微机操作系统系列丛书(二)

WPS

工具箱

香港金山公司北京开发部编写

学苑出版社

微机操作系统系列丛书(二)

WPS 工具箱

香港金山公司北京开发部　编写
雷　军　李儒雄　审校

学苑出版社

1994

(京)新登字 151 号

内 容 简 介

WPS 自八九年推出问世以来，倍受广大用户喜爱，风靡全国，成为桌面办公系统的标准，领导着桌面办公系统发展的新潮流。为了便于广大用户更好地使用 WPS，我们专门针对有些用户在使用 WPS 时出现的困难和问题，编写了一些应用程序。同时这些程序也可作为想在 SPDOS、WPS 上做些开发编程工作的读者一些示范。

本书讲述了如下内容：在任何汉字系统下，不用进入 WPS 系统，就可浏览 WPS 文件，或不用进入汉字系统就可看任何西文文件——程序 WORD；WPS 文件的口令解除方法——程序 UNPASSWD；各种汉字系统如何挂接金山字库；一次可实现多个文件格式的转换——程序 WPSCHG；如何设置 SPDOS 部分不常用的参数——程序 SPCFG；如何在 SPDOS 命令行下直接退出程序——程序 SPQUIT；如何高效彻底地清除内存——程序 RI，以及由于误操作导致 WPS 死机后，如何恢复 WPS 文件——程序 995。书中还提供了以上内容的源程序。这些源程序经过编译后，用户可直接使用，同时程序员也可借鉴学习编程技巧。

适用范围：所有使用计算机的用户和办公自动化操作员。

欲购本书的用户，请直接与北京 8721 信箱联系，邮编：100080，电话：2562329。

微机操作系统系列丛书(二)

WPS 工具箱

编　　写：香港金山公司北京开发部
审　　校：雷　军　李儒雄
责任编辑：甄国宪
出版发行：学苑出版社　　**邮政编码**：100036
社　　址：北京市海淀区万寿路西街 11 号
印　　刷：兰空印刷厂
开　　本：787×1092 1/16
印　　张：7.25　　**字数**：180 千字
印　　数：1～5000
版　　次：1994 年 5 月北京第 1 版第 1 次
ISBN 7－5077－0885－3/TP・27
本册定价：9.00 元

前　言

WPS 自八九年问世以来，以其简捷明了的菜单风格，简单实用的功能特点，开创了桌面办公系统的新天地，受到了广大用户的交口称赞，风靡全国。几年来，在广大用户的支持和鼓励下，开发人员不断创新，系统自身不断完善，形成了很多独到的特点，领导着桌面办公系统发展的新潮流。

随着 WPS 的广泛使用和市场的逐步繁荣，在软件法规不健全的情况下，解密的软件版本也随之出现，以 1990 年 WPS 版本 2.0、2.1、2.2 和 SPDOS 版本 5.0、5.1、5.2 为最多。由于解密不彻底，又没有硬件——汉卡的支持，正版的 WPS 的很多功能盗版软件没法实现，再加上没有售后服务，很多用户使用起来很不方便。这些盗版软件与金山公司的正版软件混杂在一起，使一些用户对 WPS 产生了误解，当他们在使用中出现误设密码、显示速度慢、打印机不兼容等问题以后，就认为是金山公司产品问题，这既损害了用户的利益又影响了金山公司的声誉。

大约是九三年六月的一天，著名作家张洁女士带着她心爱的便携机，找到位于白石桥科技贸易中心 311 的方正集团公司汉卡部，诉说她的委屈：她按正常的使用方法使用，但她的小说调不出来了，提示是请输入密码。这是应一家报社约稿写的连载小说，报社急待录用，这下急坏了张洁女士。她以为这就是我公司的正版软件，对我公司很有些怪罪。经我公司销售人员解释，她才知道错怪了我公司，对不法商人很是生气。在我公司北京开发部经理雷军先生的帮助下，很快就解除了密码。张洁女士喜出望外，不久就写出了散文《电脑发烧友》，在多家报纸上转载。

值得高兴的是 1991 年中国计算机软件保护条例正式颁布实行，该条例依据中华人民共和国著作权法而制定，标志着计算机行业有法可依了。从此 SPDOS 汉字系统、WPS 文字处理系统、SPT 图文混排系统等系列产品受中华人民共和国著作权法和计算机软件保护条例的保护，而且北大方正 Super 为北大方正集团公司的注册商标，受商标法保护。希望广大用户能够自觉守法，使 WPS 系统得以更好地发展，至臻至美！

虽然 WPS 系统几经更新换代，时至今日已推出方正 Super Ⅵ型汉卡，但有的用户并不需要 WPS 日益强大的功能，从价格等因素考虑希望购买 WPS2.2 版本，但我公司对这些低版本已经不再出售，有一些在汉字系统开发方面较有影响的公司，比如北京希望电脑公司、北京超想公司、昆明明星电脑公司等，他们已经正式向金山公司购买了 WPS2.2 版本使用权。那些希望使用 WPS2.2 版本软件的用户可以合法地在上述公司购买。在此，我们郑重宣布：

金山公司对任何非法销售 WPS 的公司保留法律追究的权力。

当合法用户频繁地要求技术服务的时候，我们一定替用户排忧解难，如果用户买了盗版软件，我们就无能为力，所以希望广大用户一定要根据不同需要购买方正 Super 系列汉卡及正版软件，我们对任何盗版软件以及在使用盗版软件中产生的任何问题不承担法律责任、技术咨询、版本升级等服务。

本书由香港金山公司北京开发部组织编写。其中第一章由陈波先生主笔，主要讲述了在任何汉字系统下，不用进入 WPS 系统，就可浏览 WPS 文件，或不用进入汉字系统就可看任何西

文文件；第七章由雷军先生主 笔，主要讲述如何高效彻底地清除内存的方法及其应用；由于误操作导致 WPS 死机后，恢复这些 WPS 文件；其余章节由冯志宏先生主笔，主要讲述了如下内容：WPS 文件的口 令解除方法；各种汉字系统如何挂接金山字库；一次可实现多个文件格式的转换；SPDOS 部分参数的设置；SPDOS 命令行直接退出程序。书中源程序由各章作者编写，并 经过优化编译。

全书由李儒雄先生组稿，雷军先生审定，北京开发部的孙宏等同事也提出了很多 宝贵意见。

由于我们开发任务繁重，书中的遗漏及错误，敬请朋友们批评斧正。

特别欢迎广大朋友们在 SPDOS 上开发应用产品，如有好的产品，请与我们联系。

目　录

第一章　WPS 文件浏览工具

我们将在这一章向您介绍一个浏览 WPS 文件的工具：READ。

使用 READ 是一个快速浏览 WPS 文件的办法。READ 既可以在 SPDOS 下运行，也可以在其它汉字系统下运行。READ 可以依次浏览多个文件。当您需要在一大堆文件中查找某一个文件时，一时又想不起它的文件名，您可以使用 READ。通过察看文件内容，确定所需的文件。

本章的内容分为三个小节。

第一节首先向您介绍 READ 的具体功能和用法. 如果您只想了解怎样使用 READ，阅读这一节就足够了。

第二节简要介绍 WPS 文件的结构。如果您是软件编制人员，需要在程序中直接读写 WPS 文件，或者您感兴趣的话，阅读这一节会有所帮助。这一节也是下一节内容的基础。

第三节讲解 READ 是如何具体实现的。为了方便您的阅读，我们还附上了 READ 的全部源程序. 这一节是为想了解 READ 是如何编写的读者准备的。由于 READ 是用 Pascal 语言编写，您需要懂一点 Pascal（具体地说是 Turbo Pascal）。我们建议您先简要地读一下第一节，然后仔细地读第二节，最后读第三节。

1.1　READ 的功能和用法

为了能运行 READ，您只需要 READ. EXE 这一个文件就够了。READ. EXE 必需放在当前目录，或者您放在 PATH 环境变量所指定的目录中。

一、使用 READ 调用 WPS 文件

运行 READ 之前，您需先启动中文操作系统，如启动 SPDOS 汉字系统。READ 可在很多汉字系统下运行，比如晓军 2.13，王码 480，联想汉字等。之后，在 DOS 提示符下，您先输入 READ，然后输入一个空格，再输入您要浏览的文件名，

敲回车键即可。这时，您的文件就会出现在屏幕上。屏幕的第一行是文件名，最后一行是提示行，中间显示文件内容。

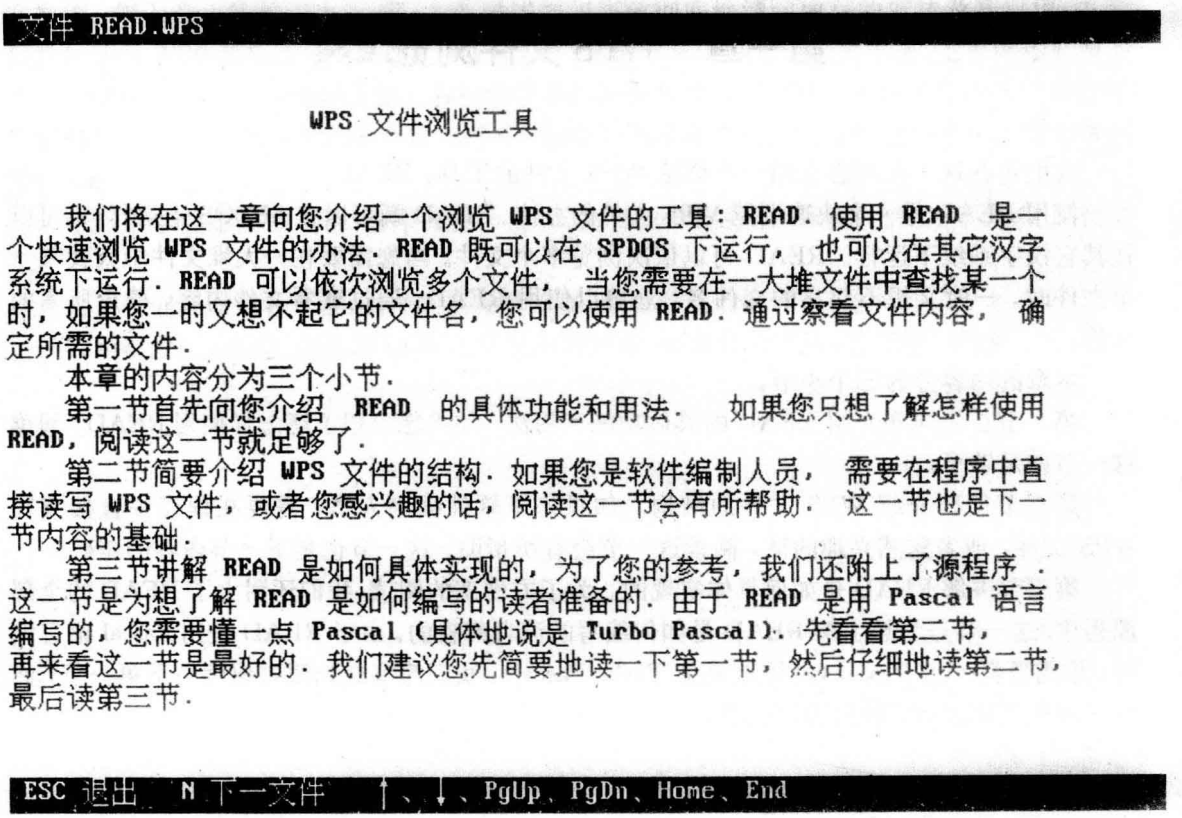
比如有一篇 WPS 文件叫 READ. WPS（它的内容就是本章前面部分）。浏览 READ。WPS 文件时，在 DOS 提示符下输入 SPDOS 命令，然后按回车键，即：

```
C>SPDOS
```

屏幕提示一些信息后，又出现 DOS 提示符。接着输入 READ READ. WPS 命令，再按回车键，即：

```
C>read read. wps
```

屏幕上就出现了文件 READ. WPS 的内容：



这时，您就可以通篇浏览您的文章了。

屏幕第一行显示您正在浏览的文件名是 READ. WPS。最后一行是提示行，具体解说见下面的“提示行功能键介绍”。

有了 READ，不必进入 WPS，就可以阅读 WPS 文件。在 READ 中，您只能读文件，而不能改动文件。因此，如果您只想看看文件内容，运行 READ 要比运行 WPS 更快速，更方便。

注意：READ 只显示 WPS 文件的文本内容，不显示 WPS 的控制字符。如果您还需要文件的控制字符显示出来，必须使用 WPS。

二、提示行功能键介绍

浏览过程就象在 WPS 中编辑文件时前后翻看文件一样。

屏幕提示行中提示的 8 个功能键含义如下：

ESC 键	退出 READ，返回到 DOS 提示符下
字母 N 键	同时浏览多个文件时，结束当前文件，浏览下一文件
Home 键	屏幕显示文件开始位置

End 键	屏幕显示文件结尾位置
↑向上光标键	屏幕显示文件的上一行内容
↓向下光标键	屏幕显示文件的下一行内容
PgUp 键	显示文件的上一屏幕内容
PgDn 键	显示文件的下一屏幕内容

如果您按下了滚动控制键而屏幕内容没有变化，这表示已到了文件头或文件尾。

读完文件，您只要敲 ESC 键，READ 将清除屏幕显示内容，返回 DOS 下。

三、同时浏览多个文件的方法

READ 能同时浏览多个 WPS 文件。浏览前，您只要将各文件名一一输入到命令行上，中间用空格分开即可。所给文件名中可以含有文件名通配符？和 * 。READ 会自动查找匹配文件并依次显示。

比如，您的当前目录下有以下四个文件：jianjie. wps，mulu. wps，1. wps 和 2. wps。输入以下命令：

```
C>read jianjie. wps mulu. wps 1. wps 2. wps
```

或

```
C>read jianjie. wps mulu. wps ?. wps
```

或

```
C>read *. wps
```

使用这三种命令，都能同时浏览这四个 WPS 文件。您先读的是 jianjie. wps。读完 jianjie. wps 后，您按字母 N 键，下一文件 mulu. wps 将显示出来。然后依次是 1. wps 和 2. wps. 浏览过程中，任何时候您都可以按 ESC 键退出 READ。

如果您指定的某一文件不存在，READ 将忽略它，继续下一文件。您可以察看屏幕第一行上的文件名，从而确定当前显示的是哪一文件。

四、浏览非 WPS 文件

READ 不只是能显示 WPS 格式文件的内容，任何其他格式的文件它都能显示，

只不过有些文件的内容不是文本，显示出来的是一些无意义的文字。READ 能很好地显示 WPS 文件和普通文本文件(如您 C:盘根目录下的 CONFIG. SYS 和 AUTOEXEC. BAT 就是文本文件)。只不过 READ 对 WPS 文件予以特殊考虑，显示时滤掉了控制字符。对其它文件，READ 按原样显示。

阅读非 WPS 文件的方法与阅读 WPS 文件一样。您只要在命令行上输入指定的文件名即可。阅读过程也完全一样。

事实上，READ 的命令行上可以跟多个任意的文件描述名。READ 会自动识别：如果是 WPS 文件，按 WPS 文件的方法显示；其它文件按普通方法显示。

五、运行信息

在 READ 的运行过程中，会出现一些信息。以下列出了全部信息及其含义：

1)屏幕显示：

用法：READ 文件名

如果您没有在命令行 READ 后跟文件名，READ 将报告这一信息，向您提示 READ 的使用方法：即在 READ 后，跟一个文件名。

2)屏幕显示：

文件 xxxxxxxx.xxx 经过加密，不能显示

READ 不能显示加密过的 WPS 文件。如果您指定了经过加密的 WPS 文件，READ 将报告这一信息。按回车键后，系统自动继续下一工作(如显示下一个文件)。

3)屏幕显示：

文件未找到

如果您在命令行上所给文件都不存在，READ 将报告这一信息。

1.2 WPS 文件结构简介

WPS 文件不同于普通文本文件，它是由两部分组成的。第一部分是文件头，含有文件的一些控制信息。第二部分是文本内容，它相当于普通文本文件，只不过含有控制字符。WPS 文件头长 400H 字节。对编程者来说，有关的信息如下：

- 文件头的第一个字节是 WPS 文件类型版本号。其有效值可以是从 0 到 4 之 间的任意一值，分别代表相应的文件类型版本号。不同版本的 WPS 文件 类型基本上是兼容的。
- 文件头的第二个字节固定为 0FFH。通过检查第一字节和这一字节，可 以判断文件类型是否为 WPS。
- 文件头位置 2DDH 处存放着文件的密码。密码是经过变形的。如果文件 没有密码，此处将是零。
- 文件头的其它地方存放着文件的编辑信息，比如块定义的起止位置，当 前光标位置，控制字符的显示状态等等。这些信息只对 WPS 有用，一 般情况下不用考虑。

WPS 文件的文本内容与文件是否设置了密码有关。如果设置了密码，那么文本内容已经过变形，直接察看将会是一些毫无意义的字符。只有 输对密码后，WPS 才能正确读出经过加密的文件内容。如果没有设置密码，文本内容可直接察看，只不过控制字符的显示与在 WPS 中的不一样。

WPS 控制字符的 ASCII 码在 80H 到 0A0H 之间，既不是汉字代码，也不是英文字母或符号，它占用了 ASCII 码字符集中图形符号的一部分。WPS 在处理这些代码时，将它们以相应的排版控制符号显示。由于 READ 无法显示这些排版控制符号，所以将它们滤掉了。

在这些控制字符中有两个需要明白其含义：8DH 和 8AH。8DH 是 WPS 的软回车符号，

8AH 是 WPS 的软换行符号。在 WPS 中，它们显示为空心的回车符。

1.3　READ 是如何实现的

READ 用 Turbo Pascal 编写，它使用了 Turbo Pascal 的 CRT 和 DOS 标准单元(UNIT)。CRT 单元中提供了显示字符的颜色控制过程 TextBackground 和 TextColor，窗口定义过程 Window，读用户输入函数 ReadKey 和光标定位过程 GotoXY。DOS 单元中提供了文件查找过程 FindFirst 和 FindNext。

READ 的主程序依次找到每个文件，并把文件交给 ViewFile 过程处理。在每处理完一个文件后，主程序检查用户是否按了 ESC 键，确定是继续下一文件，还是返回 DOS。主程序调用 FindFirstFile 函数检查用户所给文件是否存在，调用 FindNextFile 函数检查是否还存在下一个文件。

在显示文件之前，ViewFile 过程先读入文件前 10K 内容，检查 WPS 文件类型标记。如果是 WPS 文件，还要检查是否经过加密。对于加密过的文件，READ 给用户一个提示，并跳过此文件。此后，ViewFile 进入一个 repeat 循环，读用户键盘输入并滚动显示内容，直到用户按下 ESC 键或 N 键。

READ. PAS 中有两点需要注意，一个是控制字符的过滤，另一个是缓冲区的调整。

在 ViewFile 过程中子函数 GetLine 和 BackwardPos 涉及到控制字符的过滤处理。

GetLine 返回从文件的某一位置起下一行文件的内容。GetLine 从指定位置起，通过循环反复地取下一字符，直到取得硬回车符(0DH)或软回车符(8DH)。取到的控制字符予以丢弃。如果取到的字符数到了 79 个，还没有遇上回车符，GetLine 也将返回。

BackwardPos 返回从文件的某一位置起上一行文件内容的位置。它需要两个回车符来确定此位置。一个是上一行的回车符，一个是上上一行的回车符，它们之间的内容就是上一显示行。因此，BackwardPos 采用了两个循环，分别来找到它们。在这两个循环中，BackwardPos 也要忽略掉控制字符。

由于所阅读的文件大小不固定，READ 运行时不一定有足够的内存将文件整个读入，所以 READ 使用一个固定大小的缓冲区(目前为 10K，由 BuferSize 常量定义)，缓冲区内总包含了当前显示的文件内容。用户滚动屏幕内容时，READ 先判断新的显示内容是否已在缓冲区中，只有当新内容不在缓冲区中时，READ 才读磁盘文件，调整缓冲区内容。这一过程在 ViewFile 的 GetByte 函数中处理。

为了对用户的输入取得最快的反应，缓冲区既不能太大，也不能太小。太大了，每次读磁盘文件的时间就长，程序反应就慢。缓冲区太小了，用户滚屏时，读磁盘的次数就会增加，程序反应也会慢。目前缓冲区大小定为 10K，能容纳 5 整屏的内容，比较适中。

当显示内容不在缓冲区中时，如何调整缓冲区内容也会影响到程序的效率。比较好的方法是采用预读。考虑以下情况，目前显示内容正好处于缓冲区的开始位置，用户按下 PgUp 键，向上翻页，缓冲区内容就要调整，需读一次磁盘文件。如果只向前读一页内容，使显示内容仍然处于缓冲区的开始位置，当用户再向上翻页时，需要再读一次磁盘文件。这样，当用户向一个方向连续翻页时，就会导致连续读盘。为了避免这种情况发生，GetByte 向前或向后预读超过一页的内容，目前为 4K，它由 BufferAdjust 常量控制。

READ. PAS 已在 Turbo Pascal 7. 0 下编译通过. 它也应该能在 TurboPascal 5. 0 及以上版本下编译，不过笔者已没有 Turbo Pascal 早期版本的编译器，所以没有实际测试过。

附录　READ 的源程序

```
{*********************************************}
{                                             }
{            READ. PAS                        }
{            一个浏览 WPS 文件的工具          }
{                                             }
{            版权所有 Kingsoft Co. 1993－1994 }
{                                             }
{*********************************************}

program Read;

uses
  Crt, Dos;

const
  { 以字节计的文件缓冲区大小 }
  BufferSize = 10 * 1024;

  { 常用键的 ASCII 码 }
  EscKey = #$1B;
  SpaceKey = #$20;
  EnterKey = #$0D;

  { 功能键的扫描码 }
  UpKey = #$48; DownKey = #$50;
  HomeKey = #$47; EndKey = #$4F;
  PgUpKey = #$49; PgDnKey = #$51;

var
  BufSeg: Word;
  Buffer: Pointer;
  NameCounter: Integer;
  SearchInfo: SearchRec;
  EscKeyPressed: Boolean;

function GetFilePath: String;
var
```

```
    S: String;
    I: Integer;
begin
    S := ParamStr(NameCounter);
    I := Length(S);
    while (I > 0) and not (S[I] in ['/', '\', ':']) do begin
      Delete(S, I, 1);
      Dec(I);
    end;
    GetFilePath := S;
end;

function FindFirstFile: Boolean;
begin
    NameCounter := 0;
    repeat
       Inc(NameCounter);
       FindFirst(ParamStr(NameCounter), Archive, SearchInfo);
    until (DosError = 0) or (NameCounter = ParamCount);
    FindFirstFile := DosError = 0;
end;

function FindNextFile: Boolean;
begin
    repeat
    FindNext(SearchInfo);
    if (DosError <> 0) and (NameCounter < ParamCount) then
      repeat
      Inc(NameCounter);
      FindFirst(ParamStr(NameCounter), Archive, SearchInfo);
      until (DosError = 0) or (NameCounter = ParamCount);
    until (DosError = 0) or (NameCounter = ParamCount);
    FindNextFile := DosError = 0;
end;

procedure ViewFile;
var
    F: File;
    Ch: Char;
    WpsFile: Boolean;
    Size, BufBeg, BufEnd, DispBeg, DispEnd: Longint;

    procedure ReadFile(P: Longint);
```

```
  { 从位置 P 起读文件，直到缓冲区满或文件尾 }
  var
    NumRead: Word;
  begin
    BufBeg := P;
    Seek(F, P);
    BlockRead(F, Ptr(BufSeg, 0)^, BufferSize, NumRead);
    BufEnd := FilePos(F);
  end;

  function GetByte(P: Longint): Byte;
  { 取文件位置 P 处的字节 }
  const
    { 如果所取字节不在缓冲区中，将缓冲区内容前后调整的字节数 }
    BufferAdjust = 4 * 1024;
  begin
    if P < BufBeg then
      begin { 所取字节在缓充区内容之前 }
          if BufBeg - P <= BufferSize - BufferAdjust then
              ReadFile(BufBeg - (BufferSize - BufferAdjust))
            else if P > BufferAdjust then
              ReadFile(P - BufferAdjust)
              else
              ReadFile(0);
        end
    else if P > BufEnd then
        begin { 所取字节在缓充区内容之后 }
          if P - BufEnd <= BufferSize - BufferAdjust then
            ReadFile(BufEnd - BufferAdjust)
          else if Size - P > BufferAdjust then
            ReadFile(P - BufferAdjust)
          else
          ReadFile(Size - BufferSize);
        end;
    GetByte := Mem[BufSeg:(P - BufBeg)];
  end;

  function DispTop: Longint;
  { 返回文件内容的起始位置 }
begin
    if WpsFile
      then DispTop := $400          { WPS 文件的前 400H 个字节为 WPS 控制信息 }
      else DispTop := 0;
```

```
end;

function GetLine(var P: Longint): String;
{ 从文件位置 P 处取一行显示内容，滤掉特殊符号 }
var
  B: Byte;
  S: String;
 LineEnd: Boolean;
begin
  S := '';
  LineEnd := False;
  while (P < Size) and (Length(S) <= 79) and not LineEnd do begin
 B := GetByte(P);
 Inc(P);
    if (B = $0D) or (WpsFile and (B = $8D)) then
    LineEnd := True { 如遇到回车符，表示这一行结束 }
   else if WpsFile and (B in [$80..$A0]) then
   begin              { 滤掉 WPS 的控制字符 }
   if B <> $8A then
            Inc(P);
    end
   else if Not (B in [7, 8, $0A, $1A]) then
   begin                    { 滤掉响铃符，退格符，换行符和文件结束符 }
      if B in [0, 9, $7f]
         then S := S + ''        { 把其它控制字符转换成空格符 }
         else S := S + Chr(B);      { 剩下的字符是文件的实质性内容 }
       end;
. end;
  GetLine := S;
end;

function BackwardPos(P: Longint): Longint;
{ 从文件位置 P 处查找上一行显示内容的起始位置 }
var
  B: Byte;
  L: Integer;
  LineEnd: Boolean;
begin
  { 先往前查找上一行的结束位置 }
  LineEnd := False;
  while (P > DispTop) and not LineEnd do begin
    Dec(P);
    B := GetByte(P);
```

```
    if (B = $0D) or (WpsFile and (B = $8D)) then
      LineEnd := True{ 如遇到回车符，表示已找到上一行的结束位置 }
    else if not ((B in [7, 8, $0A, $1A]) or (WpsFile and (B in [$80..$A0])))
    then begin          { 如遇到非控制字符，也表示已找到上一行的结束位置 }
              LineEnd := True;
              Inc(P);
              end;
  end;
  { 再从上一行的结束位置开始，往前查找上一行起始位置 }
  L := 0;
  LineEnd := False;
  while (P > DispTop) and (not LineEnd) and (L <= 79) do begin
      Dec(P);
      B := GetByte(P);
      if (B = $0D) or (WpsFile and (B = $8D)) then
        begin          { 如遇到回车符，表示已找到上一行的起始位置 }
            LineEnd := True;
          Inc(P);
        end
      else if not ((B in [7, 8, $0A, $1A]) or (WpsFile and (B in [$80..$A0])))
        then Inc(L);
    end;
    BackwardPos := P;
 end;

procedure DispFullScr;
{ 从当前位置开始，显示一整屏内容 }
var
  I: Byte;
begin
    ClrScr;
    I := 1;
    DispEnd := DispBeg;
    repeat
        GotoXY(1, I);
    Write(GetLine(DispEnd));
    Inc(I);
    until (DispEnd >= Size) or (I > 23);
end;

procedure PrevLine;
{ 将显示内容下滚一行 }
var
```