3A-4

組込み Linux 向け S/W 開発環境構築手法の一提案

井邊　研吾†　　攝津　敦†　　落合　真一†

三菱電機株式会社　　情報技術総合研究所†

1. はじめに

組込み機器の H/W の高機能化に伴い、多くの機能をサポートする高機能 OS を利用することが増えている。その１つが Linux である。Linux はソースコードが開示されており、開発に応じたカスタマイズが可能である。また、組込み機器の H/W 構成は様々であるが、Linux のカーネルコンフィグレーション機能により各種 H/W への適用が考慮されている。このように、H/W 毎にカスタマイズさせ Linux を動作させる実行環境に関する検討はなされている。

しかし、実開発ではリリースした後も長期間保守が行われるため、それを考慮した S/W 開発が必要となる。長期間保守において、ソースコードから過去にリリースしたバイナリと同じものを生成できることが重要であり、本稿では長期間保守を考慮した S/W 開発環境の構築について検討した。

2. 組込み Linux 向け S/W 開発の長期間保守の課題と解決策

実開発での組込み Linux 向け S/W 開発は、数十年間保守する必要がある。このような長期間の保守をする場合、S/W 構成管理と外部データにおいて課題がある。以下に課題の詳細と解決策について述べる。

2.1. S/W 構成管理

開発時と不具合発生時で長い時間が経過している場合がある。このような状況下で不具合解析をする場合、再度実行環境を生成できことが重要である。しかし、カーネルだけ開発当時のソースコードを保持していても、同じ実行環境を作ることができない。

図 1 に示すように実行環境には、カーネルに加えて、ブートローダやユーザランドのバイナリが必要である。ソースコードからそのバイナリを生成するためにはコンパイラが必要となる。組込み S/W 開発では、クロスコンパイラを用いることが一般的である。クロスコンパイラは H/W や Linux の構成により異なり、汎用的でない H/W を用いた場合、クロスコンパイラから開発しなければならない。

このクロスコンパイラも開発時と同じものでなければ、同じソースコードでも同じバイナリが生成できない。このように、ソースコードだけを管理する構成管理を行っていると、不具合時に対応できなくなる問題がある。

このため構成管理には、ソースコードに加えそれらをコンパイルス開発環境も合わせてバージョン管理する必要がある。

A Proposal of a Softwere Development Environment for Embedded Linux
Kengo Ibe, Atsushi Settsu, Shinichi Ochiai
†Information Technology R&D Center, Mitsubishi Electric Corporation

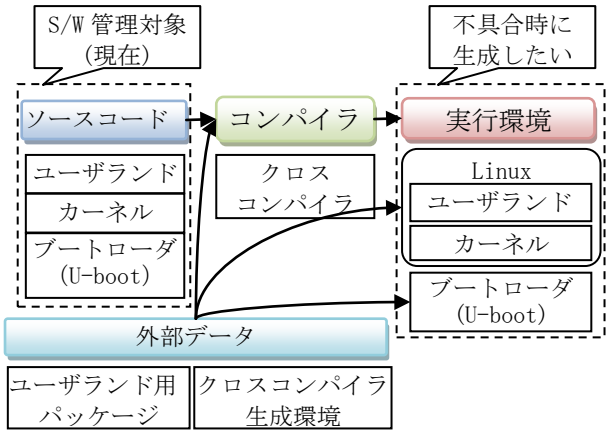

図 1 組込み Linux 向け S/W 開発

2.2. 外部データ保持

再度同じ実行環境を作るためには、ソースコードから生成できない外部のデータも保持しておく必要がある(図 1)。例えばユーザランドを生成するために、ディストリビュータのパッケージを利用している場合にはそのパッケージも保持対象となる。また、クロスコンパイラで使用するライブラリ等も保持しておく必要がある。これらの多くは、利用する時にネットワーク上から取得するが多く、それらが長期間提供される続ける保証がないため、外部データについても全て保持し、バージョン管理を行うことが重要である。

3. 組込み Linux 向けバージョン管理

長期間保守における S/W 構成管理と外部データ保持を考慮したバージョン管理について提案する。

3.1. S/W 構成管理

組込み向け Linux の S/W 構成管理は図 2 のようなリポジトリ構成を提案する。組込み向け Linux S/W 開発のような長期開発・保守をする場合には、S/W 構成パターンである Task Branch [1]を利用するのが適している。この Task Branch を組込み Linux 向け S/W 開発に適用した場合、図 2 の共通メインラインの trunk には新規開発のベースとなるソースコードを保持し、各適用開発を branches で分離する。その branches の中で、各開発の trunk、branches、tags を作成し管理する。共通メインラインの tags には新規開発時のバージョンを記録していく。各開発で機能追加等開発が終了した段階で、共通メインラインがある trunk にマージする。他の開発にも機能追加を適用することで、重複開発を防ぎ、開発コスト削減につながる。また、各開発の保守は共通メインライン一本で管理できるため保守性も高くなる。

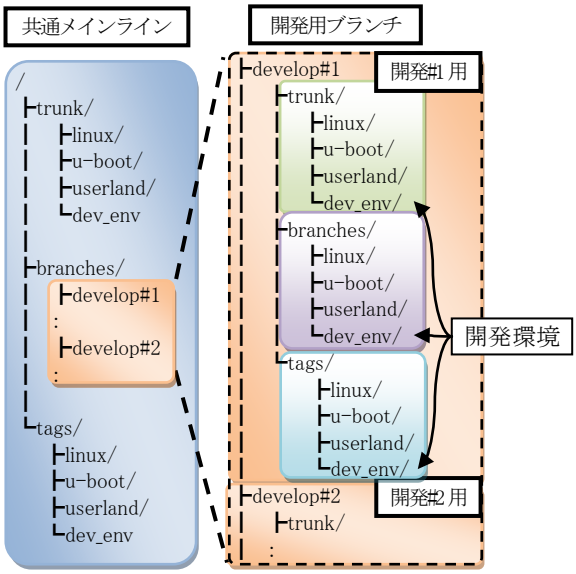

図 2 リポジトリ構成

ここまでは従来の S/W 構成管理と同様であるが、これらに加え各開発で開発環境も管理する。開発環境にはクロスコンパイラや各開発で必要となる外部データも保持し、バージョンを管理する。

開発環境は図 2 の開発#1 用の dev_env に該当する。開発環境で共通的に利用できるような外部データは共通メインラインに保持し、共通化できないものは、各開発の dev_env で管理する。もし、部分的に別の開発の dev_env を共有できるような場合には、各開発間で共有する。

不具合発生時にはリリースしたバージョンのソースコードと開発環境をコピーし、バイナリを生成する。このとき、外部からのデータを必要としないため開発時と同様の物を生成できる。

3.2. バージョン管理方法

S/W のバージョン管理には集中管理方式と分散管理方式があり、長期保守する組込み Linux 向け S/W 開発において、どちらが適切かを検討する。

3.2.1. 集中管理方式と分散管理方式

表 1 に集中管理方式と分散管理方式の比較を示す。

集中管理方式では中央にリポジトリを配置し、そのデータを開発者がコピーして作業する。作業完了後中央リポジトリにコミットすることで、変更履歴を記録する。集中管理方式では、中央リポジトリの 1 つであるため、リビジョン番号をシーケンシャルに割り当てることができ、リビジョンの前後関係が把握しやすい。また、ファイル単位での作業用のコピーができるため、管理の負荷も低く、リポジトリのバックアップも容易に行える利点がある。

一方、分散管理方式では、各開発者がリポジトリを所持する。開発者は別のリポジトリからクローニングして開発を開始する。作業完了後は開発者がローカルリポジトリにコミットして変更履歴を記録する。分散管理方式では開発者個人が、リポジトリを所持するため、一時的な作業の変更履歴も保持することができ、また、他の開発者のリポジトリを自分のリポジトリに反映させることが容易であるため、最新のソースコードを反映しやすい利点がある。

表 1 バージョン管理方法の比較

	集中管理方式	分散管理方式
バージョンの把握	○リビジョン番号がシーケンシャル	△リビジョン番号がハッシュ値
作業用コピー	◎ファイル単位	×リポジトリ単位
リポジトリの保守	◎リポジトリをバックアップする	×リポジトリが複数あり統一して管理できない
開発者間のデータ共有	△データのコピーになるため変更履歴が残らない	◎リポジトリから取り込むため。変更履歴もコピー可能
一時作業の履歴管理	×中央リポジトリにコミットした履歴のみしか残らない	◎開発者個人の履歴を保持できる

3.2.2. 考察

集中管理方式と分散管理方式で大きな違いは、ローカルリポジトリを保持するか否かである。

リポジトリが複数存在するため、リビジョン番号がハッシュ値で生成しており、リビジョンの前後関係が把握しにくいが、ツールを用いて前後のリビジョンを確認できるため、大きな問題にはならない。また、開発者間のデータ共有において集中管理では変更履歴が残らない問題も中央リポジトリを介してデータを共有することで変更履歴を残すことができる。一時作業の履歴管理は、集中管理方式の課題となるが、長期間保守の観点では、開発者個人の作業の履歴が必須ではないため課題とはしない。

長期間保守を考慮した場合、組込み Linux の保守期間の間リポジトリも保守を継続しなければならない。このため、開発者がローカルのリポジトリで作業し、リリースしてしまうような場合、開発者のローカルリポジトリを保持し続けなければならず、管理対象が複数になり、複雑化するため管理ミスが発生する可能性がある。これを防ぐため、中央管理方式を採用することが望ましい。また、リポジトリをローカルにクローニングする場合、今回のようにコンパイラや外部データまで保持しているとリポジトリが膨大になるため、リポジトリを各開発者がクローニングするとデータ量が莫大になる問題もある。

リポジトリ自身の長期間保守の観点から、分散管理方式ではなく、集中管理方式がよいと判断する。

4. おわりに

本稿では、組込み Linux 向け S/W 開発環境において、長期間保守を実現するためソースコードだけではなくコンパイラや外部データも保持する方法について提案した。また、それらのバージョン管理方法としては集中管理方式が適切であることを示した。今後は本提案手法を用いて運用を実施し、運用上の課題について検討していく。

参考文献

[1]Stephen P. Berczuk , Brad Appleton, Software Configuration Management Patterns: Effective Teamwork, Practical Integration, pp.179-184 Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2002