

PalmRamDisk Developer Addendum

Copyright (c) 2002 by HandWatch, Inc.

Written by Mark Chao-Kuang Yang

Background

In the past, developers who are new to PalmOS have to learn different PalmOS libraries and APIs to manage files. Namely Database and Resource Manager, File Stream Manger, and lately, VFS (Virtual File System). Though database architecture provides a good performance for PalmOS, its hierarchy-less structure, 64KB memory allocation limit, and in-place editing have given developers a difficult time to understand them. Not only it takes more time to port the existing libraries that need access to files, but the data conversion has to take place at desktop in order to be stored internally on Palm devices.

Although external memory cards can store any type of files, there is no way of doing so with built-in memory. So far, the main file formats supported by PalmOS are Palm Database (.pdb) and Palm Resource (.prc) files. Even so, no directory hierarchy is supported. Record category is the feature that's closest to directory hierarchy. However, only 16 categories are allowed in a database, and only single level of depth.

With PalmRamDisk, this is going to change. This is the first time for PalmOS to store any files in internal memory with directory hierarchy support. Use Palm VFS API, which is similar to C/C++ file I/O library, to access both internal and external files. This makes it easy to have consistant file API across the source code. Developers can use VFS API to access files on both built-in memory and external memory cards.

Use VFS with PalmRamDisk

While VFS makes it easy to access files stored on external memory cards, there is no easy way to use the same API to access files stored on internal memory. This is where PalmRamDisk comes in. By 'faking' a memory card using internal memory, VFS recognizes the ram disk as one of the external memory card. Thus, VFS can also be used with files stored on RAM disk. To VFS, RAM disk is just another external memory card, though the data is actually stored internally. Via the API function calls similar to C/C++ file I/O library, such as VFSFileOpen, VFSFileRead, and VFSFileWrite, it is extremely easy for developers to develop a file based application. It also makes the source codes consistant for both internal and external memory storage.

To access files on the RAM disk, use VFS to search for the volume that resides on RamDisk as the following:

```
#include <ExpansionMgr.h>
#include <VFSMgr.h>

// 10 should be large enough to contain all available volumes
#define kMaxVolumes      10

UInt16 QueryAllVolumes (UInt16* volRefList, const UInt16 maxVols);
Boolean IsRAMDisk (UInt16 volRef);
UInt16 FindRAMDiskVolRef ();
```

```

UInt16 QueryAllVolumes (UInt16* volRefList, const UInt16 maxVols)
{
    // query all volumes
    Err err = errNone;

    UInt16 i          = 0;
    UInt32 iterator    = vfsIteratorStart;

    while (i < maxVols && iterator != vfsIteratorStop)
    {
        UInt16 volRef = 0;

        err = VFSVolumeEnumerate (&volRef, &iterator);
        if (err == errNone)
        {
            volRefList[i++] = volRef;
        }
    }

    return i;
}

Boolean IsRAMDisk (UInt16 volRef)
{
    VolumeInfoType volInfo;
    VFSVolumeInfo (volRef, &volInfo);
    return (volInfo.mediaType == expMediaType_RAMDisk);
}

UInt16 FindRAMDiskVolRef ()
{
    UInt16 volRefList[kMaxVolumes];
    UInt16 numVols      = 0;
    UInt16 i;

    numVols = QueryAllVolumes (volRefList, kMaxVolumes);

    for (i=0;i<numVols;i++)
    {
        if (IsRAMDisk (volRefList[i]))
            return volRefList[i];
    }

    return 0;
}

```

Use FindRAMDiskVolRef() to find the volume reference to RAM disk:

```

UInt16 ramDiskVolRef = FindRAMDiskVolRef();

```

ramDiskVolRef is then passed to any volume related functions in VFS API:

```
Err err = errNone;  
FileRef fileRef = NULL;  
err = VFSFileOpen (ramDiskVolRef, "/Palm/readme.txt",  
..., ...);
```

Palm Database and Resource Files

VFS API also provides function calls to access plain Palm Database files (.pdb) and Resource files (.prc). If your applications still need to access those files on RAM disk, use VFSFileDB####, VFSExportDatabase#### and VFSImportDatabase#### calls. You can then access the database records and resources from the file.

Conclusion

For new Palm developers from PC Desktop, VFS API is much easier to understand and use. It cuts time to learn how to save and read from files. Most importantly, it saves time and efforts to convert existing data to conform Palm's native format and porting libraries.

Contact Information

Should you have any questions regarding how to develop applications to utilize PalmRamDisk, please email to [**handwatch@handwatch.com**](mailto:handwatch@handwatch.com).

Legal Information

PalmRamDisk is **Copyright (c) by HandWatch, Inc.**
All logos and trademarks belong to their rightful owner.