フラッシュファイルシステム作成方法

QNX Software Systems Japan
ver 1.0　2010/06/23

1. はじめに

QNX では NOR 型フラッシュメモリをファイルシステムとして利用するドライバ、各種ツール、および特定デバイスに対応させたドライバを開発するために必要な開発環境を提供しています。

フラッシュファイルシステムは連続した領域としてアクセスする RAW パーティションアクセスと POSIX ファイルシステムとしてアクセス可能なファイルシステムアクセスの2つのアクセス方法を提供します。
ファイルを圧縮した状態で書き込んだ場合、読み出し時に透過的に解凍する機能も組み込まれています。

特定のハードウエアに関連する開発環境は BSP (Board Support Package) に含まれ、提供されるソースコードをカスタマイズすることにより、リファレンス環境以外のハードウェア上で動作するフラッシュファイルシステムドライバを構築可能です。

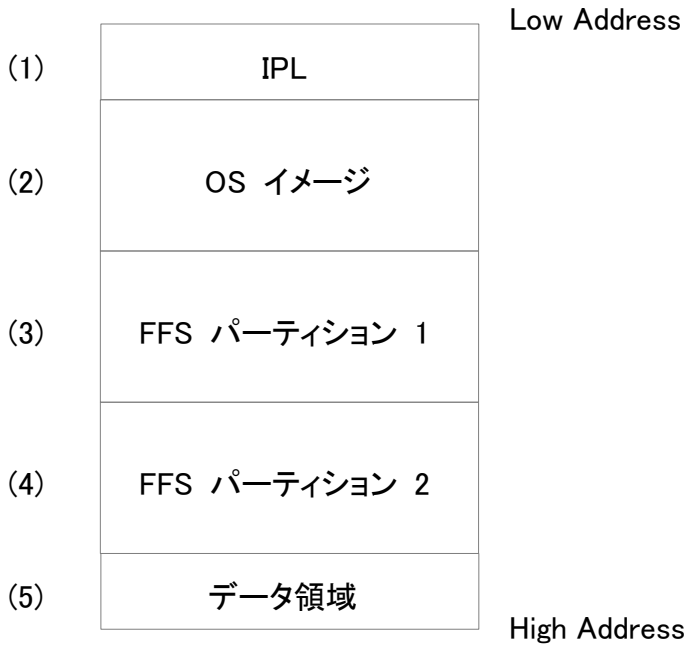
関連ドキュメント:

QNX Neutrino RTOS システムアーキテクチャ （日本語版および英語版）
　第 8 章 ファイルシステム - FF3 ファイルシステム

オンラインヘルプ
　QNX Neutrino Realtime Operating System:
　　Building Embedded System
　　Customizing the Flash FileSystem
　　Making an OS Image
　　Building flash filesystem image
　　Utilities Reference
　　devf-xxxx, flashctl, flshcmp, etc

2. フラッシュ内部

フラッシュメモリが CPU のアドレス空間に連続的にマップされている必要があります。
また、フラッシュメモリには下記のように IPL コード、OS イメージ、データ領域、フラッシュファイルシステムパーティションが混合することが可能です。



パーティションにはマーキングがされておりドライバから区切りが認識されますが、その他のエリアはドライバーからは区切りが認識されません。

この構成のフラッシュメモリをドライバで起動した場合、次のようなデバイスとして認識されます。

フラッシュメモリ全体	/dev/fs0
IPL + OS イメージ (1) + (2)	/dev/fs0p0
FFS パーティション 1 (3)	/dev/fs0p1
FFS パーティション 2 (4)	/dev/fs0p2

注1： フラッシュメモリー内の配置は CPU やハードウェアによって上記の様になるとは限りません。
また、IPL が逆になったり、間に挟まった場合でも扱えます。

注2： パーティションの区切りは、フラッシュメモリの消去単位の区切りと同じにします。

その他： spatch ツールを使用すると、/dev/fsxxx を通してフラッシュメモリの中身をダンプする事ができます。

3. マウント・アンマウント

フラッシュファイルシステムは、ブロックファイルシステムと同様にマウント・アンマウント操作が可能です。
通常は、フラッシュファイルシステムドライバを起動する時に自動マウントされます。

例として、前記のパーティションをアクティブにする場合について、自動マウントとプログラムでマウントする方法は以下の通りです。

[自動マウント]

```
# devf-xxxx
# ls /dev/fs*
/dev/fs0p0 /dev/fs0p1 /dev/fs0p2 /dev/fs0p3
# ls /fs*
/fs0 /fs1
# mount | grep flash
/dev/fs0p0 on /fs0 type flash
/dev/fs0p0 on /fs0/.cmp type flash
/dev/fs0p1 on /fs1 type flash
/dev/fs0p1 on /fs1/.cmp type flash
```

[手動マウント(上記と同等)]

```
# devf-xxxx -a
# mount -t flash /dev/fs0p1 /fs0
# mount -t flash /dev/fs0p2 /fs1
```

(-a は、自動マウント off のオプションです)

なお、マウント操作時にリードオンリーでマウントさせることにより、書き込みが行えないボリュームとして扱う事ができます。

```
# mount -t flash -r /dev/fs0p1 /fs0
```

4. リード・ライト・リクレイム (reclaim)

フラッシュファイルシステムは POSIX ファイルシステムのディレクトリー構造を持ち、fopen / fwrite / fread 等のファイルアクセス関数によって読み出し書き込みが可能です。

なお、フラッシュメモリーの特性により、書き込み時に空き領域を作成するリクレイム処理が必要となります。

ドライバを起動する時のオプションで、リクレイム処理をバックグラウンドのスレッドで行うようにすることが可能です。書き込み時間に問題がなければ通常はバックグラウンドのリクレイム処理はイネーブルにしません。

```
# devf-xxx -b5
```

5. リカバリー処理

書き込み途中での電源断、リセットなどの現象が発生すると、フラッシュへの書き込みが中断されます。
途中まで進んだ処理を継続しファイルシステムとしての整合性を取る処理をドライバーの起動時に挿入可能です。

```
# devf-xxxx -r
```

6. ファイルの圧縮と解凍

フラッシュファイルシステムには解凍機能が組み込まれており、圧縮されたファイルを圧縮マウントポイントに書き込んでおくと、読み出す際に解凍操作が自動的に行われます。

なお、ファイルシステムに書き込み際の透過的な圧縮はサポートされておらず、ファイルとして書き込む前に圧縮しておく必要があります。

```
# cp file /fs0

# flashcmp /fs0/.cmp/file
# ls -l /fs0/file
... 67996 Feb 25  2003 file
# ls -l /fs0/.cmp/file
... 46301 Feb 25  2003 file

/dev/fs0p1
       --> mount /fs0
              fs0/.cmp       圧縮ファイルマウントポイント
              fs0/           マウントポイント
```

圧縮ファイルマウントポイント以下に圧縮ファイルを書き込むと、同じディレクトリー階層で解凍したファイルとしてアクセスが可能です。

注: ドライバにはサイズと効率化のために解凍機能のみが組み込まれています。

7. フラッシュメモリへの書き込み

フラッシュメモリへの書き込みは、IPL, OS イメージ, パーティションイメージ, データをマージしたデータを直接フラッシュメモリライタで書き込んで実装する方法と、とりあえず OS イメージまで書き込み、OS を起動後にネットワークやディスクからコピーする方法が存在します。

フラッシュファイルシステムのイメージを作成するツールは mkefs です。
mkefs で作成したイメージをフラッシュファイルに書き込むと、ひとつのパーティションになります。

8. Posix ファイルシステムとして制限

フラッシュファイルシステムには、POSIX ファイルシステムとしては次の制限があります。

・ハードリンクを作成できない
　　-> シンボリックリンクを使用
・アクセス時間をサポートしない

９. バイナリ構成要素

QNX ではフラッシュファイルシステム用のドライバ、ツール類として下記のバイナリを提供しています。

devf-xxxxx	フラッシュファイルシステムドライバ
flashctl	フラッシュファイルシステムコントロール
flashcmp	フラッシュファイルシステム用ファイル圧縮ツール
mkefs	フラッシュファイルシステム構築ツール
mount, umount	マウント、アンマウントツール

１０. 組み込み方法

OS イメージに devf-xxxx (例 devf-bigsur)を組み込み、起動スクリプト中で起動します。

１１. 実際の利用例

実際に利用する例: SH4 - Platform (SolutionEngine)

参照オンラインドキュメント
Board Support Package/Hitachi Solution Engine (SH7750)

フラッシュメモリマップ

```
0x00000000 - 0x00000fff     IPL code
0x00001000 - 0x001fffff     OS image
0x00200000 - 0x003dffff     Flash File System
0x003e0000 - 0x003fffff     reserved
```

ドライバー起動

```
# devf-sengine -v
devf: fs0 socket Hitachi SH-4 S-Engine Flash
devf: chip total      = 1
devf: bus width       = 4
devf: chip interleave = 2
devf: fs0 array MBM29LV160-T U: 20 S: 020000
devf: fs0p0 raw U: 20
```

フラッシュ操作

(1) フラッシュクリア

```
# flashctl -p/dev/fs0 -o2M -l1920k -ve
Erasing device /dev/fs0
.............
devf: fs0p0 raw U: 20
```

offset 2M (0x200000) から 19200k (0x1e0000) バイトクリアするという意味です。
パーティションではなくフラッシュ全体中の位置で指定します。

注: IPL, OS イメージ等のエリアはクリアしないよう注意して下さい。

(2) パーティションフォーマット

ドライバーを再起動してパーティションアクセスを可能にします。

```
# slay devf-sengine
# devf-sengine
# ls /dev/fs*
/dev/fs0     /dev/fs0p0
```

パーティション /dev/fs0p0 は全体 /dev/fs0 と同じ領域を示します。

```
# flashctl -p/dev/fs0p0 -o2M -l1920k -f
```

フォーマットはパーティションにのみ適用できます。
ドライバーを再起動すると 3 つのパーティションに分かれます。
/dev/fs0p1 がフラッシュファイルシステム用のパーティションとなります。

```
# slay devf-sengine
# devf-sengine
# ls /dev/fs*
/dev/fs0 /dev/fs0p0 /dev/fs0p1 /dev/fs0p2
```

これ以降のパーティションのフォーマットは /dev/fs0p1 に対して行います。

```
# flashctl -p/dev/fs0p1 -f
```

(3) 自動マウント

何もオプションもつけなれければフラッシュファイルシステムが /fs0p1 にマウントされます。

(4) 圧縮ファイル書き込み

flashcmp ツールを使用してファイルを圧縮して .cmp ポイントへファイルを書き込みます。

```
# cp /shle/sbin/vi /tmp
# flashcmp /tmp/vi
# cp /tmp/vi /fs0p0/.cmp
# ls -l /fs0p0/.cmp /fs0p0
/fs0p0:
total 230
drwxrwxrwx  2 root      0               40 Jan 01 00:51 .cmp
-rwxrwxrwx  1 root      0           117216 Feb 26  2003 vi
/fs0p0/.cmp:
total 169
-rwxrwxrwx  1 root      0            86332 Feb 26  2003 vi
```

圧縮されていることが分かります。
/fs0p0/vi は /fs0p0/.cmp/vi を見せている仮想的なファイルとなります。

（5） mkefs によるフラッシュファイルイメージの作成と書き込み

イメージを mkefs で作成しフラッシュに書き込むことでフラッシュファイルシステムとして有効にできます。

例）

```
flash.build
-----------------------------------
[block_size=128K spare_blocks=1 min_size=1920K filter=flashcmp]
/shle/bin

# mkefs flash.build flash.efs
```

（6） ターゲットへのネットワーク経由での書き込み

Qnet を利用してコピーする例です。
NFS でも同様の方法が可能です。
起動イメージでネットワークまでが動作するようにしておく必要があります。

```
# devf-sengine
# ls /dev/fs*
/dev/fs0 /dev/fs0p0 /dev/fs0p1 /dev/fs0p2
                                        <- パーティションが存在することが前提
# flashctl -p/dev/fs0p1 -e              <- パーティション内を消去
# cp /net/Host/flash.efs /dev/fs0p1     <- イメージを net を通じてコピー
# slay devf-sengine
# devf-sengine                          <- 再起動
# ls /fs0p0
.cmp    bin                           <- ホストの/shle/bin 内のファイルが存在
```

（7） 直接書き込み

IPL + OS イメージ + フラッシュファイルシステムを合体させ、ROM に直接書き込みを行います。
ROM ライタ、JTAG デバッガー等による書き込みが考えられます。

書き換え後、リセット、電源投入によりシステムを起動します。