Vol.2016-IOT-35 No.9
Vol.2016-SPT-20 No.9
2016/9/24

IOTへの応用に向けたRaspbianの2次記憶長寿命化

重光 史也[1,a)]　鈴木 貢[1,b)]

概要：Raspberry Piは，2次記憶やグラフィックディスプレイアダプタ，そしてUSBホスト機能を名刺大のサイズに集積した汎用コンピュータである．Raspberry Piはさらに GPIO（汎用入出力）ピンや I^2C やSPI等のシリアル通信機能を備えており，研究や教育において独自のIoT （Internet of Things）デバイスを構築するのに使われつつある．Raspberry Piを使って実用的なIoTデバイスを構築する上で障壁となるのは，2次記憶として使われているSDカードの寿命である．本研究では，SDカードへの書き込みを大きく減らして寿命を延ばすオペレーティングシステムの構築を探求する．

Shigemitsu Fumiya[1,a)]　Suzuki Mitsugu[1,b)]

Abstract: Raspberry Pis are general purpose computer consisting of second-level storage, graphic display adaptor and USB host function with visiting card size. They also have GPIO (general purpose input output) pins and some serial communication functions such as I^2C, SPI, and researchers and educators are going to use them to construct original IoT (Internet of Things) devices. A barrier to constructing a practical IoT device with the Raspberry Pihas been the life time of the SD cards which is utilized in them as the second storage. In this study, we tried to construct an operating system that greatly reduces write operation into the SD cards to get longer lifetime of them.

1. はじめに

Raspberry Piは**図 1** に示すように，2次記憶やディスプレイ表示機能，USBホスト機能を有する汎用計算機を名刺大のボードの集約したものであり，10年前のパーソナルコンピュータや20年前のワークステーションを遥かに超えた機能や性能を有している．Raspberry Piではさらに40ピン（旧型は26ピン）のGPIO（汎用入出力）端子を備え，ビット操作指向の入出力端子として用いたり，I^2C やSPI，UARTといった各種シリアル通信，内臓タイマーのトリガー入力やパルス幅出力としても用いることができる．また，Raspbianと呼ばれるDebian Linuxベースのオペレーティングシステムが用意されており，このメモリ保護を備えた本格的なオペレーティングシステムにより，セルフ開発やテストも可能である．

このために，Raspberry Piは教育から研究に至る種々の用途に幅広く使われるようになってきた．そして，IoT（Internet of Things）を指向した種々の製作記事が，日経Linux等の雑誌に掲載されたり，単行本として出版されるようになってきた．Raspberry Piは性能によって異なるが非常に安価（5〜30ドル）であるので，実用的なIoTデバイスの中核として用いることができるように思えるが，このとき問題となるのは，2次記憶として用いられているSDカードの寿命である．

SDカードは元々ディジタルカメラ等のストレージ向けのものであるので，汎用オペレーティングシステムの2次記憶としての使用に耐え得るファームウエアでのウエアレベリング [1] が備わっていないことが多く，Raspbianでも特にソフトウェア的なウエアレベリングを行っていない．そして，実際にオペレーティングシステムが稼動しているうちに2次記憶が動作不能になることが知られている．

そこで本研究では，IoT向け汎用OSディストリビューションの1つであるVoyage Linuxの管理方式を参考にしながら，Raspbianの2次記憶の長寿命化を探求する．

[1] 島根大学
Shimane University, Matsue, Shimane 690–8504, Japan
[a)] s133044@matsu.shimane-u.ac.jp
[b)] suzuki@cis.shimane-u.ac.jp

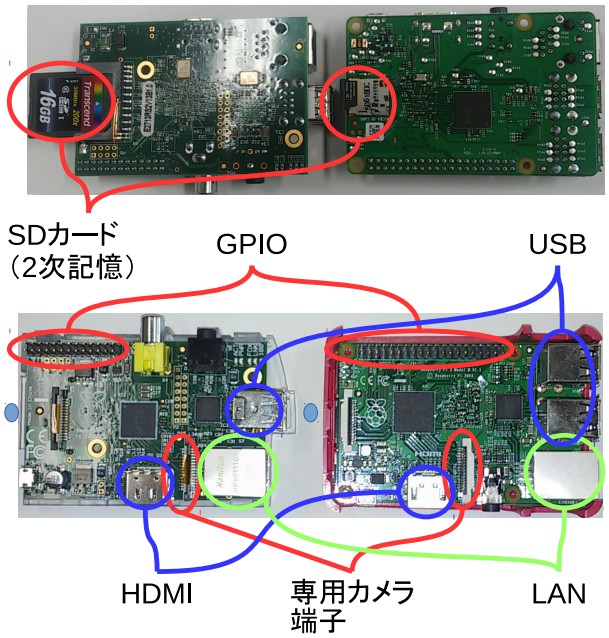

図 1　Raspberry Pi（左：1 Model A+，右：2 Model B）
Fig. 1　Raspberry Pi (left:1 Model A+，right:2 Model B)

2.　関連事項

この節では，本研究に関連する事項をまとめて説明する．

2.1　Voyage Linux

本研究は Voyage Linux[2] の設計を大いに参考にしている．このオペレーティングシステムは，x86 と x64 アーキテクチャ向けの，256MB の 2 次記憶で稼働する軽量 Linux のディストリビューションの 1 つで，パッケージングシステムとして Debian を用いている．IoT というキーワードが一般化する以前から開発が開始され，現在もバージョンアップや改良が行われ，MPD（Music Player Daemon）等の特化サブディストリビューションも用意されている．

このディストリビューションの特徴の 1 つは，デフォールトの設定では 2 次記憶をリードオンリーとしてマウントすることである．突然の電源断でもファイルシステムが壊れず，その後の再起動でも fsck は不要である．/var/log 等の書き換えがあるパーティションには，tmpfs[3] と呼ばれる一時的なファイル蓄積機能が裏に張り付くようになっていて，ファイルシステム上で書き込みの結果が反映する．

その他/usr 等のパーティションはリードオンリーであり，権限に依らず書き込みできない．`apt-get update` 等のシステムの変更時には，先に `remountrw` というスクリプトを起動してリードライトモードでリマウントする．

正しくシャットダウンを行った場合は，/var/log 等への更新を 2 次記憶に反映させるのに `voyage-sync` というコマンドスクリプトが起動され，変更を反映する．

このように柔軟性が高い Voyage Linux であるが，ARM アーキテクチャへの移植版が用意されていない．そこで，本研究では Raspberry Pi でこれと同じ構成を実現することを目標としている．

2.2　Raspbian

Raspbian は Raspberry Pi の標準オペレーティングシステムの 1 つで，この機種で最も一般的に使われるものであると考えられる．ウインドウシステムや各種開発ツールやユーティリティを備えた完備したワークステーション用オペレーティングシステムの体裁になっている．

現在の Raspbian は 4GB 以上のメディアを必要としており，このディストリビューションをそのまま用いて IoT デバイスを構成するのは，無駄が多く，また 1 節で述べたように 2 次記憶を（書き込みで）酷使するという問題点がある．また，実際に 2 次記憶として使用する SD カードとの相性問題があり，公開されている一覧表 [4] を信じて SD カードを購入してもうまく稼働できない場合 [5] さえある．

2.3　その他の OS やディストリビューション

OSMC や WEATHER STATION，PINET 等の特定目的向けディストリビューションは，新規 IoT デバイスの開発向けではないと判断されるので，Voyage Linux のようなファイルシステムのマウンティング戦略を実現しているかもしれないが，本研究の参考にはならないと考える．

2.4　AUFS マウントによる書き込み回避

ファイルシステムをユニオンマウント [6] の改良版である AUFS[7] と fsprotect を用いてマウントする手法 [8] で，SD カードへの書き込みを回避することが可能である．

本来のユニオンマウントは，Knoppix のような Live CD の実行環境で，リードオンリーのファイルシステムに対する書き込みは別の USB メモリ等の書き込み可能な媒体に吸収させ，再起動時にそれを再現してシャットダウン時の環境を再現させるようなシステムを構成するために使う．

この手法を用いると，Web ページ [9] に記載されているような方法と比べて，`/var` のリンク貼り直し等のシステムの再構成は必要としないが，再起動時には以前の変更をすべて忘れてしまうので，本研究が目標とする「シャットダウン時に変更を記録する」という目標は達成できない．

2.5　Raspberry Pi のブートのしくみ

Raspberry Pi の ARM アーキテクチャのプロセッサを含む SoC（system on a chip，BCM2835[10]，BCM2836[11]，BCM2837）のオペレーティングシステムブートのメカニズムは，通常の計算機のものとは異なり，さらに Raspbian でもバージョンによっても手順が変遷している．そこでこの節では，本稿を作成している時点の Raspbian の手順を説明する．以下の手順は，主に Web ページ [12] を参考に

し，実機の中身を解析した結果である．Raspberry Pi はパーソナルコンピュータで BIOS 等と呼ばれるブートローダの ROM を搭載していない．また，SoC のリセット直後は，主プロセッサである（ARM1176JZF-S 等）はリセットされたままである．

(1) GPU の VideoCore IV[13] が，SoC の中に作り付けてある ROM のコードを実行し最初のブートを行う．
Raspberry Pi に搭載されているものは，SD カードの FAT32 か FAT16 のパーティション（第 1 パーティションで，最終的に`/boot` にマウントされる）から，約 16KB のファイル `bootcode.bin`（GPU のコード）を GPU の L2 キャッシュにロードする．

(2) GPU は (1) でロードしたバイナリを実行する．
Raspbian では，主記憶の DRAM を活性化し，そこに同じパーティションから約 2.5MB のファイル `start.elf`（GPU のコード）をロードする．

(3) GPU は (2) でロードしたバイナリを実行する．
Raspbian では，同じパーティションからディスプレイの設定や Linux カーネル起動のパラメタ（ロード番地やコンソールの設定，`/dev` の設定等）を記述した `config.txt` を読み込む．さらに，`cmdline.txt` と約 4MB の `kernel.img`（ARM の zImage 形式）を読み込み，後者を起動する際に前者を渡す．デフォールトでは，第 2 パーティション`/dev/mmcblk0p2` を`/`に，第 1 パーティション`/dev/mmcblk0p1` を`/boot` にマウントするようになっている．

これより，本稿の目的を達成するためには，以下を行えばよいことがわかる．

(1) Raspbian のカーネルをそのまま利用できる場合は，第 1 パーティションはそのまま利用し，第 2 パーティションを再構成すればよい．

(2) カーネルの再構成が必要である場合は，所定の方法で zImage 形式のカーネルを用意して第 1 パーティションに `kernel.img` としてセットし，第 2 パーティションも目的に合わせて再構成する．

Voyage Linux のカーネルと Raspbian のカーネルの内容を比較した結果，Raspbian のカーネルも Voyage Linux のファイルシステム管理方式で必要な tmpfs 等の機能を備えていることが判ったので，(1) の方法を用いることにした．

3. 設計と実装

この節では，実装に用いるツールやインフラストラクチャの説明を交えながら，実装内容を説明する．

3.1 Linux From Scratch

boot を除く Linux ディレクトリが配置される第 2 パーティションは，Linux From Scratch[14](以下 LFS と略) を参考にして再構成する．これは，Linux システムをソースコードからコンパイルしてスクラッチから作り出すことを目的としているプロジェクトであり, カスタムメイドの Linux システムを構築する雛形となる手順が提供されている．LFS では, コンパクトな Linux システムを作成することが目的の一つに挙げられており，今回の IoT 向け Linux システムの構築に適した方法であると判断した．

LFS を用いてシステムを構成する手順を説明する．以下は LFS 公式サイトが提供する Version 7.9 向けの LFS ブック [15] の内容を簡潔にまとめたものである．

(1) LFS を構築するデバイスに LFS 用のパーティションを作成
ルートディレクトリ以下をマウントするためのパーティションと，必要に応じてスワップパーティション等も用意する．LFS ではこの他にも，ディスクレイアウトを取り決める際の指針も示しており，`/boot` や `/home` といったディレクトリのパーティションを別途設けることを推奨している．

(2) ファイルシステムの作成
作成した空のパーティションにファイルシステムを作る．LFS ではルートファイルシステムとして ext4[16] を用いることを前提としている．ext4 はクラッシュ対策として fsync をある程度の頻度で行うことを前提とするが，これは SD カードの寿命を縮めることになる．

(3) LFS システムを構築するための一時システムをビルド
LFS ブックに記載されているパッケージを，専用のディレクトリ配下（LFS では`/sources`）に取得し，別の専用のディレクトリ配下（`/tools`）にビルドやインストールを行っていく．ここで`/tools` 以下にインストールされたプログラムが一時的なシステムとなる．これは最終的なシステムを構築するためのものではない．

(4) ミニ Linux システムへの移行
前項で LFS システムの構築作業を進めるためのビルド環境が整う．仮想的なカーネルファイルシステムを準備してマウントし，`chroot` によって一時的なミニ Linux システムへ移行し，LFS システムにおけるディレクトリ構造を生成し，基本的なファイルとリンクの生成を行った後に，ソフトウェアをインストールしていく．

(5) 各種設定
ブートスクリプトのインストールと，全般的なネットワーク設定等のシステムの設定を行う．ブートスクリプトは LFS 用に予め用意されているものがある．

(6) マウントポイントの指定とカーネルの構築
LFS システムをブート可能にするために，`/etc/fstab` の作成やカーネルの構築を行う．システムのブートのために，LFS では GRUB を用いたブートプロセスの設定作業を仮定しているが利用しているブートロー

ダーが別にあればその限りではない.

3.2 設計方針

Voyage Linux ライクなシステムを開発するための設計方針を記す．LFS ブックの流れに従いながら適宜変更を加えていくことになる．Voyage Linux では，基本的にルートファイルシステムは ext2 で作成し，リードオンリーでマウントする．その理由は，ext3 が提供するジャーナリングや，さらに ext4 が提供するジャーナリング関連の信頼性向上やサブディレクトリ個数の制限撤廃等は，大半の IoT デバイスでは不要であるということであると推測する．以上から，LFS システムのファイルシステムは ext2 で作成する．また Voyage Linux のパーティションの区切り方に倣って，その他のパーティションは作成しない．

LFS システムの構築段階では，リードーオンリーでルートファイルシステムをマウントする．Voyage Linux のブートスクリプトを参考にして，LFS システムのブートスクリプトを用意する．ここで `voyage-sync` を実装することになる．また，システムを変更する際のために，`remountrw` や `remountro` に相当するスクリプトを作成する．

4. まとめと今後の予定

本報告では，Raspberry Pi で稼働するオペレーティングシステムの 2 次記憶媒体の長寿命化を目指して，2 次記憶媒体への書き込みをなるべく少なくする 1 つの方策として，Voyage Linux のファイルシステムの制御戦略を適用することを提案した．

本稿で提案した手法を用いる場合の問題点は，シャットダウン時に `voyage-sync` を行っている最中の電源断で，ファイルシステムの破壊を招く可能性が高いと考えられる．IoT デバイスという前提を置くと，基盤に `voyage-sync` を実施している間だけはシステム内部の電源が維持されるように電気 2 重層キャパシタを増設するといった，ハードウエア仕掛けの対策が考えられる．

また現状ではソースの取得からビルドや `/etc` 等のファイルの設定等に至る作業を個別に行う必要があるので，IoT デバイスの開発の手間を削減するためには，Raspbian の Debian パッケージをそのまま利用できるようにすることが望ましい．そのためには，LFS で Raspbian と同じ環境を実現することも考えられるが，そのためには結構大きな労力を要すると想像する．

将来的には作成したディストリビューションを搭載した Raspberry Pi で IoT デバイスを運用してみるといったことも考えている．

発表会場での適切なアドバイスを期待しながら，本稿を締めくくる．

参考文献

[1] 竹内 健：HDD 完全代替に向けて OS による対応が急務，日経テクノロジー online, `http://techon.nikkeibp.co.jp/article/FEATURE/20090219/165972/?ST=print&d=1472476213871` (2009.3).
[2] Voyage Design and Consultants : Voyage Linux | { x86 Embedded Linux = Green computing }, `http://linux.voyage.hk/` (2016.8.30 取得).
[3] Wikimedia Foundation : tmpfs, `https://en.wikipedia.org/wiki/Tmpfs` (2016.8.30 取得).
[4] Embedded Linux Wiki : RPi SD cards, `http://elinux.org/RPi_SD_cards` (2016.8.30 取得).
[5] 1ft-seabass.jp.MEMO : Raspberry Pi で SD カード相性が原因で起動エラーを繰り返す時の対処法, `http://www.1ft-seabass.jp/memo/2015/03/31/raspberry-pi-sdcard-error/` (2016.8.30 取得).
[6] Wikimedia Foundation : Union mount, `https://en.wikipedia.org/wiki/Union_mount` (2016.8.30 取得).
[7] Wikimedia Foundation : aufs, `https://ja.wikipedia.org/ wiki/Aufs` (2016.8.30 取得).
[8] Increments Inc. : RaspberryPi2 の SD カードを aufs と fsprotect で保護する，`http://qiita.com/zakkied/items/c22faa3f22b4167e7024` (2016.8.30 取得).
[9] Charles's Blog : Protect your Raspberry PI SD card, use Read-Only filesystem, `http://hallard.me/raspberry-pi-read-only/` (2016.8.30 取得).
[10] Broadcom co. : BCM2835 ARM Peripherals, `https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/BCM2835-ARM-Peripherals.pdf` (2016.8.30 取得).
[11] Broadcom co. : BCM2836 ARM Peripherals, `https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2836/QA7_rev3.4.pdf` (2016.8.30 取得).
[12] BeyondLogic : Understanding the Raspberry Pi Boot Process, `http://wiki.beyondlogic.org/index.php?title=Understanding_RaspberryPi_Boot_Process` (2016.8.30 取得).
[13] Broadcom co. : VideoCore IV 3D Architecture Reference Guide, `https://www.broadcom.com/docs/support/videocore/VideoCoreIV-AG100-R.pdf` (2016.8.30 取得).
[14] Gerard Beekmans : Linux From Scratch, `http://www.linuxfromscratch.org/` (2016.8.1 取得).
[15] Gerard Beekmans Bruce Dubbs : Linux From Scratch Version 7.9, `http://www.linuxfromscratch.org/lfs/downloads/stable/LFS-BOOK-7.9.pdf` (2016.8.1 取得).
[16] Ext4-Linux Kernel Newbies, `https://kernelnewbies.org/Ext4` (2016.8.30 取得) .