行界面它采用文本命令、并用某一方法输入 另一种是批处理界面, 命令以及控制这 些命令的指令可以编成文件以便执行。 最为常用的是图形用户界面 程序执行:系统应能加载程序到内存,并加以运行。程序应能结束执行,包括正 或不正常 I/O 操作: 程序运行可能需要 I/O, 这些 I/O 可能涉及文件或设备 操作系统的服务 文件系统操作,错误检测,资源分配,记账,保护与安全 通信:通信实现可以通过共享内存 (shared memory) (两个或多个进程读 写共享内存区域),也可以通过消息交换 (message passing) 对于具有多个可选命令解释程序的系统,解释程序称为外壳 (shell)。 命令解释程序的主要功能是,获取并执行用户指定的下一条命令。这层提供了许 命令解释程序 多命令来操作文件: 创建、删除、列出、打印、复制、执行等。 一种方法是,命令解释程序本身包含代码以执行这些命令。另一种方法是通过系统 程序实现大多数的命令 利用桌面 概念, 即采用基于鼠标的视窗和菜单系统。 用户移动鼠标, 定位指针到 屏幕上的图标, 而这些图标代表程序、 文件、 目录和系统功能。能手机和手持平板 用户与操作系统的界面 图形用户界面 电脑通常采用触摸屏界面。 这样, 用户交互就是在触摸屏上做手势 管理计算机的系统管理员和了解系统很透彻的高级用户 经常使用命令行界面。外壳 脚本较常用于以命令行为主的系统 界面的选择 大多数 Windows 用户喜欢使用 Windows GUI 环境 通常, 应用程序开发人员根据应用编程接口 (API)来设计程序。 API 为方便应用 程序员规定了一组函数,包括每个函数的输入参数和返回值 在后台, API 函数通常为应用程序员调用实际的系统调用。 对大多数的程序设计语言, 运行时支持系统 (由编译器直接提供的函数库) 提供 了系统调用接口,以链接到操作系统的系统调用。 系统调用接口截取 API 函数的调 系统调用 提供操作系统服务接口。 系统调用 用,并调用操作系统中的所需系统调用。 向操作系统传递参数有三种常用方法。 最简单的是通过寄存器来传递参数。 不 过,有时参数数量会比寄存器多。这时,这些参数通常存在内存的块或表中,而 块或表的地址通过寄存器来传递 结束、中止 加载、执行 创建进程、终止进程 获取进程属性、设置进程属性 等待时间 等待事件, 信号事件 进程控制 分配和释放内存 创建文件、 删除文件 打开、关闭 读、写、重新定位 文件管理 获取文件属性,设置文件属性 请求设备、释放设备 读、与、重新定位 获取设备属性 、 设置设备属性 逻辑附加或分离设备 设备管理 系统调用的类型 获取时间或日期、设置时间或日期 获取系统数据、 设置系统数据 获取进程 、 文件或设备属性 信息维护 设置进程、文件或设备属性 创建 、 删除通信连接 发送、接收消息 通信 传送状态信息 附加或分离远程设备 保护 • 文件管理。 这些程序创建、 删除、 复制、 重新命名、 打印、 转储 、 列出、 操 作文件和目录。 • 状态信息。 有些程序可从系统那里得到日期、 时间、 内存或磁盘空间的可用数 量、用户数或其他状态信息。 还有一些则更为复杂, 可提供详细的性能、 登录和 调试信息。 • 文件修改。 有多个编辑器可以创建和修改位于磁盘或其他存储设备上的文件。 也 有专用命令,可用于查找文件内容或进行文本转换。 •程序语言支持。 常用程序语言(如 C、C++、Java 和 PERL 等) 的编译程序、 汇编程序、调试程序和解释程序 4 系统程序 •程序加载与执行。程序一旦汇编或编译后, 要加载到内存才能执行。系统可以 提供绝对加载程序、重定位加载程序、链接编辑器和覆盖式加载程序。系统还要 提供高级语言或机器语言的调试程序。 • 通信。 这些程序提供在进程、 用户和计算机系统之间创建虚拟连接的机制。 它 们允许用户在彼此的屏幕上发送消息,浏览网页,发送电子邮件,远程登录,从 一台机器向另一台机器传送文件。 • 后台服务。 所有通用系统都有方法,以便在引导时创建一些系统程序的进程。 这些进程中,有的执行完任务后就终止,而有的会一直运行到系统停机。一直运 行的系统进程, 称为服务、 子系统或守护进程。 用户目标:系统应该便于使用、易于学习和使用、可靠、安全 和快速。 设计目标 系统目标:操作系统应易于设计、实现和维护,也应灵活、可靠、正确且高 一个重要原则是策略与机制的分离。 机制决定如何做, 而策略决定做什么。 操作系统结构 对于灵活性,策略与机制的分离至关重要。策略可随时间或地点而改变。在最坏 机制与策略 情况下, 每次策略的改变都可能需要改变底层机制。 对策略改变不敏感的通用机 操作系统的设计与实现 制将是更可取的。这样策略的改变只需重新定义一些系统参数。 内核的最低层可以采用汇编语言。 高层函数可用 C; 系统程序可用 C 或 C++, 也可 用解释型脚本语言如 PERL 或 Python, 还可用外壳脚本。 4 采用高级语言或至少系统实现语言来实现操作系统的优势与用高级语言来编写应用 实现 程序相同: 代码编写更快, 更为紧凑, 更容易理解和调试。 采用高级语言实现操作系统的缺点仅仅在于速度的降低和存储的增加。 常用方法是将这种系统分成子系统或模块,而不只是一个单片系统,每个模块都 应是定义明确的部分系统,且具有定义明确的输入、输出和功能。 很多操作系统缺乏明确定义的结构。 通常 ,这些操作系统最初是小的、 简单的 、 功能有限的系统,但是后来渐渐超出了原来的范围。mcdos 另一个例子, 即最初的 UNIX 操作系统, 采用有限结构。 与 MS 简单结构 -DOS — 样, UNIX 开始也受限于硬件功能。 它由两个独立部分组成: 内核和系 统程序。 内核又分为一系列接口和_驱动程序, 即操作系统分成若干层(级)。最低层为硬件,最高层为用户接口。 操作系统层采用抽象对象,以包括数据和操纵这些数据的操作。 分层法的主要优点在于简化了构造和调试。 分层方法 每层的实现都只是利用更低层所提供的操作, 且只需知道这些操作做了什么, 而 并不需要知道这些操作是如何实现的。 因此, 每层要为更高层隐藏一定的数据结 构、操作和硬件。 分层法的主要难点在于合理定义各层。由于每层只能利用更低层的功能,因此有 操作系统的结构 必要仔细规划。主要问题是与其他方法相比效率较差 采用微内核 技术对内核进行模块化。 这种方法构造的操作系统, 从内核中删除所 有不必要的部件, 而将它们当作系统级与用户级的程序来实现。 这样做的结果是 内核较小。 微内核的主要功能是,为客户端程序和运行在用户空间中的各种服务提供通信。 通信是通过消息传递 来提供的, 微内核 微内核方法的优点之一是便于扩展操作系统。 所有新服务可在用户空间内增加, 因而并不需要修改内核。 当内核确实需要修改时, 所做修改也会很小, 遗憾的是,由于增加的系统功能的开销,微内核的性能会受损。 内核提供核心服务,而其他服务可在内核运行时动态实现。 动态链接服务优于直 接添加新功能到内核,这是因为对于每次更改,后者都要重新编译内核。 模块 实际上, 很少有操作系统采用单一的 、 严格定义的结构。 相反 , 它们组合了不同 的结构, 从而形成了混合系统, 以便解决性能、安全性和可用性等问题。 混合系统 广义而言, 调试 是查找和更正系统 (包括硬件和软件) 错误 性能问题为臭虫 (bug), 因此调试也会包括性能优化, 即通过解决处理瓶颈 而改善性能。 当一个进程发生故障时, 大多数操作系统将错误信息写到一个日志文件, 以 提醒系统操作员或用户所发生的问题。 操作系统也会进行核心转储 即进程内存的 捕获,并保存到一个文件以便以后分析。 用户级进程代码的调试是一个挑战。 由于内核代码多旦复杂、 硬件控制以及用户 故障分析 级调试工具的缺乏, 操作系统的内核调试更为复杂。 内核故障称为崩溃。 当发生 崩溃时,错误信息会保存到一个日志文件,并且内存状态会保存到一个崩溃转储 操作系统的调试 操作系统应有一些手段, 以便计算和显示系统行为的度量。 有几个操作 系统通过生成系统行为跟踪列表 来做到这一点。 性能优化 性能优化的另一方案是采用专用的交互工具,以便用户和管理员检查各种系统组件 的状态来寻找瓶颈。 DTrace DTrace 工具可以动态探测正在运行的系统, 包括用户进程和内核。 这些信息确定后,可有多种使用方法。一种极端情况是完全定制,系统管理员可 以修改操作系统源代码的副本。 一种定制稍微少些的情况是: 系统描述可用来创建表, 并从预先已编译的库中选 操作系统可以为某场所的某台机器进行专门设计、编码和实现。 择模块。这些模块链接起来,可以生成操作系统。 另外一种极端情况是: 可以构造完全由表驱动的系统。 • 使用什么 CPU ? 什么安装选项(扩展指令集 、 浮点运算等) ? 对于多 CPU 系 统,可能需要描述每个 CPU。 • 启动盘如何格式化? 分成多少个分区? 每个分区的内容是什么? 操作系统的生成 • 有多少可用内存? 有些系统可以求出这个值: 通过对内存位置一个个地引用, 直 到出现非法地址,这个过程可得到最后的合法地址及可用内存的数量。 • 有什么可用设备? 系统要知道如何访问每个设备 (设备号)、设备中断号、设备 类型与型号及任何特殊设备的特性。 • 需要什么操作系统的选项, 或者使用什么参数值? 这些选项或参数包括: 应使 用多大的缓冲区、 所需 CPU 调度算法的类型、 所支持进程的最大数量等。 加载内核以启动计算机的过程,称为系统引导。大多数计算机系统都有一小块代 码, 称之为引导程序 或引导加载程序 这段代码能够定位内核, 并加载到内存以开 始执行。 引导程序可以完成一系列任务。通常,有一个任务需要运行诊断程序来确定机器 状态。如果通过诊断,则程序可以继续启动步骤。引导程序也能初始化系统的所 有方面:从 CPU寄存器到设备控制器以及内存内容。最终,它启动操作系统。 对于小型的操作系统、简单的支持硬件和耐用的操作,将操作系统存储在 ROM 系统引导

中是适合的。 该方法有一个问题: 改动引导程序代码需要改动 ROM 芯片。

信息到内存,最后执行引导块的代码。

对大型操作系统或经常改变的系统, 引导程序存放在固件上, 而操作系统存放在磁盘上。 在这种情况下, 引导程序会先进行诊断, 然后从磁盘固定位置 读取整块

用户界面: 几乎所有操作系统都有用户界面这种界面可有多种形式 D —种是命令