

```

char *fmts [MAXFMTS] =
{
    " ", // dummy so CF_TEXT (1) can be used to access "CF_TEXT", etc.
    "TEXT", // CF_TEXT
    "BITMAP", // CF_BITMAP
    "METAFILE", // CF_METAFILEPICT
    "SYLK", // CF_SYLK
    "DIF", // CF_DIF
    "TIFF", // CF_TIFF
    "EMTEXT", // CF_OEMTEXT
    "DIB", // CF_DIB
    "PALETTE", // CF_PALETTE
};

void BackPack::WMINITMENU (WinAppMsg& m)
{
    CheckMenuItem (GetSystemMenu (hWnd, 0), SC_AUTOLOAD, (MF_BYCOMMAND | (bAutoLoad
    ? MF_CHECKED : MF_UNCHECKED)));
    CheckMenuItem (GetSystemMenu (hWnd, 0), SC_AUTOPACK, (MF_BYCOMMAND | (bAutoPack
    ? MF_CHECKED : MF_UNCHECKED)));
}

void BackPack::WMCHANGECHAIN (WinAppMsg& m)
{
    // viewer list has changed
    if (m.wParam == clp.NextViewer) // our 'next' is removing itself
        clp.NextViewer = LOWORD (m.lParam);
    else if (clp.NextViewer) // not our 'next', tell our 'next'
        SendMessage (clp.NextViewer, m.msg, m.wParam, m.lParam);
}

void BackPack::AutoLoad (void)
{
    bAutoLoad = !bAutoLoad;
}

// if Auto Load is selected and app name is not in LOAD= list
if (bAutoLoad && !strchr (outbuf, AppName))
{
    strcpy (outbuf, AppName);
    strcat (outbuf, " ");
    p = &LastChar (outbuf);
    p++;
    WriteProfileString ("windows", "load", "", p, sizeof (outbuf) - (p - outbuf));
}
// if Auto Load is not selected and app name is in LOAD= list
if (!bAutoLoad && (p = strstr (outbuf, AppName))
{
    *p = '\0';
    p += strlen (AppName);
    strcat (outbuf, p);
    WriteProfileString ("windows", "load", "", outbuf);
}

void BackPack::ResizeFileHeader (void)
{
    WORD oldMax = arrayMgr.MaxElements ();
    if (oldMax < MaxItems)
    {
        char dir [MAXPATHLEN];
        GetWindowsDirectory (dir, sizeof (dir));
        strcat (dir, "\\");
        strcat (dir, szBackPackTemp);
        File TempFile (dir);
    }
}

```

```

TempFile.Create (0);
arrayMgr.UpdateOffsets (FirstRecord () - firstOffset); // create the file
UpdateHeader (TempFile); // write the new header to the f
}

TempFile.Append (StackFile, firstOffset); // tack on rest of org file
TempFile.Close (); // close 'em
StackFile.Close (); // delete the old guy
TempFile.Delete (); // rename the disk file
StackFile.Open (OF_READWRITE); // re-open it
}
else
    UpdateHeader (StackFile); // if array shrunk
}

void BackPack::PackFile (void)
{
    #ifdef OLD
    DWORD curOffset = FirstRecord ();
    WORD compress = FALSE;
    for (int i = arrayMgr.NumElements () - 1; i >= 0; i--) // for each item
    {
        if (arrayMgr.IsDupe (i, arrayMgr [i] -> offset)) // calculate it's true size
            continue; // if already copied,
        DWORD size = arrayMgr [i] -> size + sizeof (CLIP_OBJECT); // don't copy it
        if (arrayMgr [i] -> format == CF_BITMAP)
            size += sizeof (BITMAP);
        if (arrayMgr [i] -> offset != curOffset) // if not at the right offset
        {
            StackFile.CopyBytes (curOffset, arrayMgr [i] -> offset, size); // copy it to new location
            arrayMgr [i] -> offset = curOffset; // reset offset in header
            compress = TRUE;
        }
        curOffset += size; // set for next location in file
    }

    if (!compress)
        return;
    UpdateHeader (StackFile); // write the header
    char dir [MAXPATHLEN];
    GetWindowsDirectory (dir, sizeof (dir));
    strcat (dir, "\\");
    strcat (dir, szBackPackTemp); // create C:\BackPack.$$$ name
}
File TempFile (dir); // create temp file object
TempFile.Create (0); // create temp file object
TempFile.Append (StackFile, 0, curOffset); // close 'em
TempFile.Close ();
StackFile.Close (); // delete the old guy
TempFile.Delete (); // rename the disk file
StackFile.Open (OF_READWRITE); // re-open it
}
else
    strcat (dir, szBackPackTemp); // create C:\BackPack.$$$ name
File TempFile (dir); // create temp file object
TempFile.Create (0); // create temp file object
DWORD tempOffset = FirstRecord ();
HANDLE hMem = GlobalAlloc (GHND, 0xffff);
LPSTR buffer = GlobalLock (hMem);
TempFile.Append (StackFile, 0, tempOffset, buffer);
for (int i = 0; i < arrayMgr.NumElements (); i++) // for each item in array
{
    if (arrayMgr.IsDupe (i)) // if dupe

```

```

        if (!arrayMgr.IsFirstDupe(i)) // but a subsequent one,
            continue;
        // calculate it's true size
        DWORD size = arrayMgr[i]->size + sizeof(CLIPOBJECT);
        if (arrayMgr[i]->format == CF_BITMAP)
            size += sizeof(BITMAP); // if bitmap, add size of BITMAP

        TempFile.Append(StackFile, arrayMgr[i]->offset,
            size+arrayMgr[i]->offset, buffer);

        if (arrayMgr.IsDupe(i)) // if it's a dupe
            arrayMgr.UpdateDups(i, tempOffset); // update all of them
        else
            arrayMgr[i]->offset = tempOffset; // set for next location in Tempfile
            tempOffset += size; // close 'em

        TempFile.Close();
        StackFile.Close();
        StackFile.Delete(); // delete the old guy
        TempFile.Rename(StackFile.GetFileName()); // rename the disk file
        StackFile.Open(OF_READWRITE); // re-open it
    #endif
}

void BackPack::Init(void)
{
    myCursor(WAIT);
    // initialize list box here...
    if (newFile)
    {
        arrayMgr.Init(MaxItems);
        UpdateHeader(StackFile);
        newFile = FALSE;
    }
    else
    { // read array table from existing file
        // go to array table offset
        StackFile.ReadAt(0L, sizeof(header), &header);
        arrayMgr.Init(header.MaxElements); // create array and init maxEl
        arrayMgr.SetNumElements(header.NumElements); // init numEl
        arrayMgr.LastOffset(header.LastOffset); // init lastoffset
        // read the array table into array
        StackFile.ReadAt(0L+sizeof(header),
            (WORD)arrayMgr.Size(), arrayMgr.Array());

        if (arrayMgr.MaxElements() != MaxItems) // if array needs resizing
            ResizeFileHeader(); // do it.
        if (bAutoPack) // if compression turned on
            PackFile(); // do it.
        InitListBox();
    }
    clp.JoinViewers(); // become part of Clipboard chain
    myCursor(ARROW);
}

```

/*

COMPQAQ and the COMPQAQ logo are registered trademarks of the
COMPQAQ Computer Corporation.

*/