

# アーカイブマネージャ

## アーカイブマネージャについての説明

2008-07-14

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、**厳重な取り扱い、管理を行ってください。**

## 目次

---

1	はじめに .....	4
2	アーカイブ .....	4
3	アーカイブマネージャの概要 .....	4
3.1	アーカイブ利用の手順 .....	4
3.2	アーカイブ内のファイルへのアクセス .....	4
3.3	アーカイブ内のファイルへのポインタ .....	5
4	アーカイブマネージャの機能 .....	5
4.1	アーカイブマネージャの関数 .....	5
4.2	アーカイブのマウント .....	5
4.2.1	アーカイブ識別名 .....	5
4.3	ファイルへのポインタの取得 .....	6
4.3.1	ファイルIDによるアクセス .....	6
4.3.2	パス名によるアクセス .....	6
4.4	アーカイブ内のファイルのオープン .....	7
4.5	アーカイブのアンマウント .....	7

## 表

---

表 4-1	アーカイブマネージャの関数 .....	5
-------	---------------------	---

## 改訂履歴

改訂日	改訂内容
2008-07-14	関数名や型名の間違いを修正。
2008-05-30	NITRO-System の名称変更による修正 (NITRO-System から TWL-System に変更)。
2008-04-08	・改訂履歴の書式を変更。 ・TWL-SDK に対応。
2005-01-05	NITRO という表記をニンテンドーDS に統一。
2004-05-24	初版。

# 1 はじめに

TWL-System ライブラリでは、TWL-System で提供している汎用アーカイバ `nnsarc` を使用して作成されたアーカイブを TWL やニンテンドーDS のアプリケーションで容易に扱えるようにする為に、アーカイブマネージャを提供しています。

## 2 アーカイブ

アーカイブは、いくつかのファイルが1つに結合されたファイルで、TWL-System 汎用アーカイバ `nnsarc.exe` により作成されます。アーカイブファイルにはファイルだけでなく、ディレクトリ情報も含めることが可能であり、ファイルID (インデックス値) またはパス名を指定することによりアーカイブ内の個々のファイルにアクセスする事ができます。

## 3 アーカイブマネージャの概要

アーカイブマネージャは、TWL-SDK の ROM ファイルシステムの上に構築されています。アーカイブマネージャでは、メモリ上に読み出したアーカイブバイナリを ROM ファイルシステムに登録することにより、TWL-SDK の ROM ファイルシステムの API を使用したアーカイブ内のファイルへのアクセスを可能としています。

### 3.1 アーカイブ利用の手順

以下にアーカイブを利用する場合の手順を示します。

- (1) アーカイブバイナリを ROM ファイルシステムを使用して ROM から RAM に読み出す。
- (2) RAM に読み出したアーカイブバイナリを ROM ファイルシステムにマウントする。
- (3) アーカイブ内でファイルが格納されているアドレスを取得し、データを利用する。
- (4) アーカイブ内のデータが不要になれば、アーカイブを ROM ファイルシステムからアンマウントする。
- (5) アーカイブバイナリを RAM から削除する。

### 3.2 アーカイブ内のファイルへのアクセス

アーカイブマネージャでは、パス名またはファイルID (インデックス値) を指定することにより、アーカイブ内のファイルにアクセスする事が出来ます。

ファイルIDを使用してファイルにアクセスする方が、パス名を使用する場合よりファイルの検索に費やされる時間が少なく済みますが、ファイルIDはアーカイブのファイル構成が変わる度に値が変化しますので、注意が必要です。

パス名を用いる場合は、アーカイブ内にファイル名テーブルが作成されている必要があります。なお、アーカイバ `nnsarc` では、デフォルトでこのファイル名テーブルを作成するようになっています。

### 3.3 アーカイブ内のファイルへのポインタ

ROM ファイルシステムにマウントされたアーカイブ内のファイルには、ROM ファイルシステムの API を用いてアクセスする事が可能です。ROM 内のファイルを ROM ファイルシステムを用いて読み出すのと同じ方法で、アーカイブからファイルを読み出す事が可能です。

しかし、ROM ファイルシステムにマウントされているアーカイブは、その全体が RAM に読み出されていますので、アーカイブに格納されているファイルは、全て RAM 上に存在していることになります。

このことから、アーカイブマネージャでは、アーカイブ内のファイルが置かれているアドレスを取得するための API を用意しています。この API を用いますと、ROM ファイルシステムの API を使ってファイルを読み出さなくても（ファイルを複製しなくても）、アーカイブ内のファイルにアクセスすることが出来ます。

なお、ポインタによりアーカイブ内のファイルに直接アクセスする場合には、そのファイルの利用が終了するまで、アーカイブをメモリから削除する事が出来ませんのでご注意ください。

## 4 アーカイブマネージャの機能

### 4.1 アーカイブマネージャの関数

アーカイブマネージャには、下記の5つの関数が用意されています。

表 4-1 アーカイブマネージャの関数

関数	機能
NNS_FndMountArchive()	アーカイブを ROM ファイルシステムにマウントします。
NNS_FndUnmountArchive()	マウントされているアーカイブをアンマウントします。
NNS_FndGetArchiveFileByName()	パス名により指定されたファイルのアドレスを返します。
NNS_FndGetArchiveFileByIndex()	ファイルIDにより指定されたファイルのアドレスを返します。
NNS_FndOpenArchiveFileByIndex()	ファイルIDにより指定されたファイルをオープンします。

### 4.2 アーカイブのマウント

アーカイブ内のファイルにアクセスする為には、最初にアーカイブを ROM ファイルシステムにマウントしなければなりません。アーカイブのマウントには、NNS\_FndMountArchive()関数を用います。

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

この NNS\_FndMountArchive()関数は、RAM に読み出されているアーカイブバイナリへのポインタを受け取り、ROM ファイルシステムに登録すると共に、指定された NNSFndArchive 構造体を初期化します。以後、アーカイブマネージャに対して処理を行いたいアーカイブを指定する場合には、NNS\_FndMountArchive()関数で初期化された NNSFndArchive 構造体へのポインタを指定します。

#### 4.2.1 アーカイブ識別名

アーカイブをマウントする場合には、アーカイブ識別名を指定する必要があります。アーカイブ識別名は、3文字までの

英数字で指定します。アーカイブ識別名は、マウントされているアーカイブの中で、ユニークな名前である必要があります。

アーカイブ識別名は、FS\_OpenFile0等でパス名を指定する場合に、そのパスがどのアーカイブに対するものかを示すために使用します。アーカイブ識別名は、以下の様にパスの前にコロンを挟んで記述します。

```
"/data/scenel/screen.dat"      // ROMファイルシステム内のファイルへのパス。  
"ARC:/data/scenel/screen.dat"  // 識別名"ARC"のアーカイブ内のファイルへのパス。
```

## 4.3 ファイルへのポインタの取得

このアーカイブマネージャでは、アーカイブバイナリ全体が RAM 上に読み出されていることを前提としています。即ち、アーカイブに格納されているファイルは、全て RAM 上に存在していることになります。このため、データの複写が必要な場合を除いて、ROM ファイルシステムの FS\_Read0関数を用いてファイルを読み出す必要はありません。

アーカイブマネージャでは、アーカイブ内のファイルが置かれているアドレスを取得するために、2つのアクセス方法を提供しています。

### 4.3.1 ファイルIDによるアクセス

ファイルIDによりアーカイブ内のファイルのアドレスを取得する為には、NNS\_ FndGetArchiveFileByIndex0関数を用います。

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

アーカイブに格納されているファイルには、0番からの連続したファイルIDが付けられています。このファイルIDを指定することにより、ファイルが置かれているアドレスを取得します。

ファイルIDによるアクセスは非常にシンプルであり、高速ですが、アーカイブのファイル構成が変わる度に値が変化してしまいます。この為、アーカイブのファイル構成が変わった場合には、再度ファイルIDを指定しなおす必要が生じます。

これを容易にする為に、アーカイバ nnsarc では、アーカイブ作成時に -i オプションを指定することにより、C言語用のファイルID定義ヘッダファイルを作成する事が出来るようになっています。このファイルID定義ヘッダファイルをソースファイルにインクルードすることにより、定数名でファイルを指定することができます。この場合、アーカイブのファイル構成が変わった時には、そのアーカイブを使用しているソースをコンパイルし直すだけで済みます。

### 4.3.2 パス名によるアクセス

パス名によりアーカイブ内のファイルのアドレスを取得する為には、NNS\_ FndGetArchiveFileByName0関数を用います。

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

NNS\_ FndGetArchiveFileByName0関数では、指定されたパス名によりアーカイブが特定可能ですので、NNSFndArchive 構造体を指定する必要はありません。パス名による指定は、プログラマには大変判りやすく便利ですが、文字列処理が入りますので、多少の処理コストがかかります。

アーカイバ nnsarc で-n オプションを指定して作成されたアーカイブは、ファイル名テーブルが空になっていますので、NNS\_ FndGetArchiveFileByName0関数を用いることはできません。このようなアーカイブに対しては、ファイルIDによるアクセスのみが可能となります。

## 4.4 アーカイブ内のファイルのオープン

---

パス名を指定してアーカイブ内のファイルをオープンする場合は、ROM ファイルシステムの API である `FS_OpenFile()`関数を用います。対して、ファイルIDを指定してアーカイブ内のファイルをオープンする場合は、アーカイブマネージャの API である `NNS_FndOpenArchiveFileByIndex()`関数を用います。

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

ファイルシステムの API である `FS_OpenFileFast()`関数は、引数として `FSFileID` 構造体を必要としますので、アーカイブのファイルIDではオープンする事ができません。このため、アーカイブマネージャの API を用いてファイルをオープンします。

なお、オープンしたファイルのクローズには、パス名、ファイル・インデックスのどちらを使ってオープンした場合でも、ROM ファイルシステムの API である `FS_CloseFile()`関数を使用します。

## 4.5 アーカイブのアンマウント

---

アーカイブ内のファイルが全て不要になった時には、アーカイブをアンマウントします。アーカイブをアンマウントする為には、`NNS_FndUnmountArchive()`関数を使用します。

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

© 2004-2008 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。