操作系统（Operating System|OS）：配置在计算机硬件上的第一层软件，对硬件系统的首次扩充。

操作系统的目标

|  |  |
| --- | --- |
| 方便性 | 用户易学易用。 |
| 有效性 | 提高系统资源的利用率（资源）。  提高系统的吞吐量（程序）。 |
| 可扩充性 | （微内核结构具有良好的可扩充性。） |
| 开放性 | 系统遵循世界标准规范。 |

操作系统的作用

|  |  |
| --- | --- |
| 作为用户与计算机硬件系统之间的接口 | OS处于用户与计算机硬件系统之间，用户通过OS来使用计算机系统。（用户在OS帮助下能够方便、快捷、可靠地操纵计算机硬件和运行自己的程序。）  命令方式、系统调用方式、图标-窗口方式 |
| 作为计算机系统资源的管理者 | 处理机、存储器、I/O设备、文件（数据和程序） |
| 实现对计算机资源的抽象 | 裸机：一台完全无软件的计算机系统。  硬件接口（物理接口）：裸机向用户提供的接口。  扩充机器（虚机器）：覆盖了I/O设备管理软件的机器。 |

推动操作系统发展的主要动力

|  |  |
| --- | --- |
| 不断提高计算机资源利用率 |  |
| 方便用户 |  |
| 器件的不断更新换代 |  |
| 计算机体系结构的不断发展 |  |
| 不断提出新的应用需求 |  |

操作系统的发展过程

|  |  |  |  |
| --- | --- | --- | --- |
| 未配置操作系统的计算机系统 | 人工操作方式 | 由程序员将事先已穿孔的纸带（或卡片），装入纸带输入机（或卡片输入机），再启动它们将纸带（或卡片）上的程序和数据输入计算机，然后启动计算机运行。仅当程序运行完毕并取走计算结果后，才允许下一个用户上机。  纸带/卡片、纸带输入机/卡片输入机  缺点   |  | | --- | | 人机矛盾：人工操作方式严重降低计算机资源的利用率。 | |
| 脱机输入/输出（Off-Line I/O）方式 | 事先将装有用户程序和数据的纸带装入纸带输入机，在一台外围机的控制下，把纸带（卡片）上的数据（程序）输入到磁带上。当CPU需要这些程序和数据时，再从磁带上高速地调入内存。当CPU需要输出时，可先由CPU把数据直接从内存高速地输送到磁带上，然后在另一台外围机的控制下，再将磁带上的结果通过相应的输出设备输出。  外围机  优点   |  | | --- | | 减少了CPU的空闲时间。 | | 提高了I/O速度。 |   联机输入/输出（On-Line I/O）方式：在主机的直接控制下进行输入/输出的方式。 |
| 单道批处理系统（Simple Batch Processing System） | | 先把一批作业以脱机方式输入到磁带上，并在系统中配上监督程序性（Monitor），在它的控制下，使这批作业能一个接一个地连续处理。  监督程序（Monitor）  缺点   |  | | --- | | 系统中的资源得不到充分的利用（CPU等待I/O。内存。多种类型I/O设备）。 | |
| 多道批处理系统（Multiprogrammed Batch Processing System） | | 用户所提交的作业先存放在外存上，并排成一个队列，然后由作业调度程序按一定的算法，从后备队列中选择若干个作业调入内存，使它们共享CPU和系统中的各种资源。  后备队列：在多道批处理系统中，用户所提交的作业先存放在外存上，并排成一个队列。  优点   |  | | --- | | 资源利用率高（CPU、内存、I/O设备）。 | | 系统吞吐量大。 |   缺点   |  | | --- | | 平均周转时间长。 | | 无交互能力。 |   问题   |  | | --- | | 处理机争用问题 | | 内存分配和保护问题 | | I/O设备分配问题 | | 文件的组织和管理问题 | | 作业管理问题 | | 用户与系统的接口问题 |   软件   |  | | --- | | 能有效地组织和管理四大资源的软件 | | 合理地对各类作业进行调度和控制它们运行的软件 | | 方便用户使用计算机的软件 |   操作系统：一组能有效地组织和管理计算机硬件和软件资源，合理地对各类作业进行调度，以方便用户使用的程序的集合。 |
| 分时系统（Time Sharing System） | | 在一台主机上连接了多个配有显示器和键盘的终端并由此所组成的系统。  用户需求   |  | | --- | | 人-机交互 | | 共享主机 |   问题   |  | | --- | | 及时接收  多路卡 | | 及时处理   |  | | --- | | 作业直接进入内存。 | | 采用轮转运行方式。  时间片：一段很短的时间。 | |   和多道批处理系统不同点   |  | | --- | | 多路性：系统允许将多台终端同时连接到一台主机上，并按分时原则为每个用户服务。 | | 独立性：系统提供了这样的用机环境，即每个用户在各自的终端上进行操作，彼此之间互不干扰，给用户的感觉就像是他一人独占主机进行操作。 | | 及时性：用户的请求能在很短时间内获得相应。 | | 交互性：用户可通过终端与系统进行广泛的人机对话。 | |
| 实时系统（Real Time System） | | 系统能及时响应外部事件的请求，在规定的时间内完成对该事件的处理，并控制所有实时任务协调一致地运行。  实时计算：系统的正确性，不仅由计算的逻辑结果来确定，而且还取决于产生结果的时间的一类计算。  实时系统的类型   |  | | --- | | 工业（武器）控制系统 | | 信息查询系统 | | 多媒体系统 | | 嵌入式系统 |   实时任务的类型   |  | | --- | | 周期性实时任务：外部设备周期性地发出激励信号给计算机，要求它按指定周期循环执行，以便周期性地控制某外部设备的一类任务。 | | 非周期性实时任务。  截止时间（最后期限，Deadline）   |  | | --- | | 开始截止时间：某任务在某时间以前必须开始执行。 | | 完成截止时间：某任务在某时间以前必须完成。 | |  |  | | --- | | 硬实时任务（Hard Real-time Task|HRT）：系统必须满足任务对截止时间的要求，否则可能出现难以预测的后果。 | | 软实时任务（Soft Real-time Task|SRT）：联系着截止时间，但并不严格，若偶尔错过了任务的截止时间，对系统产生的影响也不会太大。 |   和分时系统比较   |  | | --- | | 多路性 | | 独立性 | | 及时性 | | 交互性 | | 可靠性 | |

微机操作系统：配置在微型机上的操作系统。

微机操作系统按运行方式分类

|  |  |
| --- | --- |
| 单用户单任务操作系统 | 只允许一个用户上级，且只允许用户程序作为一个任务运行。  CP/M、MS-DOS（Disk Operating System） |
| 单用户多任务操作系统 | 只允许一个用户上机，但允许用户把程序分为若干个任务，使它们并发执行，从而有效地改善了系统的性能。  Windows |
| 多用户多任务操作系统 | 允许多个用户通过各自的终端，使用同一台机器，共享主机系统中的各种资源，而每个用户程序又可进一步分为几个任务，使它们能并发执行，从而可进一步提高资源利用率和系统吞吐量。  UNIX OS、Xenix、Solaris OS、Linux OS |

操作系统的基本特性（多道批处理系统、分时系统、实时系统）

|  |  |
| --- | --- |
| 并发（Concurrence） | 并行性：两个或多个事件在同一时刻发生。  并发性：两个或多个事件在同一时间间隔内发生。在多到程序环境下，在一段时间内宏观上有多个程序在同时运行。  进程（Process）：在系统中能独立运行并作为资源分配的基本单位，它是由一组机器指令、数据和堆栈组成的，是一个能独立运行的活动实体。 |
| 共享（Sharing） | （OS环境下的）资源共享（资源复用）：系统中的资源可供内存中多个并发执行的进程共同使用。  资源属性/实现资源共享的主要方式   |  |  | | --- | --- | | 互斥共享方式 | 互斥式共享：当进程A要访问某资源时，必须先提出请求。若此时该资源空闲，系统便可将之分配给请求进程A使用。此后若再有其它进程也要访问该资源，只要A未用完就必须等待。仅当A进程访问完并释放系统资源后，才允许另一进程对该资源进行访问。  临界资源（独占资源）：在一段时间内只允许一个进程访问的资源。 | | 同时访问方式 |  | |
| 虚拟（Virtual） | 虚拟：在OS中，通过某种技术将一个物理实体变为若干个逻辑上的对应物的功能。  虚拟技术：用于实现虚拟的技术。   |  |  |  | | --- | --- | --- | | 时分复用技术   |  | | --- | | 虚拟处理机技术：利用多道程序设计技术，为每道程序建立至少一个进程，让多道程序并发执行。  虚拟处理器：利用多道程序设计技术，将一台物理上的处理机虚拟为多台逻辑上的处理机，在每台逻辑处理机上运行一道程序，用户所感觉到的处理机。 | | 虚拟设备技术：通过分时复用的方法，将一台物理I/O设备虚拟为多台逻辑上的I/O设备，并允许每个用户占用一台逻辑上的I/O设备。  临界资源：仅允许在一段时间内由一个用户访问的设备。 | | | 空分复用技术：利用存储器的空闲时间分区域存放和运行其它的多道程序，以此来提高内存的利用率。  虚拟存储技术：实现内存的分时复用。 | |
| 异步（Asynchronism） | 用户程序性能分类   |  | | --- | | 侧重于计算而较少需要I/O | | 计算少而I/O多 |   进程的异步性：进程是以人们不可预知的速度向前推进的。 |

操作系统的主要功能

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 处理机管理功能 | 处理机管理的主要功能：创建和撤销进程，对诸进程的运行进行协调，实现进程之间的信息交换，以及按照一定的算法把处理机分配给进程。   |  | | --- | | 进程控制  进程控制的主要功能：为作业创建进程、撤销（终止）已结束的进程，以及控制进程在运行过程中的状态转换。 | | 进程同步  协调方式   |  | | --- | | 进程互斥方式：诸进程在对临界资源进行访问时，应采用互斥方式。 | | 进程同步方式：在相互合作去完成共同任务的诸进程间，由同步机构对它们的执行次序加以协调。 | | | 进程通信  进程通信的任务：实现相互合作的进程之间的信息交换。 | | 调度   |  | | --- | | 作业调度  作业调度的基本任务：从后备队列中按照一定的算法选择出若干个作业，为它们分配运行所需的资源，在将这些作业调入内存后，分别为它们建立进程，使它们都成为可能获得处理机的就绪进程，并将它们插入就绪队列中。 | | 进程调度  进程调度的任务：从进程的就绪队列中按照一定的算法选出一个进程，将处理机分配给它，并为它设置运行现场，使其投入运行。 | | |
| 存储器管理功能 | 存储器管理的主要任务：为多道程序的运行提供良好的环境，提高存储器的利用率，方便用户使用，并能从逻辑上扩充内存。   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 内存分配  内存分配的主要任务   |  | | --- | | 为每道程序分配内存空间，使它们“各得其所”。 | | 提高存储器的利用率，尽量减少不可用的内存空间（碎片）。 | | 允许正在运行的程序申请附加的内存空间，以适应程序和数据动态增长的需要。 |   实现内存分配的方式   |  | | --- | | 静态分配方式：每个作业的内存空间是在作业装入时确定的，在作业装入后的整个运行期间不允许该作业再申请新的内存空间，也不允许作业在内存中“移动”。 | | 动态分配方式：每个作业所要求的基本内存空间虽然也是再装入时确定的，但允许作业在运行过程中继续申请新的附加内存空间。以适应程序和数据的动态增长，也允许作业在内存中“移动”。 | | | 内存保护  内存保护的主要任务   |  | | --- | | 确保每道用户程序都仅在自己的内存空间内运行，彼此互不干扰。 | | 决不允许用户程序访问操作系统的程序和数据，也不允许用户程序转移到非共享的其它用户程序中去执行。 | | | 地址映射：将地址空间中的逻辑地址转换为内存空间中与之对应的物理地址。 | | 内存扩充   |  | | --- | | 请求调入功能：系统允许在仅装入部分用户程序和数据的情况下，便能启动该程序运行。在程序运行过程中，若发现要继续运行时所需的程序和数据尚未装入内存，可向OS发出请求，由OS从磁盘中将所需部分调入内存，以便继续运行。 | | 置换功能：若发现在内存中已无足够的空间来装入需要调入的程序和数据时，系统应能将内存中的一部分暂时不用的程序和数据调至硬盘上，以腾出内存空间，然后再将所需调入的部分装入内存。 | | |
| 设备管理功能 | 设备管理的主要任务   |  | | --- | | 缓冲管理 | | 设备分配  设备分配的基本任务：根据用户进程的I/O请求、系统现有资源情况以及按照某种设备分配策略，为之分配其所需的设备。如果在I/O设备和CPU之间还存在着设备控制器和I/O通道，则还需为分配出去的设备分配相应的控制器和通道。  独占设备 | | 设备处理  设备处理程序（设备驱动程序）  设备处理程序的基本任务：实现CPU和设备控制器之间的通信，即由CPU向设备控制器发出I/O命令，要求它完成指定的I/O操作；反之，由CPU接收从控制器发来的中断请求，并给予迅速的响应和相应的处理。 | |
| 文件管理功能 | 文件管理的主要任务   |  | | --- | | 文件存储空间的管理  主要任务：为每个文件分配必要的外存空间，提高外存的利用率，进而提高文件系统的存、取速度。 | | 目录管理  主要任务：为每个文件建立一个目录项，目录项包括文件名、文件属性、文件在磁盘上的物理位置等，并对众多的目录项加以有效的组织，以实现方便的按名存取。还实现文件共享。  按名存取：用户只需提供文件名，即可对文件进行存取。 | | 文件的读写管理：根据用户的请求，从外存中读取数据，或将数据写入外存。 | | 文件保护  文件保护的目标   |  | | --- | | 防止未经核准的用户存取文件。 | | 防止冒名顶替存取文件。 | | 防止以不正确的方式使用文件。 | | |
| 操作系统与用户之间的接口 | |  |  |  |  | | --- | --- | --- | --- | | 用户接口   |  | | --- | | 联机用户接口：为联机用户提供的，由一组键盘操作命令及命令解释程序组成。 | | 脱机用户接口：为批处理作业的用户提供的。  作业控制语言（Job Control Language|JCL） | | 图形用户接口  图标（icon） | | | 程序接口：为用户程序在执行中访问系统资源而设置的，是用户程序取得操作系统服务的唯一途径。  系统调用：一个能完成特定功能的子程序。 | |

现代操作系统的新功能

|  |
| --- |
| 系统安全 |
| 网络的功能和服务 |
| 支持多媒体 |

操作系统的结构

|  |
| --- |
| 传统结构的OS：早期无结构的OS（第一代）、模块化结构的OS（第二代）和分层式结构的OS（第三代）。 |
| 现代结构的OS：微内核结构的OS。 |

|  |  |
| --- | --- |
| 无结构操作系统（整体系统结构） | 为数众多的一组过程的集合，每个过程可以任意地相互调用其它过程，致使操作系统内部既复杂又混乱。 |
| 模块化结构的OS | 模块-接口法（无序模块法）  衡量模块独立性标准   |  | | --- | | 内聚性：模块内部各部分间联系的紧密程度。 | | 耦合度：模块间相互联系和相互影响的程度。 |   优点   |  | | --- | | 提高OS设计的正确性、可理解性和可维护性。 | | 增强OS的可适应性。 | | 加速OS的开发过程。 |   缺点   |  | | --- | | 在OS设计时，对各模块间的接口规定很难满足在模块设计完成后对接口的实际需求。 | | 在OS设计阶段，设计者必须做出一系列的决定（决策），每一个决定必须建立在上一个决定的基础上，但模块化结构设计中，各模块的设计齐头并进，无法寻找一个可靠的决定顺序，造成各种决定的“无序性”，这将使程序人员很难做到“设计中的每一步决定”都是建立在可靠的基础上。 | |
| 分层式结构的OS | 目标系统、裸机系统（宿主系统）  自底向上法  优点   |  | | --- | | 易保证系统的正确性。 | | 易扩充和易维护性。 |   缺点：系统效率降低。 |
| 客户/服务器模式（Client/Server Model|C/S模式） | 客户/服务器系统的组成   |  | | --- | | 客户机：一个自主计算机，具有一定的处理能力，客户进程在其上运行。 | | 服务器：通常是一台规模较大的机器，在其上驻留有网络文件系统或数据库系统等，它应能为网上所有的用户提供一种或多种服务。 | | 网络系统：是用于连接所有客户机和服务器，实现它们之间通信和网络资源共享的系统。 |   客户/服务器之间的交互：略  优点   |  | | --- | | 数据的分布处理和存储。 | | 便于集中管理。 | | 灵活性和可扩充性。 | | 易于改编应用软件。 |   缺点   |  | | --- | | 不可靠性：在系统仅有一个服务器时，一旦服务器故障，将导致整个网络瘫痪。 | | 瓶颈：当服务器在重负荷下工作时，会因忙不过来而显著地延长对用户请求的响应时间。 | |
| 面向对象的程序设计（Object-Orientated Programming） | 略 |
| 微内核（MicroKernel）结构 | 定义   |  |  |  |  | | --- | --- | --- | --- | | 足够小的内核  内核：精心设计的、能实现现代OS最基本核心功能的小型内核。   |  | | --- | | 与硬件处理紧密相关的部分 | | 一些较基本的功能 | | 客户和服务器之间的通信 | | | 基于客户/服务器模式 | | 应用“机制与策略分离”原理  机制：实现某一功能的具体执行机构。  策略：在机制的基础上借助于某些参数和算法来实现该功能的优化，或达到不同的功能目标。 | | 采用面向对象技术 |   微内核的基本功能   |  | | --- | | 进程（线程）管理 | | 低级存储器管理 | | 中断和陷入处理 |   优点   |  | | --- | | 提高了系统而可扩展性。改善系统的灵活性。 | | 增强了系统的可靠性。  应用程序接口（|API） | | 可移植性强。  硬件隐藏层 | | 提供了对分布式系统的支持。 | | 融入了面向对象技术。 |   缺点：运行效率降低。 |

程序的执行方式

|  |
| --- |
| 顺序执行：在内存中仅装入一道用户程序，由它独占系统中的所有资源，只有在一个用户程序执行完成后，才允许装入另一个程序执行。 |
| 并发执行 |

|  |
| --- |
| 前驱图（Precedence Graph）：一个有向无环图（Directed Acyclic Graph|DAG），它用于描述进程之间执行的先后顺序。图中的每个结点可用来表示一个进程或程序段，乃至一条语句，结点间的有向边则表示两个结点之间存在的偏序（Partial Order）或前驱关系（Precedence Relation）。 |
| 初始结点（Initial Node）：没有前驱的结点。 |
| 终止结点（Final Node）：没有后继的结点。 |
| 重量（Weight）：该结点所含有的程序量或程序的执行时间。 |

程序顺序执行时的特征

|  |  |
| --- | --- |
| 顺序性 | 处理机严格地按照程序所规定的顺序执行，即每一操作必须在下一个操作开始之前结束。 |
| 封闭性 | 程序在封闭的环境下运行，即程序运行时独占全机资源，资源的状态（除初始状态外）只有本程序才能改变它，程序一旦开始执行，其执行结果不受外界因素影响。 |
| 可再现性 | 只要程序执行时的环境和初始条件相同，当程序重复执行时，不论它是从头到尾不停顿地执行，还是“停停走走”地执行，都可获得相同的结果。 |

程序并发执行时的特征

|  |  |
| --- | --- |
| 间断性 | 程序在并发执行时，由于它们共享系统资源，以及为完成同一项任务而相互合作，致使这些并发执行的程序之间形成了相互制约的关系。 |
| 失去封闭性 | 当系统中存在着多个可以并发执行的程序时，系统中的各种资源将为它们所共享，而这些资源的状态也由这些程序来改变，致使其中任一程序在运行时，其环境都必然会受到其它程序的影响。 |
| 不可再现性 | 程序在并发执行时，由于失去了封闭性，也将导致其又失去可再现性。 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 进程的定义 | |  | | --- | | 进程控制块（Process Control Block，PCB）：为了使参与并发执行的每个程序（含数据）都能独立地运行，在操作系统中必须为之配置一个专门的数据结构，这个数据结构。 | | 进程实体（进程映像|进程）：程序段、相关的数据段和PCB。  创建进程：创建进程实体中的PCB。  撤销进程：撤销进程的PCB。 | | 进程是程序的一次执行。  进程是一个程序及其数据在处理机上顺序执行时所发生的活动。  进程是具有独立功能的程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。  传统OS中的进程：进程实体的运行过程，系统进行资源分配和调度的一个独立单位。 | |
| 进程的特征 | |  |  | | --- | --- | | 动态性 | 进程的实质是进程实体的执行过程，因此，动态性就是进程的最基本的特征。  程序是一组有序指令的集合，并存放于某种介质上，其本身并不具有活动的含义，因而是静态的。 | | 并发性 | 多个进程实体同存于内存中，且能在一段时间内同时运行。 | | 独立性 | 在传统的OS中，进程实体是一个能独立运行、独立获得资源和独立接受调度的基本单位。 | | 异步性 | 进程是按异步方式运行的，即按各自独立的、不可预知的速度向前推进。 | |

进程的三种基本状态

|  |  |
| --- | --- |
| 就绪（Ready）状态 | 进程已处于准备好运行的状态，即进程已分配到除CPU以外的所有必要资源后，只要再获得CPU，便可立即执行。  就绪队列：如果系统中有许多处于就绪状态的进程，通常将它们按一定的策略（如优先级策略）排成一个队列，该队列。 |
| 执行（Running）状态 | 进程已获得CPU，其程序正在执行的状态。 |
| 阻塞（Block）状态（进程的执行受到阻塞） | 正在执行的进程由于发生某事件（如I/O请求、申请缓冲区失败等）暂时无法继续执行的状态。  暂停状态（阻塞状态、等待状态、封锁状态）：此时引起进程调度，OS把处理及分配给另一个就绪进程，而让受阻进程处于暂停状态。  阻塞队列：通常系统将处于阻塞状态的进程排成一个队列，该队列。 |

|  |  |
| --- | --- |
| 创建状态 | 进程是由创建而产生。创建一个进程是个很复杂的过程，一般要通过多个步骤才能完成：如首先由进程申请一个空白PCB，并向PCB中填写用于控制和管理进程的信息；然后为该进程分配运行时所必需的资源；最后，把该进程转入就绪状态并插入就绪队列之中。但如果进程所需的资源尚不能得到满足，比如系统尚无足够的内存使进程无法装入其中，此时创建工作尚未完成，进程不能被调度运行，于是把此时进程所处的状态称为创建状态。 |
| 终止状态 | 当一个进程到达了自然结束点，或是出现了无法克服的错误，或是被操作系统所终结，或是被其他有终止权的进程所终结，它将进入中止状态。 |

挂起操作

静止状态：当挂起操作作用于某个进程是，该进程将被挂起，意味着此时该进程处于静止状态。

激活操作

引入挂起操作的原因

|  |
| --- |
| 终端用户的需要 |
| 父进程请求 |
| 负荷调节的需要 |
| 操作系统的需要 |

进程管理中的数据结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 资源信息表（进程信息表）：在计算机系统中，对于每个资源和每个进程都设置了一个数据结构，用于表征其实体。 | |  | | --- | | 内存表 | | 设备表 | | 文件表 | | 进程表（进程控制块|PCB） |   进程控制块（Process Control Block|PCB） |
| 进程控制块作用 | 使一个在多道程序环境下不能独立运行的程序（含数据）成为一个能独立运行的基本单位，一个能与其他进程并发执行的进程。   |  |  | | --- | --- | | 作为独立运行基本单位的标志 |  | | 能实现间断性运行方式 | CPU现场 | | 提供进程管理所需要的信息 | 资源清单 | | 提供进程调度所需要的信息 |  | | 实现与其它进程的同步与通信 |  | |
| 进程控制块中的信息 | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 进程标识符 | 用于唯一地标识一个进程。   |  |  | | --- | --- | | 外部标识符 | 为了方便用户（进程）对进程的访问，须为每一个进程设置一个外部标识符。  为了描述进程的家族关系，还应设置父进程标识及子进程标识。 | | 内部标识符 | 为了方便系统对进程的使用，在OS中又为进程设置了内部标识符，即赋予每一个进程一个唯一的数字标识符，它通常是一个进程的序号。 | | | 处理机状态 | 处理机状态信息（处理机上下文）：主要是由处理机的各种寄存器中的内容组成的。   |  |  | | --- | --- | | 通用寄存器（用户可视寄存器） |  | | 指令计数器 | 其中存放了要访问的下一条指令的地址。 | | 程序状态字PSW | 其中含有状态信息，如条件码、执行方式、中断屏蔽标志等。 | | 用户栈指针 | 每个用户进程都有一个或若干个与之相关的系统栈，用于存放过程和系统调用参数及调用地址。栈指针指向该栈的栈顶。 | | | 进程调度信息 | |  |  | | --- | --- | | 进程状态 | 指明进程的当前状态，它是作为进程调度和对换时的依据。 | | 进程优先级 | 用于描述进程使用处理机的优先级别的一个整数，优先级高的进程应优先获得处理机。 | | 进程调度所需的其它信息 | 与所采用的进程调度算法有关，比如，进程已等待CPU的时间总和、进程已执行的时间总和等。 | | 事件 | 进程由执行状态转变为阻塞状态所等待发生的时间，即阻塞原因。 | | | 进程控制信息 | 用于进程控制所必须的信息。   |  |  | | --- | --- | | 程序和数据的地址 | 进程实体中的程序和数据的内存或外存（首）地址，以便再调度到该进程执行时，能从PCB中找到其程序和数据。 | | 进程同步和通信机制 | 实现进程同步和进程通信时必需的机制，如消息队列指针、信号量等，它们可能全部或部分地放在PCB中。 | | 资源清单 | 在该清单中列出了进程在运行期间所需的全部资源（除CPU以外），另外还有一张已分配到该进程的资源的清单。 | | 链接指针 | 给出了本进程（PCB）所在队列中的下一个进程的PCB的首地址。 | | |
| 进程控制块的组织方式 | |  |  | | --- | --- | | 线性方式 | 将系统中所有的PCB都组织在一张线性表中，将该表的首址存放在内存的一个专用区域中。 | | 链接方式 | 把具有相同状态进程的PCB中的链接字链接成一个队列。  就绪队列、若干个阻塞队列、空白队列、空闲队列 | | 索引方式 | 系统根据所有进程状态的不同，建立几张索引表，例如，就绪索引表、阻塞索引表等，并把各索引表在内存的首地址记录在内存的一些专用单元中。  PCB表 | |

OS内核：现代操作系统一般将OS划分为若干层次，再将OS的不同功能分别设置在不同的层次中。通常将一些与硬件紧密相关的模块（如中断处理程序等）、各种常用的设备的驱动程序以及运行频率较高的模块（如时钟管理、进程调度和许多模块所公用的一些基本操作），都安排在紧靠硬件的软件层次中，将它们常驻内存。

这种安排方式的目的：

|  |
| --- |
| 便于对这些软件进行保护，防止遭受其他应用程序的破坏。 |
| 可以提高OS的运行效率。 |

将处理机的执行状态分成系统态和用户态的目的：防止OS本身及关键数据（如PCB等）遭受到应用程序有意或无意的破坏。

|  |  |
| --- | --- |
| 系统态（管态、内核态） | 具有较高的特权，能执行一切指令，访问所有寄存器和存储区，传统的OS都在系统态运行。 |
| 用户态（目态） | 具有较低特权的执行状态，仅能执行规定的指令，访问指定的寄存器和存储区。 |

大多数OS内核包含的两大方面的功能

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 支撑功能 | 提供给OS其它众多模块所需要的一些基本功能，以便支撑这些模块工作。   |  |  | | --- | --- | | 中断处理 |  | | 时钟管理 |  | | 原语操作 | 原语（Primitive）：由若干条指令组成的，用于完成一定功能的一个过程。  原语与一般过程的区别：是“原子操作（Atomic Operation）”。  原子操作：一个操作中的所有动作要么全做，要么全不做。（它是一个不可分割的基本单位。） | |
| 资源管理功能 | |  |  | | --- | --- | | 进程管理 |  | | 存储器管理 |  | | 设备管理 |  | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 进程的创建 | 父进程（Parent Process）、子进程（Progency Process）：在OS中，允许一个进程创建另一个进程，通常把创建进程的进程称为父进程，而把被创建的进程称为子进程。  （在UNIX中）进程家族（组）：进程与其子孙进程共同组成一个进程家族（组）。  句柄  进程图（Process Graph）  祖先进程：创建父进程的进程。  进程家族的祖先（Ancestor）：进程树的根结点。  引起创建进程的事件   |  |  | | --- | --- | | 用户登录 |  | | 作业调度 |  | | 提供服务 |  | | 应用请求 |  |   进程的创建（Creation of Process）   |  | | --- | | 申请空白PCB，为新进程申请获得唯一的数字标识符，并从PCB集合中索取一个空白PCB。 | | 为新进程分配其运行所需的资源，包括各种物理和逻辑资源，如内存、文件、I/O设备和CPU时间等。  共享段 | | 初始化进程控制块。   |  | | --- | | 初始化标识信息，将系统分配的标识符和父进程标识符填入新PCB中。 | | 初始化处理机状态信息，使程序计数器指向程序的入口地址，使栈指针指向栈顶。 | | 初始化处理机控制信息，将进程的状态设置为就绪状态或静止就绪状态，对于优先级，通常是将它设置为最低优先级，除非用户以显式方式提出高优先级要求。 | | | 如果进程就绪队列能够接纳新进程，便将新进程插入就绪队列。 | |
| 进程的终止 | 引起进程终止（Termination of Process）的事件   |  |  | | --- | --- | | 正常结束 | 表示进程的任务已经完成，准备退出运行。 | | 异常结束 | 进程在运行时发生了某种异常事件，使程序无法继续运行。  常见的异常事件：   |  |  | | --- | --- | | 越界错 |  | | 保护错 |  | | 非法指令 |  | | 特权指令错 |  | | 运行超时 |  | | 等待超时 |  | | 算数运算错 |  | | I/O故障 |  | | | 外界干预 | 进程应外界的请求而终止运行。   |  |  | | --- | --- | | 操作员或操作系统干预 |  | | 父进程请求 |  | | 父进程终止 |  | |   进程的终止过程   |  | | --- | | 根据被终止进程的标识符，从PCB集合中检索出该进程的PCB，从中读出该进程的状态。 | | 若被终止进程正处于执行状态，应立即终止该进程的执行，并置调度标志为真，用于指示该进程被终止后应重新进行调度。  调度标志 | | 若该进程还有子孙进程，还应将其所有子孙进程也都予以终止，以防它们成为不可控的进程。 | | 将被终止进程所拥有的全部资源或者归还给其父进程，或者归还给系统。 | | 将被终止进程PCB从所在队列（或链表）中移出，等待其它程序来搜集信息。 | |
| 进程的阻塞与唤醒 | 引起进程阻塞和唤醒的事件：   |  |  | | --- | --- | | 向系统请求共享资源失败。 |  | | 等待某种操作的完成。 |  | | 新数据尚未到达。 |  | | 等待新任务的到达。 |  |   进程阻塞过程  阻塞是进程自身的一种主动行为。   |  | | --- | | 由于该进程还处于执行状态，所以应先立即停止执行，把进程控制块中的现行状态由“执行”改为阻塞，并将PCB插入阻塞队列。  现行状态 | | 转调度程序进行重新调度，将处理机分配给另一就绪进程，并进行切换，亦即，保留被阻塞进程的处理机状态，按新进程的PCB中的处理机状态设置CPU的环境。 |   进程唤醒过程   |  | | --- | | 把被阻塞的进程从等待该事件的阻塞队列中移出，将其PCB中的现行状态由阻塞改为就绪。 | | 将该PCB插入到就绪队列中。 | |
| 进程的挂起与激活 | 进程挂起过程   |  | | --- | | 检查被挂起进程的状态，若处于活动就绪状态，便将其改为静止就绪；对于活动阻塞状态的进程，则将之改为静止阻塞。 | | 为了方便用户或父进程考查该进程的运行情况，而把该进程的PCB复制到某指定的内存区域。 | | 若被挂起的进程正在执行，则转向调度程序重新调度。 |   进程激活过程   |  | | --- | | 将进程从外存调入内存，检查该进程的现行状态，若是静止就绪，便将之改为活动就绪；若为静止阻塞，便将之改为活动阻塞。假如采用的是抢占调度策略，则每当有静止就绪进程被激活而插入就绪队列时，便应检查是否要进行重新调度，即由调度程序将被激活的进程与当前进程两者的优先级进行比较，如果被激活进程的优先级低，就不必重新调度；否则，立即剥夺当前进程的运行，把处理机分配给刚刚被激活的进程。 | |

在单处理机系统中的进程同步机制

进程同步机制的主要任务：对多个相关进程在执行次序上进行协调，使并发执行的诸进程之间能按照一定的规则（或时序）共享系统资源，并能很好地相互合作，从而使程序的执行具有可再现性。

|  |  |
| --- | --- |
| 间接相互制约关系 | 对于像打印机、磁带机这样的临界资源，必须保证多个进程对之只能互斥地访问，由此，在这些进程间形成了源于对该类资源共享的所谓间接相互制约关系。 |
| 直接相互制约关系 | 进程间的直接相互制约关系就是源于它们之间的相互合作。 |

进程的异步性：在多道程序环境下，由于存在着上述两类相互制约关系，进程在运行过程中是否能获得处理及运行与以怎样的速度运行，并不能由进程自身所控制。

与时间有关的错误：由此会产生对共享变量或数据结构等资源不正确的访问次序，从而造成进程每次执行结果的不一致。这种差错往往与时间有关。

临界资源（Critical Resource）

|  |  |
| --- | --- |
| 临界区（critical section） | 在每个进程中访问临界资源的那段代码。 |
| 进入区（entry section） | 每个进程在进入临界区之间，应先对欲访问的临界资源进行检查，看它是否正被访问。因此，必须在临界区前面增加一段用于进行上述检查的代码，这段代码。 |
| 退出区（exit section） | 用于将临界区正被访问的标志恢复为未被访问的标志。 |
| 剩余区 | 进程中除进入区、临界区及退出区之外的其它部分的代码。 |

同步机制应遵循的规则（准则）

|  |  |
| --- | --- |
| 空闲让进 |  |
| 忙则等待 |  |
| 有限等待 |  |
| 让权等待 |  |