**The UNIX Time-Sharing System**

**陈双-P17206032-中电十五所**

**介绍：**

UNIX是一个通用的、多用户的、交互式的操作系统，适用于较大的数字设备公司计算机。它提供了即使在更大的操作系统中也很少发现的一些特性，包括一个分层文件系统，I包含可拆卸卷、II兼容文件、设备和进程间I/O，III启动异步进程的能力，IV系统命令语言可按每个用户选择，V超过100个子系统，包括十几种语言，vi高度端口。能力。

本文讨论了文件系统和用户命令接口的性质和实现。也许UNIX的最重要的成就是证明一个强大的交互操作系统不需要昂贵的设备或人力成本：就像它可以运行在硬件成本低达40000美元，不到两个男人年花费在主系统软件上。然而，我们希望用户发现系统最重要的特点是它的简单性，优雅，易用。

**文件系统：**

系统最重要的作用是提供一个文件系统。从用户的角度来看，有三种文件：普通磁盘文件、目录和特殊文件。

普通文件包含用户在其上放置的任何信息，例如，符号或二进制（对象）程序。系统没有预期的特定结构。一个文本文件简单地包含一个字符串，由换行符标出的行。目录文件提供文件名和文件本身之间的映射，从而在整个文件系统中诱导一个结构。每个用户都有他自己的文件的目录；他也可以创建子目录来包含方便地一起处理的文件组。一个目录的行为与普通文件完全一样，只是它不能被非特权程序写入，所以系统控制目录的内容。但是，任何具有适当权限的人都可以像任何其他文件一样读取目录。特殊文件构成UNIX文件系统最不寻常的特性。每个支持的I/O设备与至少一个这样的文件相关联。特殊文件的读取和写入与普通磁盘文件一样，但是读写请求导致关联设备的激活。这样处理I/O设备有三个优点：文件和设备I/O尽可能相似；文件和设备名称具有相同的语法和含义，因此期望文件名作为参数的程序可以通过设备名；最后，特殊文件具有相同的名称。保护机制作为常规文件。

虽然文件系统的根始终存储在同一设备上，但不必将整个文件系统层次结构驻留在该设备上。有一个带有两个参数的挂载系统请求：一个现有普通文件的名称，以及一个特殊文件的名称，其相关的存储卷（例如磁盘包）应该具有包含其自己的目录层次结构的独立文件系统的结构。

对不同设备上的文件进行相同处理的规则只有一个例外：一个文件系统层次结构和另一个文件系统之间不存在链接。这种限制是强制执行的，以避免精心设计的簿记，否则将需要确保当移除可拆卸卷时移除链接。

虽然访问控制方案非常简单，但它具有一些不寻常的特性。系统的每个用户被分配一个唯一的用户标识号。当一个文件被创建时，它被标记为它的所有者的用户ID。还提供了新的文件是一组十个保护位。其中九个指定为文件所有者、其组的其他成员以及所有剩余用户独立地读取、写入和执行权限。如果第十位为ON，系统将临时将当前用户的用户标识（此后，用户ID）更改为文件的创建者，只要文件作为程序执行。

然而，当两个用户同时参与诸如在同一文件上写入、在同一目录中创建文件或删除彼此的打开文件等活动时，存在足够的内部互锁来保持文件系统的逻辑一致性。

**过程与图像：**

图像是计算机执行环境。它包括内存映像、一般寄存器值、打开文件状态、当前目录等。图像是伪计算机的当前状态。

一个过程是一个图像的执行。当处理器代表一个进程执行时，图像必须驻留在主存储器中；在执行其他进程期间，它保持在主存储器中，除非一个活跃的、较高优先级的进程的出现迫使它被交换到磁盘。

图像的用户存储器部分被划分为三个逻辑段。程序文本段在虚拟地址空间中的位置0开始。在执行过程中，这个段是写保护的，在执行同一程序的所有进程之间共享它的一个副本。在第一硬件保护字节边界之上，虚拟地址空间中的程序文本段开始非共享可写数据段，其大小可以通过系统调用扩展。从虚拟地址空间中的最高地址开始的是堆栈段，堆栈指针随着堆栈指针的波动而自动向下生长。

**UNIX系统设计：**

UNIX系统的成功很大程度上是因为它没有被设计成满足任何预先定义的目标。在整个过程中，是在与机器建立一种舒适的关系，并在操作系统和其他软件中探索想法和发明。

UNIX设计的考虑因素。首先，自然地设计了系统，以便于编写、测试和运行程序。我们对编程方便的最重要的表达是，该系统被安排用于交互式使用，即使原始版本只支持一个用户。一个适当设计的交互系统比“批次”系统更有效率和令人满意的使用。此外，这样的系统很容易适应非交互式使用，反之则不适用。

再者，对系统及其软件一直存在相当严重的尺寸限制。考虑到对合理效率和表现力的部分对立的欲望，尺寸约束不仅鼓励了经济，而且鼓励了设计的某种优雅。这可能是一个“痛苦的救赎”哲学的变相版本，但在我们的例子中却是奏效的。

最后，我们知道，系统就能够并且确实维持了它自己。如果一个系统的设计者被迫使用该系统，他们很快就会意识到它的功能和表面缺陷，并强烈动机去纠正它们，但为时已晚。因为所有的源程序总是可用的，并且很容易在线修改，所以当新的想法被他人发明、发现或建议时，我们愿意修改和重写系统和它的软件。

**小结：**

本文展示了这些设计考虑的前两个方面。例如，从编程的角度来看，文件系统的接口非常方便。最低可能的接口级别被设计为消除各种设备和文件之间的区别，以及直接访问和顺序访问之间的区别。不需要大的“访问方法”程序来将程序员与系统调用隔离，事实上，所有用户程序要么直接调用系统，要么使用一个小的库程序，少于一页长，缓冲多个字符并同时读取或写入它们。

编程方便的另一个重要方面是没有“控制块”，其复杂结构由文件系统或其他系统调用部分维护和依赖。一般来说，程序的地址空间的内容是程序的属性，并且我们试图避免对该地址空间内的数据结构进行限制。

由于要求所有程序都可以作为输入或输出与任何文件或设备一起使用，还需要将依赖于设备的考虑因素推送到操作系统本身。唯一的替代方案似乎是加载所有程序、处理每个设备的例行程序，这些设备在空间上是昂贵的，或者依赖于在实际需要时动态地链接到每个设备的例行程序的某些方法，这在开销或硬件开销方面都是昂贵的。

UNIX的成功与其说是新发明，不如说是充分利用了精心挑选的一套富饶的创意，尤其是展示了它们是实现一个小型而强大的操作系统的关键。