《文件》微课脚本设计

|  |  |
| --- | --- |
| 本节微课名称 | 文件 |
| 知识点描述 | 文件的相关原理 |
| 知识点来源 | 《现代操作系统》第四章 第1节 |
| 教学类型 | 讲授-启发-讨论-问答 |
| 设计思路 | 对于大部分学生来说如果单是通过对书本上的内容来对该章节的知识进行学习的话会觉得有些枯燥乏味。而如果是通过看视频的方式，能更灵活直观地表述出需要表达的内容，通过文字、图片和动画特效等，更加能够获取学生的兴趣，从而可以对页面置换算法的知识点有更深入的了解与认知。 |

教学过程：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 内容 | 画面 | 时间 |
| 一、片头 | 老师出场，介绍本节课知识点：  1. | 同学们好，今天我给大家带来的是《现代操作系统》的“文件”，这节课我们将学习以下内容（列出本次微课的知识点提纲） |  |
| 二、正文讲解 | 1.文件概念 | 文件在我们的日常生活中并不罕见，但是大家清楚文件的概念吗，到底什么是文件呢？就如同进程是对CPU的抽象、地址空间是对内存的抽象一样，文件是对硬盘的抽象。所谓的文件，实际上指的是一组带标识的、在逻辑上有完整意义的信息项的序列。标识指的就是文件名，逻辑上的完整意义是由文件的建立者或使用者来决定的，而信息项则是构成文件的基本单位，基本单位可以是单个或多个字节，各信息项之间具有一定的顺序关系。对于操作系统而言，在读写文件时需要定位读写指针，指向某一个具体的信息项。接下来我们从用户的角度来考察文件，即用户如何使用文件？文件具有哪些特性？ |  |
|  | 2.文件命名 | 也许任何一种抽象机制的最重要的特性就是对管理对象的命名方式，所以我们将从对文件的命名开始考察文件系统。在进程创建文件时，它给文件命名，在进程终止时，该文件仍旧存在，并且其他进程可以通过这个文件名对它进行访问。  文件的具体命名规则在各个系统中是不同的，不过所有的现代操作系统都允许用1至8个字母组成的字符串来作为合法的文件名（例如andrea、bruce和cathy）。通常，文件名中也允许有数字和一些特殊字符（例如Fig2-14、2和urgent!）。许多文件系统支持长达255个字符的文件名，其中有些区分大小写字母，有些则不区分。在前一类文件系统中，这三个文件名分别对应着三个不同的文件（maria、Maria和MARIA），而在后一类文件系统中，它们是同一个文件。  许多操作系统支持文件名用圆点隔开分为两部分，如文件名prog.c，圆点后面的部分称为文件扩展名，文件扩展名通常表示文件里的一些信息。在UNIX系统中，文件扩展名的长度完全由用户决定，一个文件甚至可以包含两个或更多的扩展名。如当前所示的文件名，这里的.html表示HTML格式的一个Web页面，.zip表示该文件已经采用zip程序压缩过（homepage.html）。一些常用的文件扩展名及其含义如上图所示。  在某些系统中（如所有UNIX版本），文件扩展名只是一种约定，操作系统并不强迫采用它。名为file.txt的文件也许是文本文件，这个文件名更多的是提醒所有者，而不是表示传送什么信息给计算机。但是另一方面，C编译器可能要求它编译的文件以.c结尾，否则它会拒绝编译。然而，操作系统并不关心这一点。  对于可以处理多种类型文件的某个程序，这类约定是特别有用的。例如，C编译器可以编译、链接多种文件，包括C文件和汇编语言文件。这时扩展名就很有必要，编译器利用它来区分哪些是C文件，哪些是汇编文件，以及哪些是其他文件。  与UNIX相反，Windows关注扩展名且对其赋予了含义。用户（或进程）可以在操作系统中注册扩展名，并且规定哪个程序“拥有”该扩展名。当用户双击某个文件时，“拥有”该文件扩展名的程序就会启动并运行该文件。例如，双击file.docx启动了Microsoft Word程序，并以file.docx作为待编辑的初始文件。 |  |
|  | 3.文件结构 | 文件可以有多种构造方式，在图中列出了常用的三种方式，图a中的文件是一种无结构的字节序列，事实上操作系统不知道也不会关心文件的内容是什么，操作系统所见到的就是字节，其文件内容的任何含义只在用户程序中解释。把文件看成字节序列为操作系统提供了最大的灵活性。用户程序可以向文件中加入任何内容，并以任何方便的形式命名。操作系统不提供任何帮助，但也不会构成障碍。对于想做特殊操作的用户来说，后者是非常重要的。  图b表示在文件结构上的第一步改进。在这个模型中，文件是具有固定长度记录的序列，每个记录都有其内部结构。把文件作为记录序列的中心思想是：读操作返回一个记录，而写操作重写或追加一个记录。  第三种文件结构如图c所示。文件在这种结构中由一颗记录树构成，每个记录不必具有相同的长度，记录的固定位置上有一个“键”字段。这棵树按“键”字段进行排序，从而可以对特定的“键”进行快速查找。虽然在这类结构中取“下一个”记录是可以的，但是基本操作并不是取“下一个”记录，而是获得具有特定“键”的记录。如图c所示的文件，用户可以要求系统读取“键”为Pony的记录，而不必关心记录在文件中的确切位置。更进一步地，用户可以在文件中添加新记录。但是，用户不能决定把记录添加在文件的什么位置，这是由操作系统决定的。这类文件结构与无结构字节流明显不同，但它在一些处理商业数据的大型计算机中获得广泛使用。 |  |
|  | 4.文件类型 | 很多操作系统支持多种文件类型，如UNIX和Windows都有普通文件和目录，其中UNIX还有字符特殊文件和块特殊文件。普通文件是包含有用户信息的文件，在图中列出的所有文件都是普通文件，目录是管理文件系统结构的系统文件，字符特殊文件和输入/输出有关，用于串行I/O类设备，如终端、打印机网络等，块特殊文件用于磁盘类设备，接下来主要讨论的是普通文件。  普通文件一般分为ASCII文件和二进制文件。ASCII文件由多行正文组成，在某些系统中，每行用回车符结束，其他系统则用换行符结束，有些系统还同时采用回车符和换行符，文件中各行的长度不一定相同。ASCII文件的最大优势是可以显示和打印，还可以用任何文本编辑器进行编辑。再者，如果很多程序都以ASCII文件作为输入和输出，就很容易把一个程序的输出作为另一个程序的输入。其他与ASCII文件不同的是二进制文件，打印出来的二进制文件是无法理解的、充满混乱字符的一张表。通常，二进制文件有一定的内部结构，使用该文件的程序才了解这种结构。  如图a是一个简单的可执行二进制文件，它取自某个早期版本的UNIX。尽管这个文件只是一个字节序列，但只有当文件的格式正确时，操作系统才会执行这个文件。这个文件有五个段，分别为文件头、正文、数据、重定位位及符号表。文件头以所谓的魔数开始，表明该文件是一个可执行的文件（防止非这种格式的文件偶然运行），魔数后面是文件中各段的长度、执行的起始地址和一些标志位，程序本身的正文和数据在文件头后面，这些被装入内存，并使用重定位位重新定位，符号表则用于调试。  二进制文件的第二个例子是UNIX的存档文件，它由已编译但没有链接的库过程（模块）组合而成。每个文件以模块头开始，其中记录了名称、创建日期、所有者、保护码和文件大小。该模块头与可执行文件一样，也都是二进制数字，打印输出它们毫无意义。  所有操作系统必须至少能够识别它们自己的可执行文件的文件类型，其中有些操作系统还可识别更多的文件类型。一种老式的TOPS-20操作系统甚至可检查可执行文件的创建时间，然后它可以找到相应的源文件，看它在二进制文件生成后是否被修改过，如果修改过，操作系统自动重新编译这个文件。在UNIX中，就是在shell中嵌入make程序。这时操作系统要求用户必须采用固定的文件扩展名，从而确定哪个源程序生成哪个二进制文件。如果用户执行了系统设计者没有考虑到的某种操作，这种强制类型的文件有可能会引起麻烦。比如在一个系统中，程序输出文件的扩展名是.dat（数据文件），若用户写一个格式化程序，读入.c（C程序）文件并转换它（比如把该文件转换成标准的首行缩进），再把转换后的文件以.dat类型输出。如果用户试图用C编译器来编译这个文件，因为文件扩展名不对，C编译器会拒绝编译。若想把file.dat文件复制到file.c文件也不行，因为系统会认为这是无效的复制（防止用户错误）。尽管对初学者而言，这类“保护”是有利的，但一些有经验的用户却感到很烦恼。，因为他们要花很多精力来适应操作系统对合理与不合理操作的划分。 |  |
|  | 5.文件访问 | 早期操作系统只有一种文件访问方式：顺序访问。进程在这些系统中可从头按顺序读取文件的全部字节或记录，但不能跳过某一些内容，也不能不按顺序读取。顺序访问文件是可以返回到起点的，需要时可多次读取该文件。在存储介质是磁带而不是磁盘时，顺序访问文件是很方便的。  当用磁盘来存储文件时，可以不按顺序地读取文件中的字节或记录，或者按照关键字而不是位置来访问记录。这种能够以任何次序读取其中字节或记录的文件称作随机访问文件。许多应用程序需要这种类型的文件。随机访问文件对很多应用程序而言是必不可少的，如数据库系统。如果乘客打电话预订某航班机票，订票程序必须能直接访问该航班记录，而不必先读出其他航班的成千上万个记录。  有两种方法可以指示从何处开始读取文件。一种是每次read操作都给出开始读文件的位置。另一种使用一个特殊的seek操作设置当前位置，在seek操作后，从这个当前位置顺序地开始读文件。UNIX和Windows使用的是后一种方法。 |  |
|  |  | 以上就是本节课的全部内容了，同学们掌握好这节课的知识了吗？下面我们来做几道题巩固一下吧。 |  |
| 三、结尾 | 本节课结束 通过几道小问题回顾今天所学知识 | 转到互动答题 |  |