kconfig の多国語表示対応

佐藤嘉則

平成 20 年 8 月 23 日

概 要

Linux kernel のコンフィグレーションに使用する kconfig システムにおいて、多国語メッセージを扱うための拡張を試みた。本論文では拡張した仕様及び実装について紹介する。

1 はじめに

Linux kernel を構築する際にはコンフィグレーションを行う必要があるが、ここで表示されるヘルプメッセージはすべて英語であるため、内容の理解が困難な場合がある。

バージョン 2.4 までの kconfig ではヘルプメッセージを記述してあるファイルが独立していたため、内容を翻訳した同名のファイルに差し替えることで [1] 英語以外のヘルプメッセージを表示することができた。

しかし、現在最新であるバージョン 2.6 では kconfig システムが全面的に書き換えられたため、ファイルを差し替えての多国語表示は不可能になった。その改善策として kconfig を gettextize し、po ファイルを配布しているグループがある [2]。しかし、この様な用途に gettext を使用するのは適切ではないと考えるため、gettext を使用せずに翻訳メッセージを扱えるように kconfig を拡張した。

2 現在の実装における問題点および解決策

2.1 gettext の動作

まず、gettext の動作について簡単に説明する。glibc-2.7 の実装を参考にしている。

1. 指定された msgid に対応する msgstr が既に取得されている場合、それを返す
2. .mo ファイル内を検索し、該当する msgstr を取得する
3. 次回に備えて msgid/msgstr ペアを保存する

上記より msgid をキーとした検索が

- 取得済み msgstr の検索
- .mo 内の msgstr の検索

の、二回行われる事がわかる。取得済み msgstr については tfind(3)/tsearch(3) による二分木でキャッシュされるようになっているので、既に取得したメッセージについてはこの二分木を探索して返すようになっている。

.mo ファイルにある msgstr の検索はハッシュ表が msgstr のインデックスになっており、msgid から求められるハッシュ値で msgstr を取り出せるようになっている。このハッシュ表は msgfmt(1) で.mo を生成する時に作成されており、アルゴリズムとしてはオープンアドレスを採用している。

2.2 効率の問題

前述したグループが配布している po ファイルは msgid が一万を越える巨大なメッセージカタログになっており、これを gettext で処理している。現在の実装ではオープンアドレスのハッシュ表を使用して検索を行っているので、処理量の最悪値がデータ量に比例してしまう。ハッ

シュ値の衝突が少なければそれほどの影響は無いと思われるが、現状のハッシュ関数では平均して50%程度の衝突が確認できた。したがって、カタログ内にあるmsgidの数が効率に大きく影響すると考えられる。ただし、キャッシュとして使用する二分木は赤黒木として実装されているので(glibc-2.7)、一回読み込んだメッセージについては高速に取得できる。しかし、kconfigでは複数回参照されるメッセージはそれほど多くないため、キャッシュの効果は期待できない。

2.3 用途の問題

gettextは実行形式に埋め込まれている文字列を動的に変更する事を目的としているので、動的に読み込んでいるメッセージの翻訳に使用するのは、正しい使い方とは言えない。

2.4 解決策

メッセージカタログが巨大になってしまうのは、*Kconfig*ファイル内に記述されている項目名とヘルプメッセージを含んでいるのが原因である。これらのメッセージが記述されているファイルは実行ファイルと独立しているので、実行時に指定された言語のメッセージデータを読み込めば、gettextを使用しなくても各種言語の表示が可能であると考えた。また実際に日本語を埋め込んだ*Kconfig*ファイルを作成し正常に表示される事を確認した。

3 kconfigの調査

この推測が正しいことを確認するために、実際にコンフィグレーションを行うkconfigの調査を行った。

3.1 内部構造

まず、現状のkconfigの構造について簡単に解説する。基本的な構造としてはフロントエンド部分とバックエンド部分に分けらる。図1に大まかな構造を示す。フロントエンドはUIを担当しており、各種UIを提供するために複数用意されている。バックエンドは設定ファイルの解析や構成情報の管理を担当している。

3.2 バックエンドの動作

まずバックエンド側がどのようにメッセージを扱っているか調査した。

1. *Kconfig*を読み込み構文解析を行う。解析結果は構文木として保存される。また既存の*.config*を読み込み設定の初期値とする。

2. その後フロントエンド側からの要求に対し、条件式や依存関係を評価した結果設定可能な項目や現在の設定値、ヘルプ文章を返していく。設定変更できない項目については無視される。

3. フロントエンドからユーザーの項目選択状況が通知されるので、設定値を更新していく。

4. 終了時に設定値から*.config*ファイルを生成する。

3.3 フロントエンドの動作

フロントエンド側では、取得したメッセージをmsgidとしてgettext関数を実行し、得られたテキストを表示ライブラリ(curses,GTK,qt)に渡して表示を行っている。メッセージに対して特別な処理は行っていないので、表示ライブラリが扱える文字であれば特に制限はないと思われる。

上記の結果から、前節での推測通りバックエンドが読み込む*Kconfig*内のメッセージをあらかじめ翻訳しておき、フロントエンド側に渡すのが国際化の手法として妥当であると考えた。

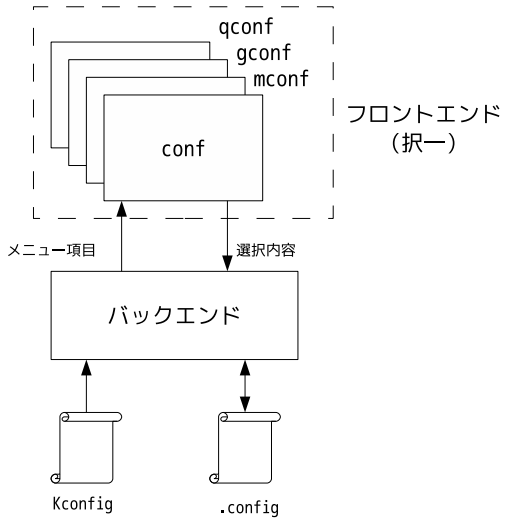

図 1: kconfig の構造

4 翻訳メッセージファイルの扱い

次に翻訳したメッセージファイルの取扱いについて検討を行った。

4.1 要求事項

以下の点を考慮して仕様を検討した。gettext を使用した場合と比較しても制限が厳しくならないように考慮している。

- kernel source tree 内部で完結する
  source tree の外部にファイルを置く場合 (別ディレクトリに.mo をインストールする等)、ビルド手順が変更になってしまう可能性がある。またパッケージビルドシステムを使用する場合に、作業用ディレクトリ以外に書き込んでしまう事になるため、余計な手間がかかる。

- 複数言語への対応
  mainline へのマージを考えると、世界中で利用されるために特定言語へ依存することは好ましくない。

- 元の *Kconfig* を変更しない
  現在の *Kconfig* に新たに翻訳メッセージを付加してしまうと、環境によっては正しく編集できなくなってしまう可能性が考えられる。これは mainline へのマージを考慮した場合に問題になると思われる。

- バージョン依存性の排除
  翻訳ファイルが特定バージョンに依存してしまうと、
  - 更新後に翻訳が追いつくまでまったく使えなくなる
  - 常に最新版への追従が必要なので作業者の負荷が上がってしまう

  などの問題が発生する。この問題は利用者にも負担を強いてしまう。

4.2 検討結果

検討を行った結果、以下のような仕様を決定した。可能な限り限り現在のコードを変更しないで対応する事を考慮している。

- 翻訳済みメッセージを別ファイルに用意し、元ファイル名+言語コードのファイル名を付けて、元ファイルと同じディレクトリに配置する。これにより、ソースツリー内での完結、複数言語への対応、元

の *Kconfig* を変更しないを実現できる。また、既存ファイルの修正が不要なので、別配布の状態であっても

```
$ tar xjf linux-2.6.foo.tar.bz2
$ tar xjf kconfig-ja-2.6.bar.tar.bz2
```

というように上書きするだけでよいので、取扱いが容易になる。

- メッセージファイルの文法は通常の *Kconfig* ファイルと同じ物を使用し、メッセージ部分のみを翻訳する。メッセージに無関係な”default”や”depends on”などは記述されていても無視する（オリジナル *Kconfig* の記述が有効になる）。これによりバージョンアップなどで記述が食い違ってしまった場合でも、メッセージ部分はそのまま使うことができる。

この仕様で要求は完全に満たすことが出来ると判断した。

なお以下の仕様については今後検討していくことにした。

- 国コードの扱い
- 翻訳ファイルの文字コード

この部分については、筆者に国際化に関する知識が不足しており十分な検討が行えていない。

4.3 翻訳の例

以下にこの仕様に基づいて翻訳した場合の例を示す。

- *Kconfig*

```
config FOO
   bool "use FOO"
   default y
   help
     FOO is ...
```

- kconfig.ja

```
config FOO
   bool "○○を使用する"
   default y
   help
     ○○は…
```

5 試験実装

検討した仕様の妥当性を評価するため、現状の kconfig を改造し翻訳メッセージを扱えるようにしてみた。現在の kconfig からの差分とテスト用の翻訳ファイルを公開 [3] している。検討時の予想通りコードの修正量はそれほど多くない。

5.1 変更点

以下の処理を追加した。

1. *Kconfig* ファイルの解析が終わった後、翻訳ファイルを解析する
   これで翻訳された構文木が生成される。
2. 翻訳構文木の節にあるテキストを、同じ項目を持つ英語の節にコピーする
   これで翻訳文章が出力されるようになる。翻訳が存在しない場合はそのままになる。
3. すべての節をコピーしたら翻訳の枝を切る
   つながっているとメニュー項目が重複してしまう。

処理のイメージを図 2 に記しておく。

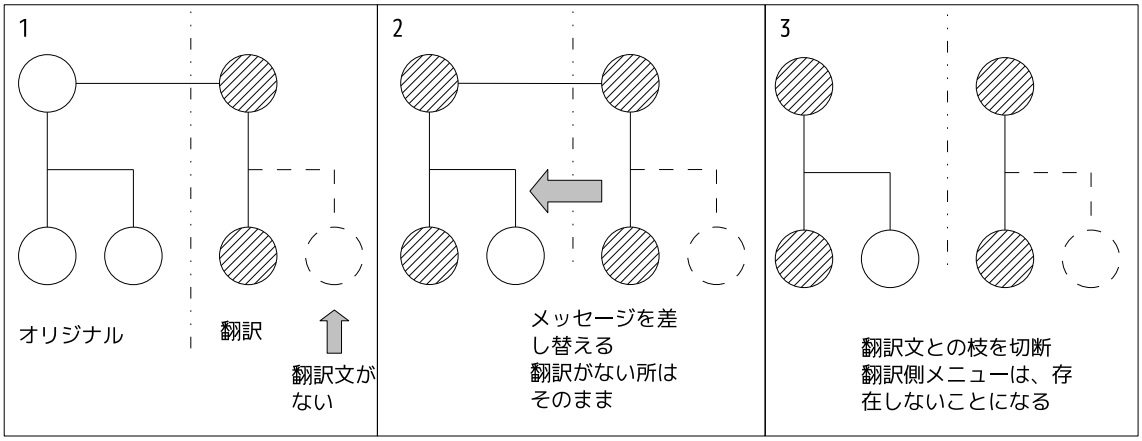

図 2: 追加処理の概要

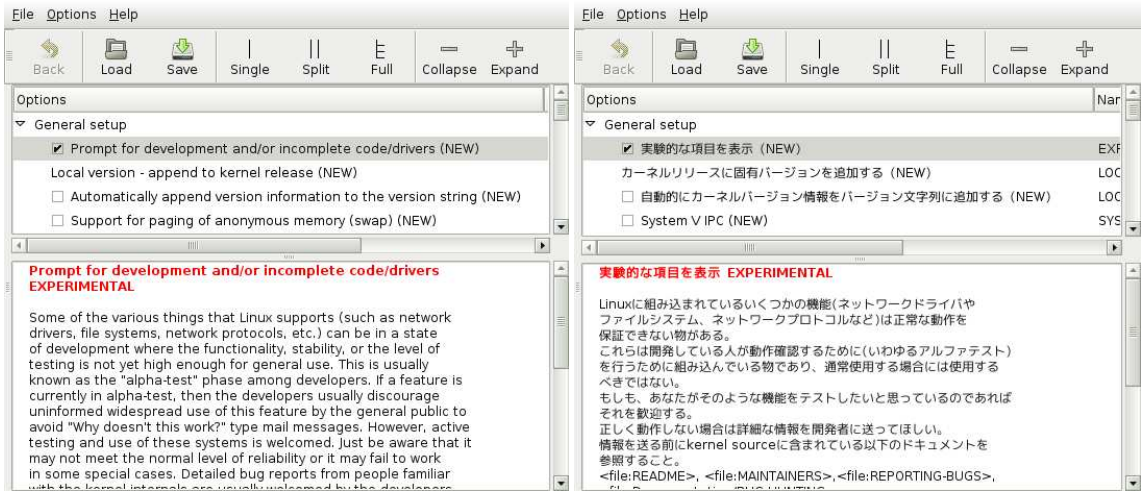

図 3: 実行結果　(右 - 変更前、左 - 変更後)

当初の予定では英語の *Kconfig* と翻訳 *Kconfig* を平行して読み込み、メッセージを置き換えていく予定であったが、現状の構文解析器の実装を流用して行うのは困難であるため、英語 *Kconfig* を読み込んだ後に翻訳 *Kconfig* を読み込み、メッセージを一括して置き換える方法にした。そのために生成される構文木が巨大になり効率がよくないので、将来的には修正の必要がある。

5.2　評価

修正 kconfig と翻訳ファイルを用いて実際に日本語化してみた結果を図 3 に示す。

項目名とヘルプ文章が日本語で表示されている事を確認できる。英語のまま表示されている部分は日本語訳を用意していない項目である。この様に翻訳ファイルが完全に対応していない場合でも問題なく設定を行うことが出来る。これにより翻訳のタイムラグで翻訳ファイルを使用できない状況は発生しない。

以上のように概ね満足する結果を得ることが出来た。

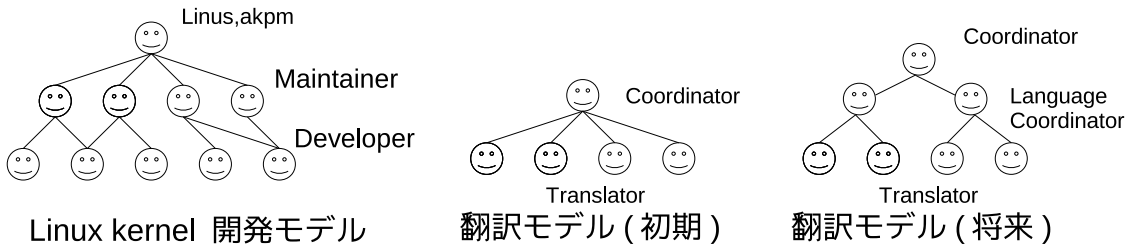

図 4: 作業モデル

6 翻訳

6.1 *Kconfig* ファイルの現状

各ディレクトリに含まれる *Kconfig* ファイルの数を表**??**に示す。最終的にはこれだけの翻訳を行うことが目標になる。arch と drivers が全体の約 8 割を占めているが、これは Linux が多数のアーキテクチャ・ドライバをサポートしているためと推測される。

まずは複数の翻訳者と取りまとめ役の二階層でスタートし、複数の言語について同様な組織が出来上がった所で三階層のモデルに移行するのが適切でないかと考える。また、この段階まで成長すればその他の国際化関連の作業全般 (ドキュメント翻訳など) もこのチームで担当出来るようになるのではないかと期待する。筆者には翻訳プロジェクトの参加経験がないので、この部分に関してはそういった方面からのフィードバックに期待したい。

6.2 作業体制

前項より、かなり大量の *Kconfig* ファイルについて翻訳する必要がある事がわかる。ただ、数が多いものの個々のファイルについてはそれほど大きな物でなく相互の依存関係もないので、複数人が分担して作業することも容易であると考える。図 4 に想定している作業モデルを示す。

7 今後の展開

kconfig 本体の国際化については

- 日本語以外での動作検証
- 効率の改善

といった作業が必要であると考えている。
また

- 翻訳メッセージ仕様の確定
- メッセージの翻訳

については外部の協力を得て進めていきたい。

これらの作業がある程度完了した時点で kernel 開発コミュニティへ提案を行い、標準機能として取り込まれるよう活動していきたいと考えている。

8 最後に

最近ではディストリビューションによって提供される kernel をそのまま使っている人が多いと思われるが、メッセージが母国語で表示されることで、kernel 再構築の敷居を少しでも下げることが出来れば幸いである (言語以外の問題もあるが、英語であるというのはそれなりに障害であると思う)。また、システムの根幹にある kernel を自分で再構築することで、kernel やシステムの内部に興味を持ってくれる人が少しでも増えてくれることを願う。

参考文献

[1] Configure.help Japanese translations. http://www.linux.or.jp/JF/JFdocs/Configure.help/

[2] The Linux kernel Transration Project http://tlktp.sourceforge.net

[3] 国際化対応パッチ http://uclinux-h8.sourceforge.jp/kconfig-i18n/kconfig-i18n-test.tar.gz