NILFS2 ソフトウェアツールの開発

関場 治朗

株式会社ウニークス

E-mail: jir@unicus.jp

あらまし Linux でファイルシステムはカーネル内の VFS とブロックデバイスレイヤーの間に作られたソフトウェアモジュールである．様々なファイルシステムモジュールが存在する．一方，カーネル内のモジュールとは別に，これらのファイルシステムに操作を行う，様々なソフトウェアツールが存在する．NILFS2 はユニークな特徴を持ったファイルシステムである．しかし，これらのツールへの対応が不十分であった．今回，これらの様々なツールを NILFS2 に対応させた．個々のパッチはメインツリーに取り込まれた．この結果，利用者はツールを最新版にアップデートするだけで，NILFS2 に対する様々な操作を簡単に行えるようになった．

キーワード NILFS2, grub2, util-linux, util-linux-ng, udisks, palimpsest, gnome-disk-utilities, GNU parted

1. はじめに

ファイルシステムはディスク上にファイルを格納するための機能でオペレーティングシステムに不可欠な機能である．

Linux 上でファイルシステムは VFS とブロックデバイスレイヤーの間にソフトウェアモジュールとして作成される(図 1)．このモジュールには ext2, ext3, ext4, jfs, xfs, btrfs など様々な実装がある．

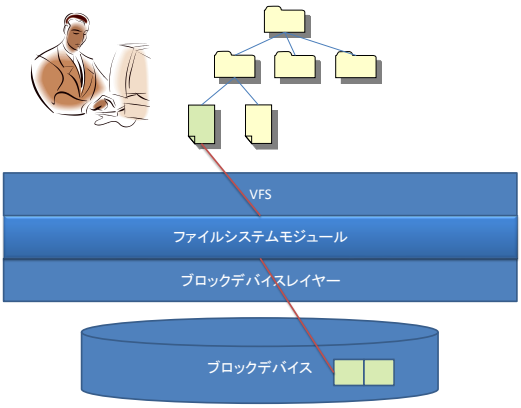


図 1: Linux 上のファイルシステム

ファイルシステムに対して利用者が行う主な操作は，フォーマットとマウントである．例えば，mkfs などのソフトウェアツールでブロックデバイスをフォーマットする際にファイルシステムを指定する．その後，mount コマンドでブロックデバイスを任意のディレクトリにマウントする.マウントすれば，作成，コピー，変更，削除など基本的なファイル操作が可能となる．しかし，実際にファイルシステムを定常的に利用するには，フォーマットやマウント以外のファイルシステムの利便性を高める機能も必要となる．

ファイルシステムの利便性を高める機能として，ブートデバイスとしての使用，パーティション分割，パーティションの拡張/縮退，ブロックデバイス以外の指定によるマウント，などがある．

これらの機能は様々なソフトウェアツールでサポートされている．これらのソフトウェアツールは，Linux ディストリビューションで標準的に利用できる．

NILFS2 (New Implementation of Log File System 2)[1][2][3] は Linux カーネルに搭載されたファイルシステムである．連続的な checkpoint 作成などのユニークな特徴を備えている．しかし，基本的な機能以外のソフトウェアツールへの対応が遅れており，ブートローダなどで利用できない．

そこで，今回 NILFS2 の利便性向上の為にこれらのツールの開発を行う．

以下に本稿の構成を述べる，第2章で NILFS2 の概要を説明する．第3章で利便性向上の機能とそれらを実現しているツールを紹介する．第4章，第6章，第5章，第7章ではそれぞれ，GNU GRUB2[5], GNU Parted[6], blkid, palimpsest[7] での実装方法を紹介する．第8章はまとめである．

2. NILFS2

NILFS2 は NTT サイバースペース研究所で開発されたファイルシステムである．このファイルシステムは Linux カーネル 2.6.30 からバニラカーネルに同梱されている．NILFS2 の特徴は:

- 追記型ファイルシステム
- ファイルシステムの履歴を保存する連続的 checkpoint
- ファイルシステムの状態を保存するスナップショット
- 空き領域作成の為のガベージコレクタ

などがあげられる．

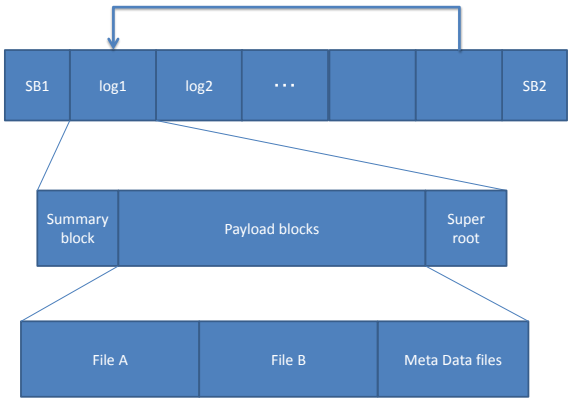
個々の特徴を簡単に紹介する．

2.1. 追記型ファイルシステム

NILFS2 は追記型のファイルシステムである．ファイルの変更が行われても，ディスクブロックの変更は行われず別のブロックへ新たに書き込まれる．

NILFS2 のディスクフォーマット[4]の概要を図 2に示す．

NILFS2 のディスクフォーマットは一つのパーティションを巨大なリングバッファとして扱う．ファイルシステムの変更は”log”単位でディスクに追記されていく．



一つのパーティションをring bufferとして利用

図 2: NILFS2 ディスクフォーマット概要

パーティションの先頭と末尾にスーパーブロック，SB1 と SB2 が存在する．NILFS2 で唯一書き換えが行われるブロックがこの 2 つのスーパーブロックである．

スーパーブロックや log ではチェックサムが使用されている．

2.2. 連続的 checkpoint

NILFS2 ではファイルの変更が行われ，log が追加されると新たな checkpoint が作られる．checkpoint とはファイルシステムのある時点の状態をさす．このため，ファイルの削除や変更など，ファイルシステムの変更履歴がすべてディスクに保存される(図 3).

2.3. スナップショット

2.2章で作られた checkpoint は，スナップショットに変更することでリードオンリーマウントが可能になる．checkpoint は2.4章で述べるガベージコレクタで削除されるまでディスク上に存在している．このため，現在だけでなく，過去の時点のファイルシステムの状態をスナップショットとして保存できる．

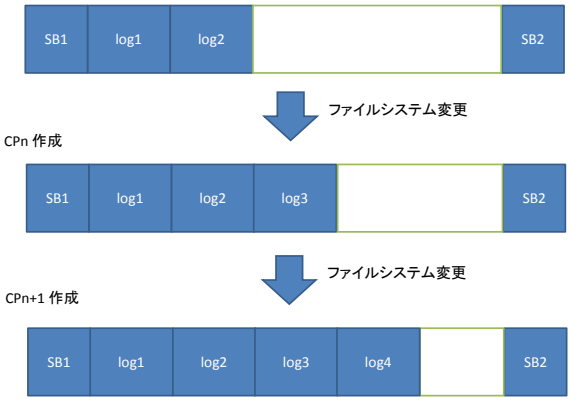


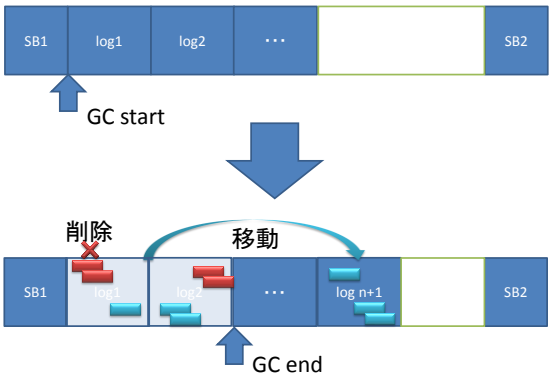
図 3: 連続的 checkpoint

2.4. ガベージコレクタ

NILFS2 は追記型なので，そのままでは追加された log によりディスク空き容量が減少する．古い log を削除しなければ，現在のファイルシステムの全ファイルのサイズの合計とは関係なく空き領域がなくなり，書き込みができなくなってしまう．

この問題を解決するため，NILFS2 にはガベージコレクタデーモンが存在する．ガベージコレクタデーモンは checkpoint を検査し，予め設定した期間より古い checkpoint に含まれているブロックを削除していく．

削除方法はいわゆるコピー型ガベージコレクションの動作である(図 4). このため，NILFS2 では，ファイルが変更されなかったとしても，ファイルの物理的な位置は移動している可能性がある．



活きているブロックを移動させて、空き領域を作る

図 4: NILFS2 ガベージコレクタ概要

2.5. NILFS2 の現状

このように NILFS2 は他のファイルシステムにはないユニークな特徴を備えている．また NILFS2 は他のファイルシステムと同等の安定性を備えている．しか

し NILFS2 は，ext3 等の代表的なファイルシステムのように利便性を高めるソフトウェアツールの対応が不十分であった

3. 利便性を上げる機能とツール

この章では，ブロックデバイスを直接扱う機能や主なツールの紹介を行う．

3.1. ブロックデバイスを直接扱う機能

現代のオペレーティングシステムではブートローダから GUI まで様々なソフトウェアが直接的に/間接的に，カーネルのファイルシステムモジュールを利用せず，直接ブロックデバイスへアクセスし，ファイルシステムの情報や，ファイルの操作を行う(図 5)．以下にこれらの機能を述べる．

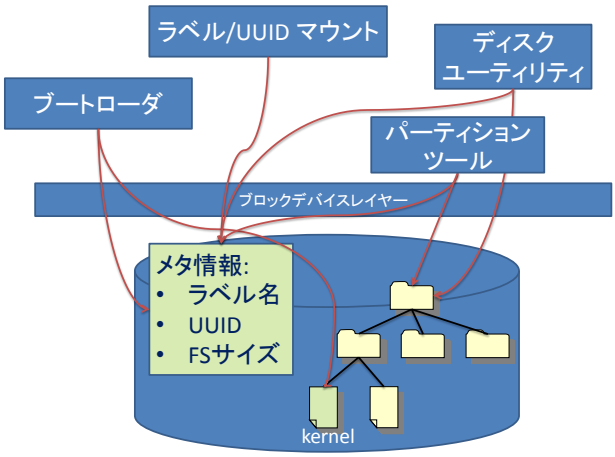


図 5: ブロックデバイスに直接アクセス

ブートローダ

ブート直後に動作し，カーネルをロードする．カーネルの指定はカーネルが格納されている物理的なブロックを埋め込んでいる場合もあるが，この場合新しいカーネルを起動するためにはブートローダをインストールし直さなければならない．また，間違って同じカーネルを上書きしてしまった場合，ブロックアドレスが変わってしまいカーネルのロードに失敗する．

最近のブートローダはファイルシステムを読み込み，パスを指定することでカーネルをロードすることが可能である．このためファイルシステムを読み込む機能が必須となる．

Linux で主に利用されるブートローダとしては GNU GRUB および GRUB2 がある．

ラベル/UUID マウント

通常ブロックデバイスをマウントするには対応するデバイスファイルを指定する．しかし，デバイスファイルはデバイスの追加順序などのより異なるファイル名になる．

ラベルや UUID を指定したマウントは，ファイルシステムの属性として付加されている UUID やラベルを指定してマウントする機能である．このため，デバイスファイル名が異なる環境でも正しくマウント出来る．

ラベルや UUID を指定したマウントにはファイルシステムのラベルや UUID を読み込む機能が必須となる．

Linux では blkid がラベルや UUID の読み込みに利用される

パーティショニング

パーティショニングツールはディスクのパーティション分割や，分割したパーティションのリサイズなどを行うつールである．

ディスクをパーティションに分割するだけでなく，分割したパーティションにファイルシステムを構築や，既にファイルシステムが存在するパーティションのリサイズ時にファイルシステムのリサイズを同時に行なう機能などがある．

Linux の実装では fdisk や GNU Parted などが主に利用されている．

ディスクユーティリティ

ディスクユーティリティはディスクのパーティションに分割，削除，パーティションのフォーマット，パーティションラベルの変更，パーティションサイズの変更など，あらゆるディスク周りの作業を行うグラフィカルなツールである．

Linux では gnome-disk-utilities (palimpsest)が主に利用されている．

4. GNU GRUB2

本章ではブートローダである GNU GRUB2(以下 GRUB2)の概要と開発の考慮点，パッチの作成について述べる．

4.1. GRUB2 概要

GRUB2 は高機能なブートローダである．Linux のディストリビューションの多くで採用されている．GRUB2 はブロックデバイスのブロック番号ではなく，直接ファイルシステムの中に格納されているカーネルを読み出すことができる．ブートローダはオペレーティングシステムをブートさせるソフトウェアである．そのため，ブートローダはオペレーティングシステムの機能であるファイルシステムモジュールを利用でき

ない．このためブートローダ自身がファイルシステムを読み，パスからファイルのデータを読み出す機能が必要となる．

4.2. GRUB2 用ファイルシステムモジュールの開発

GRUB2 ではファイルシステムを読み込むために，カーネルと同様にファイルシステム毎にソフトウェアモジュールを備えている．新たなファイルシステムに GRUB2 を対応させるためには，そのファイルシステム用のソフトウェアモジュールが必要になる．

GRUB2 のファイルシステムや raid などのモジュールは，機能毎にダイナミックに追加可能である．これらのモジュールを読み込むために grub-core が存在する．grub-core は初期のブートから様々なモジュールを読み込むコアの機能を提供している．

2.4章で述べたように，NILFS2 ではガベージコレクタにより，活きているブロックは常に移動している．このため，ブートローダの core 機能が NILFS2 モジュールをブロックアドレスで指定して読み込むことは出来ない．また，PC の MBR は 512B しか領域がないので MBR にファイルシステムモジュールをインストールすることはできない．このため，GRUB2 で NILFS2 をサポートするには他のファイルシステムとは異なる方法が必要と考えられた．

一方，PC のディスクは典型的に MBR の直後に DOS 互換の為の領域がある[8]．GRUB2 ではこの領域にファイルシステム読み込むまでに必要なモジュールを保存していた(図 6)．このため NILFS2 のようにブロックが絶えず移動しているファイルシステムでもサポートすることは可能であった．

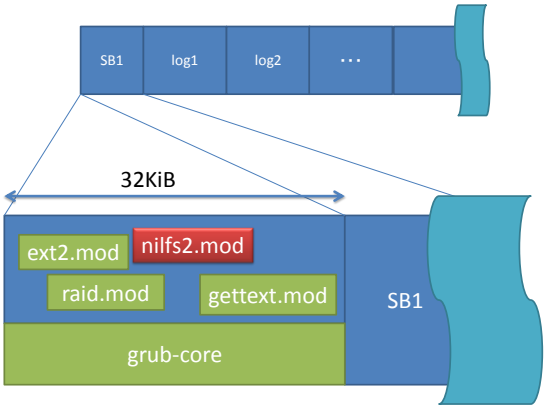


図 6: grub2 使用時のディスクの状態

4.3. GRUB2 用 NILFS2 パッチ作成

GRUB2 のファイルシステムモジュールを実装するには，ファイルを読み込む為に必要となる操作のハンドラを実装する．実装したハンドラの一覧を図 7に示す．よくあるファイルシステムのハンドラと似ているが，mount は行われていないので，ファイルシステムの状態は保存されない．そのため,open や read のハンドラが行われるたびにスーパーブロックから最新ログを探す操作が必要となる．

```
static struct grub_fs grub_nilfs2_fs = {
  .name = "nilfs2",
  .dir = grub_nilfs2_dir,
  .open = grub_nilfs2_open,
  .read = grub_nilfs2_read,
  .close = grub_nilfs2_close,
  .label = grub_nilfs2_label,
  .uuid = grub_nilfs2_uuid,
  .mtime = grub_nilfs2_mtime,
#ifdef GRUB_UTIL
  .reserved_first_sector = 1,
#endif
  .next = 0
};
```

図 7: grub2 ファイルシステムハンドラ一覧

GRUB2 は GPL3 によって配布されており，GPL2 で配布されている NILFS2 のモジュールのソースコードは流用できない．そのため NILFS2 のファイルを読み出すモジュールを新たに GRUB2 用に開発した．

開発は C 言語．作成したモジュールの行数は約 1000 行であった．

コーディングスタイルは GNU コーディングスタイルを利用しており，パッチを投稿するさいには indent プログラムなどでインデントを調整することが望まれた．

開発したモジュールは GRUB2 のメインラインに既にマージされている．また，debian/sid ではパッケージとして利用可能である．

また，GRUB2 は Free Software Foundation[9] が開発しているため，約 10 行以上のパッチの場合，別途書面にてソフトウェアの譲渡契約を結ぶ必要があった．この事務作業には 2～4 週間程度の時間がかかった．

5. blkid

本章では blkid の概要と開発の考慮点，パッチの作成について述べる．

5.1. blkid 概要

blkid は util-linux-ng[10]に同梱されているブロックデバイスの属性を表示するプログラムである．blkid を使うことでが，次の三つの情報を得ることができる．

即ち:

1. どのファイルシステムであるか
2. ファイルシステムをのラベル
3. ファイルシステムの UUID

である．また，取得された情報を/etc/blkid.tab に保存する．ディスクへのアクセス権限がないユーザが blkid を実行した場合このファイルの内容を表示する．

実際に mount システムコールを呼び出す場合はデバイスファイルの指定が必要になる．その為，ラベルや UUID を指定してマウントを行う場合，ラベルや UUID とデバイスファイルをマッピングする必要がある．

このマッピングを行うのが blkid である．blkid は util-linux-ng に同梱されているプログラムでブロックデバイスからスーパーブロックを読み出し，そのブロックデバイスのファイルシステムや UUID などの情報をする．

mount などのプログラムがそのファイルを読み出し，ブロックデバイスに変換してマウント操作を行う．

5.2. blkid のファイルシステムモジュールの開発

図 8に blikd のアーキテクチャを示す．

blkid は libblkid を利用している．実際にブロックデバイスから属性を取り出す機能は libblkid に備わっている．

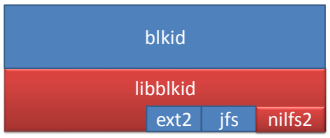


図 8: blkid architecture

libblkid はブロックデバイスからファイルシステムの属性などの情報を取り出すために，様々なファイルシステムのスーパーブロックを読み出せる．ファイルシステムを認識し読み出すためには，ファイルシステム毎にスーパーブロックの情報を読み出す機能が必要になる．この機能を追加するためにはスーパーブロックの情報と，スーパーブロックの validation 機能を持ったハンドラを作成する

5.3. blkid パッチ作成

図 9に実際に作成した登録したハンドラを示す．

blkid はブロックデバイスが登録された全てのハンドラで認識できない場合，ブロックデバイスを無視する．このため，サポートされていないファイルシステムが存在した場合 blkid を起動しても認識も表示もされない．

図 10 に今回作成した NILFS2 用のハンドラを組み込んだ blkid の実行画面を示す. sda2 が NILFS2 として認識され, UUID とラベル名が正しく表示されている．

```
const struct blkid_idinfo nilfs2_idinfo =
{
        .name           = "nilfs2",
        .usage          = BLKID_USAGE_FILESYSTEM,
        .probefunc      = probe_nilfs2,
        .magics         =
        {
                {
                        .magic = NILFS_SB_MAGIC,
                        .len = 2,
                        .kboff = NILFS_SB_KBOFF,
                        .sboff = NILFS_MAG_OFF
                },
                { NULL }
        }
};
```

図 9: blikd のハンドラ

```
$ blkid
/dev/sda1: LABEL="hoge" UUID="c2bed900-02cc-45f7-a8ab-36526a3f58b4" TYPE="ext4"
/dev/sda5: UUID="c2c7c8e6-8868-448f-b788-d7a9fffe5801" TYPE="swap"
/dev/sdb1: LABEL="xfs" UUID="0dee0fe3-daf7-47d8-a243-8fb0ac3beffd" TYPE="xfs"
/dev/sdb2: LABEL="NILFS2 is great!" UUID="b6c8260f-3f58-44ee-b417-133d5f63093f"
TYPE="nilfs2"
/dev/sdb3: UUID="4c96f277-e04f-42ca-aa46-ff7fc7b6566d" TYPE="ocfs2"
$
```

図 10: blkid 実行画面

開発には C 言語．作成したモジュールは約 100 行だった．

開発したパッチは既にメインラインにマージされている．

6. GNU Parted

本章では GNU Parted(以下 Parted)の概要と開発の考慮点，パッチの作成について述べる．

6.1. Parted の概要

Parted はディスクのパーティショニング，パーティションの拡張/縮退などを行うツールである．パーティショニングがメインだが，既にファイルシステムが存在するディスクを対象にした場合には，ファイルシステムの種類を判別し表示を行う．

また，パーティションの作成と同時にファイルシステムの作成，パーティションの拡張/縮退と同時にフ

ファイルシステムの拡張縮退などの機能も持っている．

6.2. Parted 用ファイルシステムモジュールの開発

図 11に簡単な parted のアーキテクチャを示す．

parted は主に二つのコンポーネントからなっている．すなわち，parted，及び libparted である．parted はキャラクターベースのインタフェースを持つフロントエンドで実際の機能はダイナミックライブラリである libparted を利用している．libparted はスタティックライブラリである libfs を利用している．

libfs はファイルシステム毎に独立したパーツでできており，各種の機能(probe, open, create, close, check, copy 等)を実現するハンドラを作成することで特定のファイルシステムの機能を parted に提供することができる．

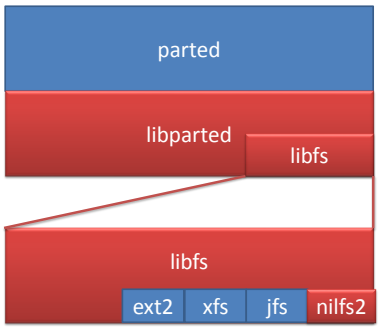


図 11: parted architecture

6.3. Parted 用 NILFS2 パッチ

Parted のファイルシステムモジュールのハンドラは多岐に渡る．このため，

すべての機能を実装する必要はなく，最低限 probe 機能を実装すればよい．ハンドラの一覧を図 12に示す．

```
static PedFileSystemOps nilfs2_ops = {
        probe:                  nilfs2_probe,
#ifndef DISCOVER_ONLY
        clobber:                nilfs2_clobber,
#else
        clobber:                NULL,
#endif
        open:                   NULL,
        create:                 NULL,
        close:                  NULL,
        check:                  NULL,
        copy:                   NULL,
        resize:                 NULL,
        get_create_constraint:  NULL,
        get_resize_constraint:  NULL,
        get_copy_constraint:    NULL
};
```

図 12: parted ハンドラ一覧

今回作成した NILFS2 用のモジュールは probe とファイルシステムの破棄機能のみを実装した．probe はスーパーブロックを読み込み，CRC のチェックを行っている．NILFS2 には2.1章で述べたように，二つのスーパーブロックが存在する．作成したモジュールは SB1 か SB2 のどちらかが正しいスーパーブロックであれば NILFS2 パーティションとして認識する．ファイルシステムの破棄は二つのスーパーブロックを破壊する．

実際に動作している様子を図 13に示す．1 番目のパーティションにある NILFS2 パーティションが認識されているのが確認できる．

```
$ sudo ./parted/parted /dev/hdb
GNU Parted 2.3.10-5896
Using /dev/hdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: QEMU HARDDISK (ide)
Disk /dev/hdb: 8590MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      32.3kB  165MB   164MB   primary  nilfs2
 2      165MB   411MB   247MB   primary  ext3

(parted)
```

図 13: 拡張した Parted の実行画面

開発は C 言語．作成した NILFS2 モジュールの行数は約 200 行だった．

Parted は GRUB2 同様に，Free Software Foundation が開発しているソフトウェアである．このため，GRUB2 とは事なるソフトウェアの譲渡契約が必要になる．現在，契約は FSF の処理待ちになっている．

7. Palimpsest

本章では Palimpsest の概要と開発の考慮点，パッチの作成について述べる．

7.1. Palimpsest 概要

Palimpsest は gnome-disk-utility プロジェクトのディスクとメディア管理の為のツールである．ファイルシステムへの操作としては，ファイルシステムの作成，マウント，ラベルのリネーム，ファイルシステムチェック，拡大，縮小などがある．

Palimpsest は root ユーザではないユーザが利用することを前提にしており，root 権限が必要なディスクへのアクセスは D-Bus[11]を介して行われる．palimpsest 自体にも個々のファイルシステム毎の機能は存在するが，ディスクへの操作は D-Bus を介したサーバサイドのサービスが行っている．

7.2. Palimpsest 用ファイルシステムモジュール

図 14に palimpsest と palimpsest が利用する D-Bus サーバサイドのサービスのアーキテクチャを示す.

サーバサイドのサービスは三つのコンポーネントから成り立っている. 即ち, udisks[12], libparted, libblkid である. udisks がディスクを直接アクセスする D-Bus のサービスプロバイダとして動作している.

liblkkid はファイルシステムの認識に, libparted はパーティションの作成/削除などに利用されている. libblkid および libparted は既に5章と6章で述べた通りである.

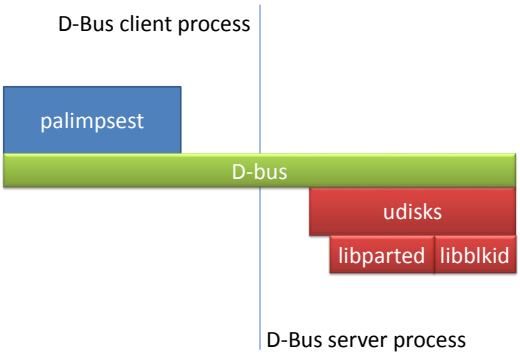


図 14: palimpsest architecture

udisks は D-Bus のクライアントに対してファイルシステムの情報とファイルシステムへの操作を提供する.

ファイルシステムの情報としては, id(blkid が表示するファイルシステムの名前), name, ラベルの最大バイト数などである.

palimpsest 自身にも個々のファイルシステムを個別に取り扱う部分も存在するが, ブロックデバイスに対する全ての操作は udisks が行っている. このため, palimpsest で利用出来るファイルシステムを増やすためには udisks を新たなファイルシステムに対応させる必要がある.

7.3. udisks 用 NILFS2 パッチ

udisks で新たにファイルシステムを対応させる為には, そのファイルシステムで可能な操作を capability として登録し, 操作毎に作られたファイルに個別のファイルシステムの操作方法を記述する必要がある.

図 15に capability テーブルを示す. また, **図 16**にファイルシステムのラベルを変更するための操作である, change-filesystem-label の断片を示す.

```
{
      "nilfs2", /* id */
      "NILFS2", /* name */
      TRUE, /* supports_unix_owners */
      TRUE, /* can_mount */
      TRUE, /* can_create */
      80, /* max_label_len */
      TRUE, /* supports_label_rename */
      FALSE, /* supports_online_label_rename*/
      FALSE, /* supports_fsck */
      FALSE, /* supports_online_fsck */
      FALSE, /* supports_resize_enlarge */
      FALSE, /* supports_online_resize_enlarge */
      FALSE, /* supports_resize_shrink */
      FALSE, /* supports_online_resize_shrink */
},
```

図 15: udisks 用 capability テーブル

```
    …
  }
else if (strcmp (fstype, "nilfs2") == 0)
  {
    if (!validate_and_escape_label (&new_label, 80))
      goto out;
    command_line =
      g_strdup_printf ("nilfs-tune -L ¥"%s¥" %s",
                       new_label, device);
  }
else
  {
    …
```

図 16: helpers/job-change-filesystem-label.c

今回, ファイルシステムの操作として, ファイルシステムの作成とファイルシステムラベルのリネーム機能を実装した.

図 17に実行画面を、図 18に一部拡大画面を示す.

図 17では NILFS2 パーティションの認識と, ラベルの変更ボタンが表示されていることが確認できる.

開発は C 言語. パッチのサイズは, テストへのパッチも含め約 400 行であった.

7.2章で述べたように, udisks は libparted や libblkid の実装に依存している. 今回実装を行った部分に関しては libblkid によるファイルシステムの認識が必要になった.

udisks へのパッチを送付した時点では, blkid 用の NILFS2 モジュールはパッチとして投稿されていたが, メインラインには取り込まれていなかった. このため, udisks の開発コミュニティにパッチを投稿しても, 反応は得られなかった.

util-linux-ng へパッチが取り込まれた後に, blkid による認識が可能になったことを伝えるメールを投稿したところ, パッチへの対応を行ってくれた.

現在, udisks 用のパッチはメインラインに取り込ま

れている．

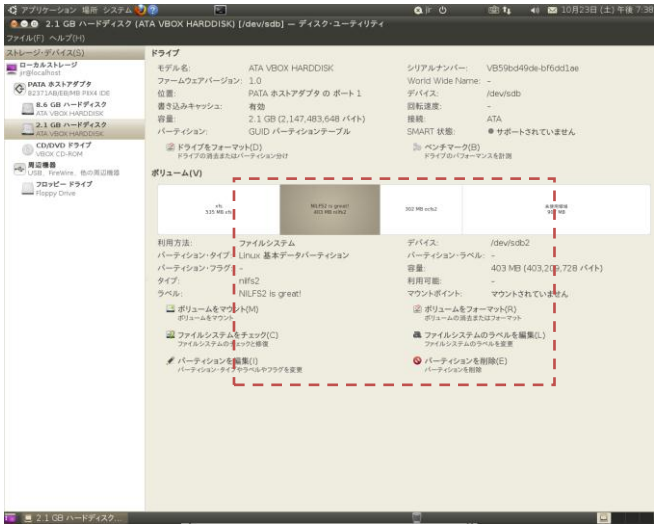


図 17: palimpsest 実行画面

図 18: 図 17 拡大画面

8. まとめと今後の課題

ファイルシステムの利便性を高める様々な機能を満たすツールが Linux に備わっていることを述べた．NILFS2 はユニークな特徴を持ったファイルシステムであるが，これらツールは対応していなかった．今回，これらのツールを NILFS2 に対応させるためのパッチを作成した.作成したパッチはほぼ全て，ツールのメインツリーに取り込まれた．これにより，利用中の Linux シससテム上でツールをアップデートするだけで，利用者が NILFS2 に対する操作を簡単に行えるようになった．具体的には，①NILFS2 のブートファイルシステムとしての利用，②NILFS2 パーティションのラベルや UUID 指定でのマウント，③Parted でのファイルの認識，破棄，④Palimpsest を使った操作，である．

今後の課題は，NILFS2 のユニークな機能を生かした操作を利用者が行えるようにすることである．具体的には，grub2 で checkpoint を指定したファイルシステムへのアクセス，unclean なマウント状態から最新ログへのロールフォワード．Parted によるパーティションの拡張などが上げられる．

9. 謝辞

本稿で述べたツールの開発にあたり NILFS2 の実装の詳細についてアドバイスして頂いた NTT サイバースペース研究所の小西隆介氏に感謝いたします．

忙しい中執筆に協力してくれた株式会社ウニークス社員に感謝いたします．

文　献

[1] NILFS2. http://www.nilfs.org/

[2] 佐藤ほか，ログ構造化ファイルシステム NILFS の設計と実装，情報処理学会 論文誌コンピューティングシステム(ACS)，Vol.2，No.1，pp.110-122，2009.

[3] Ryusuke Konishi, The NILFS2 Filesystem: Review and Challenges, Japan Linux Symposium 2009 Presentation material, Oct 2009. http://www.nilfs.org/papers/jls2009-nilfs.pdf

[4] Linux-2.6/Docmuntation/filesystems/NILFS2.txt http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=blob;f=Documentation/filesystems/NILFS2.txt;h=d5c0cef38a7122ed378371acba045f928f891c0b;hb=HEAD

[5] GNU GRUB http://www.gnu.org/software/grub/

[6] GNU Parted http://www.gnu.org/software/parted/

[7] Palimpsest http://library.gnome.org/users/palimpsest/

[8] The PC hard disk layout http://www.pixelbeat.org/docs/disk/

[9] Free Software foundation http://www.fsf.org/

[10] util-linux-ng http://userweb.kernel.org/~kzak/util-linux-ng/

[11] D-Bus http://www.freedesktop.org/wiki/Software/dbus

[12] udisks http://www.freedesktop.org/wiki/Software/udisks