

The Archive Manager

Explanation of the Archive Manager

Version 1.0.0

The contents of this document are strictly confidential and the document should be handled accordingly.

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo of America Inc. and/or Nintendo Company Ltd. and are protected by Federal copyright law. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Contents

1	Introduction	5
2	Archives	6
3	Outline of the Archive Manager	7
3.1	How to Use an Archive	7
3.2	Accessing Files in the Archive	7
3.3	Pointers to Files in the Archive	7
4	Features of the Archive Manager	8
4.1	The Functions of the Archive Manager	8
4.2	Mounting an Archive	8
4.2.1	Archive Identifier	8
4.3	Getting the Pointer to a File	9
4.3.1	Using a File ID to Access a File	9
4.3.2	Using a Path Name to Access a File	9
4.4	Opening a File Inside an Archive	10
4.5	Unmounting an Archive	10

Tables

Table 4-1 Functions of The Archive Manager	8
--	---

Revision History

Version	Revision Date	Description
1.0.0	1/5/2005	Changed an instance of "NITRO" to "Nintendo DS."
0.1.0	5/24/2004	Initial version.

1 Introduction

The Archive Manager is provided in the NITRO-System library in order to help Nintendo DS applications handle archives that have been created using the NITRO-System general-purpose archiver `nnsarc`.

2 Archives

The NITRO-System general-purpose archiver `nnsarc.exe` collects a number of files into a single file called an archive. In addition to files, this archive file can also contain directory information, allowing the files to be accessed by specifying a file ID (an index value) or a path name.

3 Overview of the Archive Manager

The Archive Manager is built on the NITRO-SDK's ROM file system. By loading the archive binary into memory and then registering it in the ROM file system, the Archive Manager can access the files in the archive using the NITRO-SDK's ROM file system API.

3.1 How to Use an Archive

Here is the procedure for using an archive:

1. Use the ROM file system to read the archive binary from ROM into RAM.
2. After the archive binary is loaded into RAM, mount it into the ROM file system.
3. Get the address of the file stored in the archive and then use the data.
4. When the data in the archive is no longer needed, unmount the archive from the ROM file system.
5. Delete the archive binary from RAM.

3.2 Accessing Files in the Archive

The Archive Manager can access files in the archive by using either a path name or a file ID (an index value).

Using a file ID to access a file requires less search time than using a path name, but be aware that the file ID value changes whenever the archive's file structure changes.

The use of path names requires the creation of a Filename Table inside the archive. The `nnsarc` archiver creates this Filename Table by default.

3.3 Pointers to Files in the Archive

The files in an archive that has been mounted to the ROM file system can be accessed using the ROM file system's API. Files can be read from the archive in the same way that files are read from ROM using the ROM file system.

However, since an archive that has been mounted to the ROM file system is entirely loaded into RAM, all of the files in that archive exist in RAM. The Archive Manager has an API for getting the addresses where the files are located in the archive, and it can use this API to access the files inside the archive without using the ROM file system API to read the files (i.e., without copying the files).

Note that if you are using a pointer to directly access a file in the archive, you cannot delete the archive from memory until after you are finished using that file.

4 Features of the Archive Manager

4.1 The Functions of the Archive Manager

The Archive Manager is equipped with the following five functions:

Table 4-1 Functions of The Archive Manager

Function	Action
<code>NNS_FndMountArchive()</code>	Mounts archive to ROM file system.
<code>NNS_FndUnmountArchive()</code>	Unmounts a mounted archive.
<code>NNS_FndGetArchiveFileByName()</code>	Gets the address of a file specified with a path name.
<code>NNS_FndGetArchiveFileByIndex()</code>	Gets the address of a file specified with a file ID.
<code>NNS_FndOpenArchiveFileByIndex()</code>	Opens a file specified with a file ID.

4.2 Mounting an Archive

In order to access the files in the archive, the archive must first be mounted to the ROM file system. This is done using the `NNS_FndMountArchive()` function, as follows:

```
void* NNS_FndMountArchive(  
    NNSFndArchive* archive, const char* arcName, void* arcBinary);
```

The `NNS_FndMountArchive()` function gets the pointer to the archive binary that has been loaded into RAM and registers this in the ROM file system. It also initializes the specified `NNSFndArchive` structure. Once this is done, you specify the pointer to the `NNSFndArchive` structure when specifying the archive you want to process with the Archive Manager.

4.2.1 Archive Identifier

To mount an archive, specify an archive identifier. This identifier can have up to three alphanumeric characters. The archive identifier must be unique among mounted archives.

The archive identifier is used with functions like `FS_OpenFile()` when specifying a path name in order to indicate which archive in that path to operate on. The archive identifier is given with a colon at the front of the path, as follows:

```
"/data/scen1/screen.dat" // Path to the file in the ROM file system  
"ARC:/data/scen1/screen.dat" // Path to the file in the archive with the  
                               identifier "ARC"
```


4.3 Getting the Pointer to a File

The Archive Manager operates on the assumption that the entire archive binary has been loaded into RAM. In other words, it assumes that all files inside that archive are present in RAM. As a result, there is no need to use the ROM file system's `FS_Read()` function to read files, except when data needs to be duplicated.

The Archive Manager has two ways to get the address of the location of a file in the archive: it can use a file ID, or it can use a path name.

4.3.1 Using a File ID to Access a File

To use a file ID to get the address of a file in the archive, use the `NNS_FndGetArchiveFileByIndex()` function, as follows:

```
void* NNS_FndGetArchiveFileByIndex(NNSFndArchive* archive, u32 index);
```

Each file stored in the archive is allocated a file ID, starting in order from 0. By specifying a file's file ID, you can get the address of the location of that file in the archive.

Using a file ID is an extremely fast and easy way to access a file. However, file IDs change whenever the file structure of the archive is altered, so you need to re-specify the file IDs whenever this happens. To simplify this process, the archiver `nnsarc` can be set with the `-i` option to create a C-language file ID definition header file when it creates the archive. Placing this file ID definitions header file in an include statement in the source file enables you to specify files using their constant names. If you do this, all you need to do is recompile the archive source when the file structure of the archive is altered.

4.3.2 Using a Path Name to Access a File

To use a path name to get the address of a file in the archive, use the `NNS_FndGetArchiveFileByName()` function, as follows:

```
void* NNS_FndGetArchiveFileByName(const char* path);
```

Since this function can indicate an archive by specifying a path name, there is no need to specify the `NNSFndArchive` structure.

This procedure of specifying a path name is readily understandable and convenient for the programmer, but it adds the burden of processing a character string.

You cannot use the `NNS_FndGetArchiveFileByName()` function on an archive that has been created by `nnsarc` with the `-n` option because the archive has an empty Filename Table. For such archives, the only way to access the files is by using their file IDs.

4.4 Opening a File In an Archive

To open a file inside an archive specified with a path name, use the ROM file system's API function `FS_OpenFile()`. To open a file inside an archive specified with a file ID, use the Archive Manager's API function `NNS_FndOpenArchiveFileByIndex()`.

```
BOOL NNS_FndOpenArchiveFileByIndex(NNSFndArchive* archive, u32 index);
```

You use this function because the ROM file system's API function `FS_OpenFile()` requires the `FS_FileID` structure as an argument, and thus cannot open a file with the file ID.

Use the ROM file system's API function `FS_CloseFile()` to close an open file, regardless of whether the file was opened using a path name or a file ID.

4.5 Unmounting an Archive

Unmount the archive when none of the files in the archive are required anymore. To unmount an archive, use the `NNS_FndUnmountArchive()` function, as follows:

```
void* NNS_FndUnmountArchive(NNSFndArchive* archive);
```


© 2004-2005 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed or loaned in whole or in part without the prior approval of Nintendo Co. Ltd.