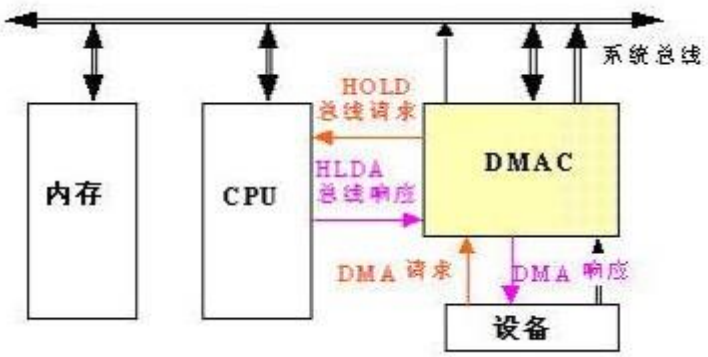
第一章

操作系统（operating system）：操作系统是管理电脑硬件与软件资源的程序，同时也是计算机系统的内核与基石。操作系统是控制其他程序运行，管理系统资源并为用户提供操作界面的系统软件的集合。操作系统身负诸如管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等基本事务。

引导程序（bootstrap program）：计算机初始化程序，位于 ROM 或 EEPROM 中，是计算机硬件中的固件，初始化系统所有部分，包括 CPU 寄存器、设备控制器和内存。

存储器直接访问 DMA（Direct Memory Access）：这是指一种高速的数据传输操作，允许在外部设备和存储器之间直接读写数据，既不通过 CPU，也不需要 CPU 干预，以块为单位传送数据，每块产生一个中断。整个数据传输操作在 DMA 控制器的控制下进行的。CPU 除了在数据传输开始和结束时做一点处理外，在传输过程中 CPU 可以进行其他的工作。这样，在大部分时间里，CPU 和输入输出都处于并行操作。因此，使整个计算机系统的效率大大提高。

DMA 工作的四个步骤：由 IO 发出 DMA 请求；CPU 执行完当前周期后释放总线，DMA 响应；DMA 传输，CPU 执行内部操作；DMA 结束释放总线，同时向 IO 发出结束信号，IO 向 CPU 发出中断请求表示 IO 完成，数据通过总线传入 CPU 接受处理。



多处理器系统（parallel system）：包含多台功能相近的 CPU，处 CPU 之间彼此可以交换数据，所有处理器共享内存，I/O 设备，控制器，及外部设备，整个硬件系统由统一的操作系统控制，在处理器和程序之间实现作业、任务、程序、数组极其元素各级的全面并行。

三个优点：增加吞吐量、规模经济、增加可靠性

多道程序设计（multiprograming）：在计算机内存中同时多个作业，使它们在管理程序控制之下，相互穿插的运行。特征：多道、宏观上并行、微观上串行。

对称多处理（Symmetrical Multi-Processing）：在一个计算机上汇集了多个 CPU，并分享总线、内存和外部中断，将任务对称分布在多个 CPU 之上，负载均匀分布。

分时系统（time-sharing system）：由多个用户共享处理器时间，由操作系统控制每个用户程序以很短的时间为单位（时间片）交替执行。

双重模式操作：分为用户模式和特权模式，现代操作系统中用两个模式位表示当前模式，当计算机在执行用户程序时出于用户模式，当需要调用系统服务时，转为特权模式。

定时器（timer）：确保操作系统维持对 CPU 的控制，防止用户程序进入死循环且不将控制权返回到操作系统，将定时器设置为给定时间后中断计算机。

中断（interrupt）：指当出现需要时，CPU 暂时停止当前程序的执行转而执行处理新情况的程序和执行过程。

中断的处理过程为：关中断、保护现场、执行中断服务程序、恢复现场、开中断。
主要来源：外设请求、故障（除 0 溢出）、时钟请求、数据通道中断
中断向量表：所有中断服务程序的入口地址集合，中断向量的位置存放一条跳转到中断服务程序入口地址的跳转指令。
高速缓存（cache）：在 CPU 和主存之间的高速小容量存储器，与主存构成一级存储器。
Buffer 和 Cache 区别：buffer 是即将要被写入磁盘的，而 cache 是从磁盘中读出来的 buffer 是要保证高速与低速部件的吻合。

第二章
操作系统包括什么：用户界面、**程序执行、IO 操作、文件系统、通信、错误检测、**资源分配、统计、安全和保护
系统调用（system call）：操作系统在内核实现编写好一系列具备预定功能的函数，通过一组称为系统调用的接口呈现给用户。
传参方法：寄存器直接传参、寄存器传送参数块首地址、程序压栈操作系统出栈
类型：进程控制、文件管理、设备管理、信息维护、通信
通信模式（communication）：实现进程协作，生产者-消费者问题，在内存建立缓冲区工生产者写，消费者读。
内存共享（shared-memory system）：进程间建立共享内存区域以进行通信和数据传递，但是在保护和同步方面存在问题。
消息传递（message-passing system）：直接通信在进程间建立线路，明确发送者和接受者。间接通信使用邮箱，进程间至少要有一个公共邮箱才能进行通信。

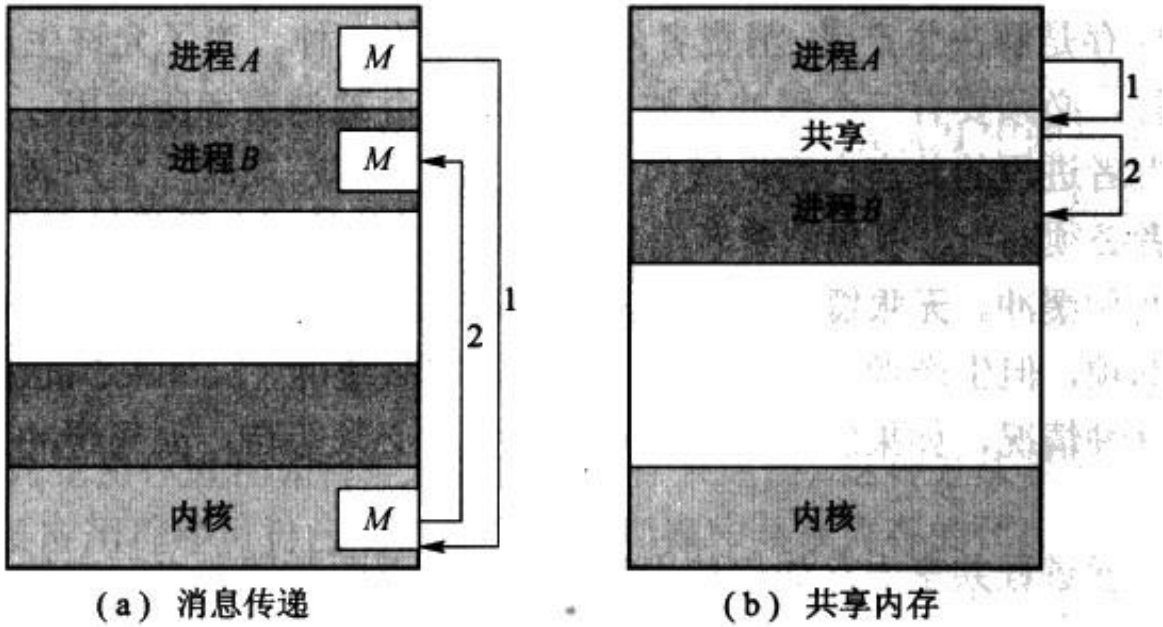


图 3.13 通信模型

机制与策略（mechanism&policy）：机制决定如何做，策略决定做什么，将这两者分开保证系统容易被修改，使操作系统更加灵活也更加轻巧。
操作系统结构：分层法优点在于构造和调试的简化，每层只可利用较低层的功能和服务，简化了调试和验证，从最底层开始向上调试，但是增加了数据参数的传送。

模块化内核使大部分操作在用户模式下进行，使之更安全更方便，利于扩充系统操作且安全性可靠性更高。微内核最主要的缺点是与进程间通信的过度联系和为了保证用户程序和系统服务相互作用而频繁使用操作系统的消息传递功能。

虚拟机（virtual machines）：通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。

优点是共享内存和文件更加安全简便、利于研究开发新的操作系统。

缺点是虚拟用户模式和虚拟内核模式的转换比较繁琐，磁盘系统的占用和使用，指令执行时间飘忽不定，与主机接口的协调。

第三章

进程（process）：进程是一个具有独立功能的程序关于某个数据集合的一次运行活动。它可以申请和拥有系统资源，是一个动态的概念，是一个活动的实体。它不只是程序的代码，还包括当前的活动，通过程序计数器、堆栈段、数据段的内容来表示。

进程控制块（PCB）：进程状态、程序计数器、CPU 寄存器、内存管理信息、IO 状态信息，是用来表示进程的一个单位。

进程状态：正在创建、执行、等待、就绪、终止

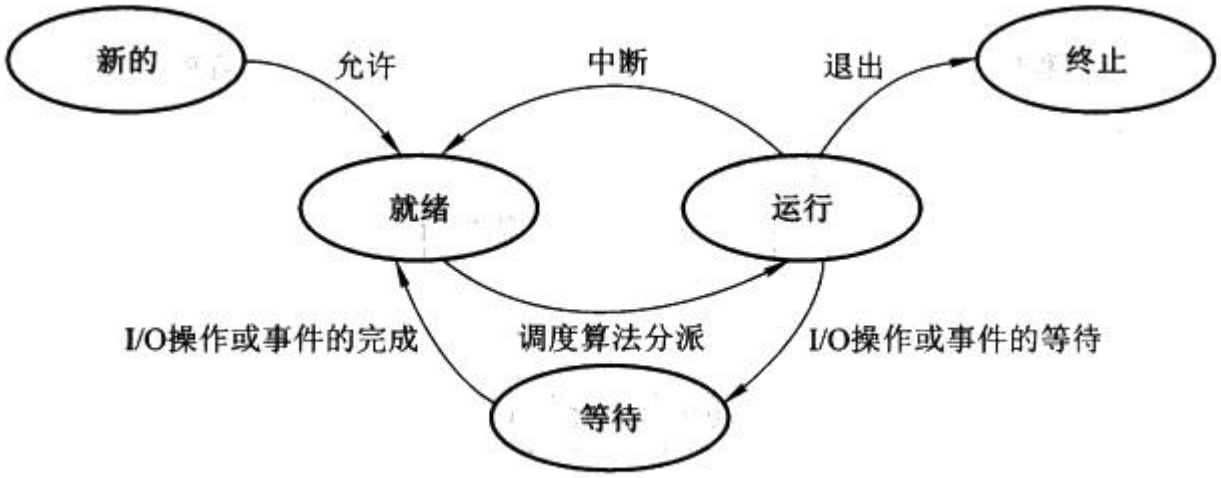


图 3.2　进程状态图

调度程序（schedulers）：分为短程调度、中程调度和长程调度，短程调度从内存中的就绪进程中选择，长程调度从进程池中选择，要将 CPU 为主和 IO 为主的进程组合好，使之达到平衡。

中程调度（mid-term scheduler）：决定处于交换区中的就绪进程中哪一个可以调入内存，以便直接参与对 CPU 的竞争。在内存资源紧张时，为了将进程调入内存，必须将内存中处于阻塞状态的进程调至交换区，以便为调入进程腾出空间。

上下文切换（context switch）：当进程产生中断时，系统用一个 PCB 保存 CPU 状态、寄存器状态等信息处理中断，中断处理结束后执行状态恢复重新运行进程。

第四章

线程（thread）：一个标准的线程由线程 ID，当前指令指针(PC)，寄存器集合和堆栈组成。另外，线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。线程也有就绪、阻塞和运行三种基本状态。具有响应度高、资源共享、经济的优点。

进程与线程区别：进程资源相互独立，线程资源共享；进程通信需要 IPC，线程可以直接写进程数据段；调度和上下文切换的速度。

线程模型（thread model）：多对一、一对一、多对多，一个内核一次只能处理一个线程，未增加并发性，但是提高了灵活性。

第五章

进程调度（process scheduling）：

- 先到先服务（FCFS）：FIFO 队列实现，非抢占短程调度，进程顺序对等待平均等待时间影响很大。
- 最短作业优先（SJF）：最佳调度策略，但是无法预测下一个 CPU 区间，通过大量的历史数据可以做大概的预测。
- 优先级调度（priority）：引发无穷阻塞，造成低优先级进程永不被执行，可以用老化来解决。
- **轮转法调度（RR scheduling）：**类似于 FCFS 调度，由时间片（time quantum）决定每个进程每次运行多久。

第六章

临界区（critical-section）：多个进程中代码块共享同一数据或资源，并需要并发地对这些数据进行操作，会产生 RC 竞态条件，就发生了临界区问题。

同步（synchronization）：使多个进程操作的执行在时序和空间上存在某种约束。

硬件同步：基于锁的实现，对于单内核，在修改共享变量时关中断即可；对于多内核的并行系统，可以通过加锁，结合特殊硬件指令使程序原子地执行。

自旋锁的优势：进程在等待锁的过程中不进行上下文切换，因为上下文切换可能非常耗时，所以如果等待时间短，那么用自旋锁能够节约时间。

信号量（semaphore）：是操作系统内核定义的一种数据结构，是一种封装行很强的受保护对象，由一个整数变量与 P、V、init 操作组成，且信号量只能够通过 PV 操作进行修改，通忙等待或者阻塞型两种实现方式达到进程同步。

饥饿（starvation）：饥饿 ，与死锁非常相似。指一些进程永远得不到资源，但不是死锁进程。饥饿有可能在优先级调度算法中出现，低优先级的进程永远得不到资源，可以通过先来先服务资源分配策略来避免。

第七章

死锁（dead lock）：两个或两个以上的进程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

四个必要条件：

- 互斥，存在一个不被共享的资源，一次只允许一个进程使用如果有另一进程申请必须等待到当前进程结束。

- 占有并等待：一个进程占有一个资源并在等待另外一个资源。
- 非抢占：资源不能被抢占，只能在进程结束后自动释放。
- 循环等待：一组等待的进程形成了环路。

第八章

基地址寄存器（base register）：含有最小合法物理地址。

界限地址寄存器（limit register）：决定了可访问的物理地址范围。

只有操作系统可以修改

逻辑地址与物理地址：逻辑地址是访问内存指令或用户程序给出的相对地址，物理地址是内存中的实际地址；逻辑地址空间受到地址线根数限制，物理地址空间受内存大小影响。

指令与数据绑定：编译时绑定，在编译时生成绝对代码向后扩展，首地址即是进程的内存地址；加载时绑定，生成可重定位代码，将指令和数据加载到内存时绑定；执行时绑定，进程操作时可以从一个内存段移到另一内存段，绑定必须在执行时才能进行。

内存管理单元（MMU）：将虚拟地址映射到物理地址，重定位寄存器加上 CPU 生成的虚拟地址形成物理地址。

动态加载（dynamic loading）：子程序保存在磁盘上，需要运行时才加载到内存，需要加载时重定位加载需要的程序，提高内存使用率。

交换、滚入滚出（swapping）：内存中的进程不执行时调出到备份存储上，需要执行时再加载，交换后的进程位置由加载方式决定，如果是编译时加载则回到原内存区域中。

连续内存分配（contiguous memory allocation）：操作系统底层驻留，用户进程高层存储。

动态内存分配：

- 首次适应（first fit）：将内存中第一个足够大的孔分配给进程。
- 最佳适应（best fit）：将内存中最小的足够大的孔分配给进程。
- **最差适应（worst fit）**：将内存中最大的孔分配给进程。

外部碎片（external fragmentation）：内存中小段的可用空间，当所有可用内存只和可以满足进程请求，但并不连续时，就出现了外部碎片问题。

内部碎片（internal fragmentation）：以固定内存单元分配内存时，所分配的内存大于进程所需要内存，内存单元减进程内存需求即是内部碎片。

分页技术：为每一个进程维护一个页表，给出了进程每一页对应的帧的位置。帧表用来记录每一帧的状态，被哪个进程的哪一页占用。对用户来说不可见，需要地址映射。

- **地址转换**：逻辑地址映射到物理地址时，物理地址=页号*页容量+页内偏移量。
- **碎片问题**：不产生外部碎片，但是几乎肯定有内部碎片，因为帧大小固定。
- **保护措施**：在页表内为每一页设置一个有效-无效位，如果有效，则表示进程页在逻辑内存中，如果访问无效的页，则产生错误中断。
- **页共享**：支持多用户，同一进程的只需要维护一个页表，不需要为 N 个用户维护 N 个。

分段（segmentation）：将进程的代码段和相关数据划分成一组段，对于用户来说是可见的，采用短号和段内偏移量映射到物理地址，缺点是程序员必须清楚段长度的最大限度。

分段与分页区别：分页是为了实现进程的离散分配，提高内存利用率，分段是进程的逻辑单位含有一组连续的完整意义内容；页大小固定而段大小不一；作业地址一二维之分。

多级页表：将页表再分成页表的页表。

第九章

虚拟内存（virtual memory）：在主存中可只装入最近经常要访问的某些区域的指令和数据，剩余部分就暂时不必装入，等到以后要访问到它们时再调入内存。如果主存较紧张，必要时可将已不大访问的信息调出内存，再执行调入操作。由于作业的指令和数据可以存放在外存

中，用户的程序就不受实际内存大小的限制，好像计算机系统向用户系统提供了容量极大的“主存”，而这个大容量的“主存”是靠存储管理的软件和硬件通过大容量的辅存作为后援存储器扩充而获得的，是程序设计员感觉到的，而实际上并不存在的存储器，故称虚拟存储器。

按需调页（page demanding）：对单个进程而言，只有在需要某些页时才将这些页调入内存与交换相比是在进程内操作，而交换是在进程间操作。

缺页中断：检查进程页表，确定引用是否合法；非法引用则终止进程，引用有效但未调入内存则调入；找到空闲帧；调度磁盘，将页调入空闲帧；修改进程内部表和页表，表示已经调入；重新开始因陷阱而中断的指令。

有效访问时间=（1-页错误率）*内存访问时间+页错误率*页错误处理时间

三个主要的页错误处理时间，中断处理，读入页和重启进程。

页面置换算法：

- FIFO：置换时每次都置换最旧的一页。Belady 异常，对于有些页置换算法，页错误率会随着分配帧增加而增加。
- 最优置换（optimal）：对于已有帧的页码，替换页调用序列中距离当前操作页最远的页。但是这个算法因为无法知道后续的序列所以难以实现，只是用于与其他算法作比较测试。
- 近期最少使用法（LRU）：对于已有帧的页码，替换当距离当前页最远的页（向前）。

系统颠簸（thrashing）：CPU 处理多道程序时，内存没有足够的帧供多个进程同时运行，导致需要不断置换页，连续产生页错误，使页置换时间大于执行时间。

工作集模型（working-set model）：基于局部性假设的用于研究一个进程实际正使用多少帧的模型，定义一个△表示最近使用的△个页的集合作为工作集。

第十章

文件（file）：记录在外存上相关信息的具有名称的合集，是逻辑外存的最小分配单元。基本操作有创建、读写文件、删除文件、文件重定位和截短文件。

访问方式：

- **顺序访问（sequential access）：**按照顺序一个记录接着一个记录处理，不仅适用于顺序访问设备，并且适用于随机访问设备。
- 直接访问（direct access）：文件由固定长度的逻辑记录组成，允许程序按任意顺序进行读写，是基于文件的磁盘模型。

目录操作：搜索文件、创建文件、删除文件、遍历目录、重命名文件和跟踪文件系统。

目录结构（directory structure）：

- **单层目录（single-level）：**所有文件都包含在同一目录中，优点是便于理解和支持，缺点是文件名难以被用户记住，无法较好解决名称冲突。
- **双层结构（two-level）：**为每个用户创建独立目录，称为主文件目录（MFD），每个用户有用户文件目录（UFD），解决名称冲突，但是隔离了用户，不利于共享。
- **树状结构目录（tree-structured）：**可以看作两层结构目录的扩展，可以扩展为任意高度的树，允许用户创建自己的子目录组织文件。

第十一章

虚拟文件系统（VFS）：通过定义清晰的 VFS 接口将文件系统的通用操作和具体实现分开，并为每种类型的对象都定义了一组必须实现的操作通过函数表进行指向。

文件系统结构：将多种文件系统整合成为一个目录结构。

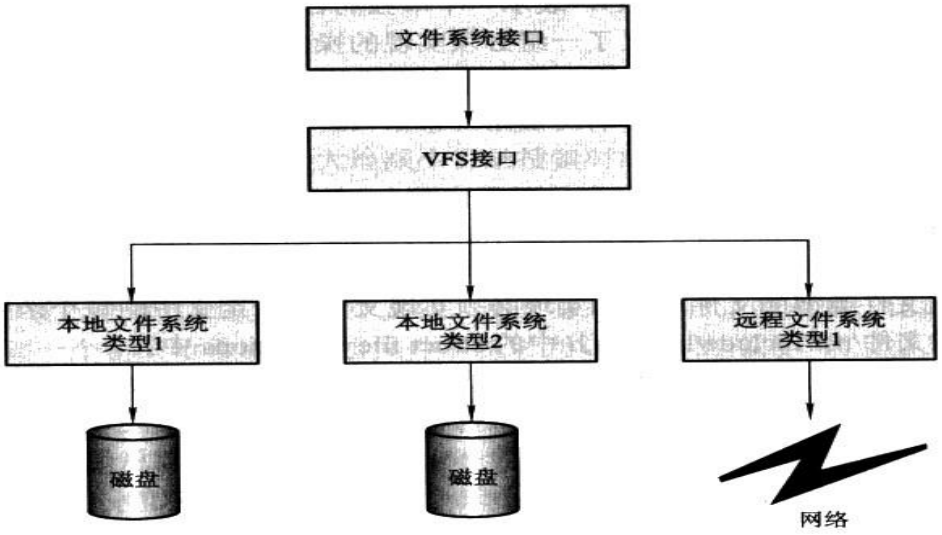


图 11.4　虚拟文件系统示意图

分配方法（allocation methods）：

- 连续分配（contiguous）：每个文件占用连续的内存单元，优点是访问简单而且支持直接访问，缺点是会产生动态内存分配问题，即碎片，以及文件的可扩展性太差。
- 链接分配（linked）：用链表连接起文件的各个块。优点是没有外部碎片，只要有空闲快就能够扩展文件，而不需要合并磁盘空间；缺点是不支持直接访问且指针需要占用空间，指针的可靠性不强。
- 索引分配（indexed）：将连接分配中每个文件的所有指针放在一起组成索引块，类似于页表。优点是支持直接访问且没有外部碎片问题，但是需要索引块需要内存空间，且比链接分配的指针效率要低。

空闲空间管理（free-space management）：

- 位向量（bit vector）：每个块都用一个二进制位表示其是否空闲，有多少块就位图就有多大。
- 链表（linked list）：将所有空闲快用链表连接。
- 组和计数（group&count）

网络文件系统（NFS）： NFS 允许一个系统在网络上与他人共享目录和文件。通过使用 NFS，用户和程序可以像访问本地文件一样访问远端系统上的文件。优点在于本地工作站使用更少磁盘空间，因为数据可以通过网络在别的机器上访问；不必再每个用户机器建立一个 home 目录。

第十二章

磁盘调度（disk scheduling）： SSTF 和 CLOOK 最常用，因为最平衡

- 先来先服务（FCFS）：比较公平，但是访问空间的跨度很大。
- 最短寻道时间优先（SSTF）：可以大大减少寻道时间，但是由于请求是随时到达的，所以可能有些请求一直得不到服务。
- SCAN 算法（电梯算法 elevator algorithm）：从第一个服务开始先向正方向扫描，处理完路上所有请求，然后反向扫描处理剩下的请求。
- C-SCAN 算法：提供更均匀的等待时间，当磁头扫描到尾部的时候立即返回磁盘起始点再扫描。
- LOOK 调度&SCAN 算法，C-LOOK 调度&C-SCAN 算法，朝某方向扫描到最远的请求即返回。

第十三章

端口（port）：CPU 通过接口寄存器或特定电路与外设进行数据传送，这些寄存器或特定电路称之为端口。

总线（bus）：总线是一种内部结构，它是 cpu、内存、输入、输出设备传递信息的公用通道，主机的各个部件通过总线相连接，外部设备通过相应的接口电路再与总线相连接，计算机各种功能部件之间传送信息的公共通信干线。

控制器（controller）：按照预定顺序改变主电路或控制电路的接线和改变电路中电阻值来控制电动机的启动、调速、制动和反向的主令装置。