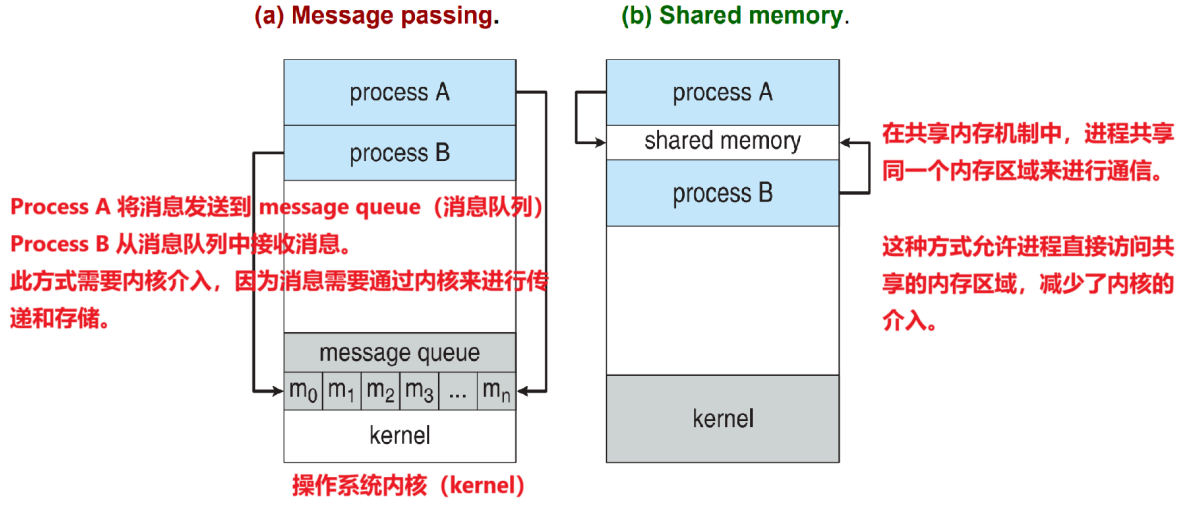
为什么需要协作进程?

- **信息共享（Information Sharing）** - 例如，需要访问同一个文件的进程。
- **计算加速（Computation speedup）** - 如果一个问题可以分解成同时解决的子任务，那么问题可以更快地解决。（比如Java中的`new Thread()`）
- **模块化（Modularity）** - 将系统分解为协作的模块（例如，具有客户端-服务器架构的数据库）。
- **方便性（Convenience）** - 即使是单个用户也可能在多任务处理，例如在不同的窗口中编辑、编译、打印和运行相同的代码。

[6] 通信模式



有 **Message Passing（消息传递）**和 **Shared Memory（共享内存）**两种。



	优点	缺点
Message Passing（消息传递）	1. 提供了严格的进程间隔离，**安全性较高。**2. 适用于分布式系统中的进程间通信。	1. 内核介入增加了通信开销，性能可能较低。2. 消息队列可能成为瓶颈，影响系统效率。
Shared Memory（共享内存）	1. 由于直接访问共享内存，通信速度快，效率高。2. 适用于需要频繁、大量数据交换的场景。	1. 共享内存需要进程间的同步机制，以防止数据竞争和一致性问题。 2. 如果没有适当的访问控制和**同步机制**，安全性较低。（例如并发操作）

综合比较

- **消息传递**适合需要强隔离和简单通信的场景，尽管其性能可能稍差。
- **共享内存**适合需要频繁、高速数据交换的场景，但需要有额外的同步和安全考虑。

消息传递

> [6] 通信模式中的第一种。Message-Passing

至少提供如下两种操作：

1. 发送
2. 接受

如果进程 **P** 和 **Q** 需要通信，必须在他们之间建立一个**通信链路**(communication link)。

逻辑上实现链路和 `send()` / `receive()` 操作的方法有几种：

- **直接通信 / 间接通信 [Direct / Indirect communication]**
- **同步通信 / 异步通信 [Synchronous / Asynchronous communication]**

直接通信 / 间接通信

> [Direct / Indirect communication]

在消息传递系统中，通信进程交换的消息存储在一个**临时队列（temporary queue）**中。

三种**缓冲机制 Buffering**

1. **零容量（Zero capacity）**：队列的最大长度为零，因此通信链路中不能有任何等待的消息。

   > 发送方和接收方必须同时准备好，发送方发送消息后，必须等待接收方接收到消息后才能继续操作。
   >
   > 如果没有接收方准备好，发送方会一直阻塞，直到接收方接收消息。因此也叫做**没有缓冲空间**。

2. **有界容量（Bounded capacity）**：队列的长度为有限值 n，因此最多可以容纳 n 条消息。
3. **无界容量（Unbounded capacity）**：队列的长度没有限制，理论上可以存储无限多的消息。

直接通信

- 进程必须显式地命名彼此：

    - `send(P, message)` —— 向进程 `P` 发送消息。
    - `receive(Q, message)` —— 从进程 `Q` 接收消息。
- 直接通信的实现方式是进程使用特定的**进程标识符（specific process identifier）**进行通信，但在某些场景下，预先确定发送方是困难的。

间接通信

- 创建一个新的邮箱（端口 **port**）。
- 通过邮箱发送和接收消息：
    - `send(A, message)` —— 向邮箱 `A` 发送消息。
    - `receive(A, message)` —— 从邮箱 `A` 接收消息。
- 销毁邮箱。

同步通信 / 异步通信

> [Synchronous / Asynchronous communication]**

消息传递可以是**阻塞（blocking）**或**非阻塞（non-blocking）**的。

阻塞（BLOCKING）被认为是同步的

- 阻塞发送（Blocking send）：发送方会被阻塞，直到消息被接收为止。
- 阻塞接收（Blocking receive）：接收方会被阻塞，直到有消息可用为止。

非阻塞（NON-BLOCKING）被认为是异步的。

- 非阻塞发送（Non-blocking send）：发送方发送消息后可以继续执行。
- 非阻塞接收（Non-blocking receive）：接收方会收到以下两种情况之一：
    - 一个有效的消息
    - 空消息（Null message）

内存共享

> [6] 通信模式中的第二种。Shared Memory

共享内存区域：由协作进程共享的一块内存区域。

信息交换：进程通过读取和写入共享区域中的所有数据来交换信息。

两种缓冲区类型：

- **无界缓冲区（Unbounded-buffer）**：对缓冲区的大小没有实际限制。
- **有界缓冲区（Bounded-buffer）**：假设缓冲区的大小是固定的。