怎样用 MFC 实现打印功能？

最佳答案

```
int count=this->m_logList.GetItemCount(),page,row=0;
int i,j;
page=count/40+1;


    CPrintDialog print(false);
if(print.DoModal()==IDOK)
{
    CDC printed;
    printed.Attach(print.GetPrinterDC());
    DOCINFO pdoc;
    pdoc.cbSize=sizeof(pdoc);
    pdoc.lpszDocName=L"pdoc";
    pdoc.lpszDatatype=NULL;
    pdoc.fwType=NULL;
    pdoc.lpszOutput=NULL;
    if(printed.StartDoc(&pdoc)>=0)
    {
     LOGFONT logfont;
     memset(&logfont,0,sizeof(LOGFONT));
     logfont.lfHeight=75;
     CFont font;
     CFont *oldfont=NULL;
     if(font.CreateFontIndirect(&logfont))
      oldfont=(CFont*)printed.SelectObject(&font);
```

```
for(j=1;j<=page;j++)
{
 printed.StartPage();
 int x=500,y=400;//A4 纸，页面中的位置,横向为 x 轴，纵向是 y 轴,A4 maxX=4000 maxY=7000 建议按字符大小为 75，每页安排 40 条纪录，初试纪录开始位置为 x=500 y=200
 CString pageHead,pageBottom;
 pageHead.Format(_T("日志信息纪录统计表"));
 printed.TextOut(1500,100,pageHead); //打印页眉
 CString title;//设置标题栏
 title.Format(_T("序号                    时间                    操作"));
 printed.TextOut(500,200,title); //打印页眉
 CString stt;

stt.Format(_T("_____"));
 printed.TextOut(500,200+80,stt); //打印页眉

 for(i=1;(i<40)&&(row<count);i++)
 {
  CString record(_T(""));
  record+=this->m_logList.GetItemText(row,0)+L"    ";
  record+=this->m_logList.GetItemText(row,1)+L"    ";
  record+=this->m_logList.GetItemText(row,2);
  printed.TextOut(x,y,record);
  y+=80;
  printed.TextOut(x,y,stt);
  y+=80;
```

```
        row++;
      }
      pageBottom.Format(_T("共%d 页     第%d 页"),page,j);
      printed.TextOut(1500,y,pageBottom);

      printed.EndPage();//此页结束
    }
    font.DeleteObject();
    if(oldfont!=NULL)
      printed.SelectObject(oldfont);
    printed.EndDoc();
  }
  printed.DeleteDC();
}
```

MFC 简单打印方法

2001-04-06· ·vc.faq··yesky

通过 MFC 来完成打印作业有很多的方法，应用场合也有所不同。我们通常是利用视图框架在 MFC 基础之上按照 OnPreparePrinting() à OnBeginPrinting() ? à OnPreprareDC() à OnPrinting () à OnEndPrinting()的顺序来编程的。对于我们使用单文档或者多文档的视图框架时这无疑是一种很好的方式。但是，在基于对话框的应用程序中，或者在很多打印要求不高的情况下（如只是简单的打印中间计算结果、报告程序异常等），我们就没有必要再绕上面的那么一个大弯子，可以直接利用 MFC 封装的打印对话框 CPrintDialog 来执行简单的打印任务。当然，如果你不怕烦，这种方式也可以执行很复杂的打印作业。

下面做一个简单的例子程序。利用 Visual C++ 6.0 的 AppWizard 生成一个对话框应用程序框架。然后，利用资源编辑器添加一个 Edit 控件，属性设置为：选择 Multiline、Want Return ,根据你的兴趣选择滚动条有关的项，绑定成员变量 m_strPrintString。使“确定（OK）”按钮不接受程序的“Enter”键消息，Caption 改为“打印（Print）”。此键按下的处理函数如下：

```
if (!UpdateData(TRUE))

{
```

```
AfxMessageBox("数据交换有误!");

}

if (m_strPrintString==_T(""))

{

AfxMessageBox("请输入需要打印的文字?quot;);

return;

}

CPrintDialog dlg(FALSE,

PD_NOPAGENUMS|PD_NOSELECTION,

this);

if(dlg.DoModal() == IDOK)

{

CDC dc;

dc.Attach(dlg.GetPrinterDC());//把打印设备环境附加到 DC 对象

DOCINFO di;

di.cbSize = sizeof(DOCINFO);

di.lpszDocName = "SimplePrintDoc";

di.lpszOutput = NULL;

di.lpszDatatype = NULL;

di.fwType = 0;

dc.StartDoc(&di); //通知打印机驱动程序执行一新的打印任务

dc.StartPage();//通知打印机驱动程序打印新页
```

```
dc.SetMapMode(MM_HIENGLISH);//设置当前影射模式为：单位 0.001 英寸

//X 方向向右增加，Y 方向向上增加

CRect rectPrint(0, 0,

dc.GetDeviceCaps(HORZRES),//返回设备的.以毫米为单位的物理显示宽度

dc.GetDeviceCaps(VERTRES));//返回设备的.以毫米为单位的物理显示高度

dc.DPtoLP(&rectPrint);//设备物理单位转化为逻辑单位

dc.SetWindowOrg(0, -rectPrint.bottom);//设置原点

CFont font;

VERIFY(font.CreatePointFont(120,

"Arial",

&dc));//为 DC 创建字体

CFont* def_font = dc.SelectObject(&font);//保存现在的字体

dc.SetTextAlign(TA_TOP|TA_LEFT);

CString s = m_strPrintString; //要打印的字符串（注意有长度的限制）


s += "\n"; //必须增加一个换行符号（因为后面 while 循环的要求）

CString ss;

int index;

CSize size;

int x = 300;

int y = 9000;//注意原点位置和坐标增加的方向

size = dc.GetTextExtent("00", 2);//计算使用当前字体输出时文本所占大小、宽 度

while((index = s.Find("\n")) != -1)
```

```
{
ss = s.Left(index);
if(ss.Find("\r") != -1)//输入的字符串有回车符
ss = s.Left(index-1);
s = s.Mid(index+1);//取剩余的字符串  // AfxMessageBox("A"+ss+"A");
dc.TextOut(x, y, ss);//打印到缓冲区
y -= size.cy;
}

dc.SelectObject(def_font); //恢复原来的字体
font.DeleteObject();
dc.EndPage(); //通知打印机驱动程序页结束
dc.EndDoc();//通知打印机驱动程序打印完毕
DeleteDC(dc.Detach());
}
```

程序代码很简单，一看就明白，我想应该适合大多数的简单打印场合吧。另外，这个例子对于初学 Windows 编程的朋友来说，无疑也是一个理解 Windows 设备无关特性的好实例。

如果大家有更好更简便的、功能强大的方法请告诉我：vc.faq@263.net

MFC 打印操作

如果你曾试过在基于 C 的 SDK 程序中编码以产生打印输出，你会喜欢 MFC 打印支持。它虽不是一个彻底的解决办法，但它确实大大超过了 SDK 编码。由于打印支持是由 CView 派生的，文档/视图结构有必要关注烦琐的程序内务。

一 设备无关性

设备需要一个描述表用作绘画的逻辑画布。正如显示有一个显示设备描述表一样，打印机也要有一个打印设备描述表。当使用图形函数时，MFC 使用同一代码在显示设备描述表上和打印机设备描述表上绘画。这种两重性是通过 CView::OnDraw()方法来实现的。传入 CView::OnDraw()的设备描述表指针可以来自表示两个不同设备的任一地方，这种安排为 MFC 的设备无关打印及打印预览提供了多种手段。

打印循环 CView::OnFilePrint()

MFC 使用 8 种主要方法作为其"打印引擎"，打印过程的每一步包含了对一种 CView 方法的调用，见下表

CView 打印方法

方法	说明
OnFilePrint()	运行打印循环的主打印方法
OnPreparePrinting()	调用 DoPreparePrinting()以显示 Print 对话框
DoPrepareprinting()	显示 Print 对话框
OnBeginPrinting()	分配用于在打印机 DC 上绘画的 GDI 资源的位置
OnPrepareDC()	在 OnPrint()前由 MFC 调用。如同映像方式，该方法应位于设置 DC 属性处
OnPrint()	用打印机 DC 调用 OnDraw()
OnDraw()	把文档数据再现于打印或打印预览方式的打印机 DC
OnEndPrinting()	当打印结束后由 MFC 调用，使用该方法释放任被特殊分配的特定打印机的 GDI 资源

为使得 MFC 能响应打印命令，必须定义一个消息映像项，它使预定义的 MFC 值 ID_FILE_PRINT 与控制打印过程的 CView::OnFilePrint()方法相关联。该方法调用其他 CView 帮助方法来使大量的打印过程自动化。可以直接用代码调用 CView::OnFilePrint().或者，MFC 调用它以响应具有 ON_FILE_PRINT 值的任何命令消息（如菜单上的打印命令）。

因为 OnFilePrint()是保护类成员，该项必须驻留在视图类的消息映像中，如下述：

```
BEGIN_MESSAGE_MAP(CMyView,CView)

...

//Required for printing

ON_COMMAND(ID_FILE_PRINT,CView::OnFilePrint)

END_MESSAGE_MAP()
```

准备打印

为了对 MFC 应用程序的打印及打印预览提供支持，仅有一个方法你必须重新设计：CView::OnPreparePrintinf()，该虚拟方法在打印或预览文档前由 MFC 调用，其原型如下：

virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);

pInfo 参数是指向包含当前打印作业信息的 CPrintInfo 对象的指针。

注意：如果打印作业被用户在结果打印对话框中取消，则 CView::OnPreparePrinting()方法调用 CView::DoPreparePrinting()方法并返回零值。

CPrintInfo 类结构

---

strut CPrintInfo //printing information structure

{

CPrintInfo();

~CPrintInfo();

CPrintDialog* m_pPD; //pointer to print dialog

BOOL m_bPreview; //TRUE if in preview mode

BOOL m_bDirect; //TRUE if bypassing Print Dialog

BOOL m_bContinuePrinting; //set to FALSE to end printing

UINT m_nCurPage; //Current pages

UINT m_nNumPreviewPages;//Desired number of preview pages

CString m_strPageDesc; //Format string for page number display

LPVOID m_lpUserData; //pointer to user created struct

CRect m_rectDraw; //rectangle defining usable page area

void SetMinPage(UINT nMinPage);

void SetMaxPage(UINT nMaxPage);

```
UINT GetMinPage() const;

UINT GetMaxPage() const;

UINT GetFromPage() const;

UINT GetTopPage() const;

}
```

CPrintInfo 类数据成员

数据成员	说明
m_pPD	指向作为 Print 对话框的 CPrint Dialog 对象的指针
m_bDirect	指示文档是否正在直接打印的标志，它旁路 Print 对话框
m_bPreview	指示文档是否正在预览的标志
m_bContinuePrinting	指示 MFC 是否应该保持在打印循环内的标志
m_nCurPage	当前正在打印的页数
m_nNumPreviewPages	确定多少页应该显示在打印预览窗口：1 或 2
m_lpUserData	指向应用程序定义结构的指针
m_rectDraw	定义打印机 DC 上的当前可用页区域的矩形框
m_strPageDesc	含有页号显示信息的格式字符串

CPrintInfo 类方法

方法	说明
SetMinPage()	指定文档的第一个页数（通常为 1）
SetMaxPage()	指定文档的最末页码
GetMinPage()	获取文档的第 1 页码
GetMaxPage()	获取文档的最末页码
GetFromPage()	获取待打印的第 1 页码
GetToPage()	获取待打印的最末页码

Print 对话框

CView::OnPreparePrinting()方法的默认实现如下所示：

```
BOOL CDocView2View::OnPreparePrinting(CPrintInfo *pInfo)

{

//default preparation

return DoPreparePrinting(PInfo);

}
```

此默认处理仅调用 CView::DoPreparePrinting()方法，后者自动处理调用，而不考虑 Pring 对话框。

该对话框从获得指向 CPrintInfo 对象的指针（PInfo)的 CView::DoPreparePrinting()方法中被调用。CView::DoPreparePrinting()方法具有以下原型：

BOOL DoPreparePrinting(CPrintInfo* pInfo);

MFC 使用由用户在 Print 框输入的值来填充各种 CPrintInfo 类数据成员，然后，Print 对话框作为 CPrintInfo 对象成员（pInfo->m_pPD)，创建打印设备描述表作为其 hDC 成员，方法如下：

```
if(pInfo->m_pPD->m_pd.hDC==NULL)

{

//call CreatePrinterDC if DC was not created by above

if(pInfo->m_pPD->CreatePrinterDC()==NULL)

return FALSE;

}
```

提示：在调用 CView::DoPreparePrinting()之前，设置 CPrintInfo 类成员以控制 Print 对话框上的显示值。你指定的值的出现在相应的对话框控件中。

如果 m_bPreview 成员为 FALSE，则仅显示 Print 对话框；在打印预览过程中该对话框不出现。可以用改变 m_bPreview 成员值的办法控制对话框的出现，旁路实际打印作业的 Print 对话框。打印设置描述表被创建及在 CView::DoPreparePrinting()中的 CPrintInfo 数据被初始化之后，其调用方法 CView::OnPreparePrinting()即退出。MFC 的打印例程的下一站是 CView::OnBeginPrinting()方法。

开始打印作业

CView::OnBeginPrinting()方法的首要任务是使合法的打印 DC 对你的应用程序是可用的。重新设计该方法以提供基于打印机 DC 特性的初始化。该方法的原型表示如下：

```
virtual void OnBeginPrinting(CDC* pDC,CPrintInfo* pInfo);
```

打印或打印预览作业开始时，MFC 调用该方法，后者在默认的 CView 实现中不做什么工作。通过重新设计 OnBeginPrint()的途径，你可分配在打印过程中可能需要的任何 GDI 资源。如果你分配 OnBeginPrint()

中的 GDI 对象，例如，选定几把刷子进入打印 DC，可从优先的 CView::OnPrint()方法内部来选，打印实际在其内部进行（通过 CView::OnDraw).

从 OnBeginPrint()返回后，OnFilePrint()方法开始主打印循环，在此循环内，文档的每一页都初始化并（以长度）被打印。

准备打印设备描述表

在设备描述表（即在 OnDraw()方法中）上使用任何图形方法之前，调用 OnPrepareDC()方法以使应用程序调整 DC 的特性。该方法的原型表述如下：

virtual void OnPrepareDC(CDC* pDC,CPrintInfo* pInfo=NULL);

该方法通常被重新设计以设置新的映像方式或 DC 的某些其他特性。

打印作业

调用 OnPrepareDC()之后，MFC 调用 CView::OnPrint()方法来打印或预览每一个文档页：

virtual void OnPrint(CDC* pDC,CPrintInfo* pInfo);

OnPrint()使用 CPrintInfo 结构的 m_nCurPage 成员来确定打印的当前页，然后 OnPrint()调用 OnDraw()来绘制打印机 DC 上的页。

文档的每一页调用一次 OnPrint()方法。如果有多页文档需重新设计此涵函数以实现页打印逻辑。这需要通过操作视窗原点仅绘制 OnDraw()的当前页，从而对打印循环的每次迭代视图会移动一页，它也可使你仅打印褶边如页眉及页脚。

这里有一个最小的已重新设计的 OnPrint()方法，它最后调用具有打印 DC 作为参数的 OnDraw()方法：

```
void CSomeView::OnPrint(CDC *pDC, CPrintInfo *pInfo)

{

OnDraw(pDC);

}
```

如果文档超过一页，打印循环的下一次迭代从 OnPrint()返回后开始。

清除

文档的全部页码打印完后，打印循环结束（或打印预览结束），MFC 调用可以重新设计的 CView::OnEndPrinting()方法来释放分配在 OnBeginPrinting()中的任何 GDI 资源。该方法的原型如下所示：

virtual void OnEndPrinting(CDC* pDC,CPrintInfo* pInfo);

注意：如果打印机图像不同于显示图像，且需要附加打印机特定的 GDI 资源时，通常才使用分配和释放 OnBeginPrinting()方法中的 GDI 资源。