# Ubuntu14.04 リアルタイム化概要説明

2016/3/30

株式会社アイザック

## 1. リアルタイム化概要

OS のリアルタイム化を行こうとで、規定された周期でのプログラム動作が可能となり、より正確な制御を実現することができる。

Ubuntu ではリアルタイム化を行うためには、ハードリアルタイム（-realtime,-rt）のカーネルをインストールする必要がある。Ubuntu12.04 でのリアルタイムカーネルのインストールはコマンド（apt-get install linux-rt）により容易に実行できる。しかしながら Ubuntu 12.04 は 2017 年の 4 月にはサポートが切れてしまう。そのため、サポート期間の長い（2019 年の 4 月まで）Ubuntu 14.04 のリアルタイム化を行う。

Ubuntu 14.04 ではインストールコマンド（apt-get install linux-rt）が機能しないため、バニラカーネルへのリアルタイム化パッチの適用、カーネルのビルド、ビルドしたカーネルのインストールを行う必要がある。

リアルタイム化には RTC のリアルタイム化対応も必要である。詳細は OpenRTM 公式サイトや以下フォーラムを参照のこと。

[openrtm-users 02918] OpenRTM-aist 用のリアルタイム性を強化した実行コンテキスト

## 2. 開発環境・使用カーネル

使用したカーネルを Table 1 に示す。バニラカーネルは linux-3.12.48、リアルタイム化パッチには patch-3.12.48-rt66 を使用した。また、セキュリティパッチとして apparmor compatibility 3.12 を使用した。カーネル、リアルタイム化パッチには多くの種類があり、この組み合わせのみでしかビルドできないという事はない。

**Table 1 OS・Karnel**

| 名称 | データ型 |
|---|---|
| OS | Ubuntu14.04 |
| Linux バニラカーネル | linux-3.12.48 |
| リアルタイム化パッチ | patch-3.12.48-rt66 |
| セキュリティ関連パッチ | apparmor compatibility2.10<br>3.12 用パッチを使用 |

## 3. ビルド・インストール方法

*ビルド方法

・linux-3.12.48 をダウンロード。任意の場所に展開・フォルダに入る

$ cd linux-3.12.48

・patch-3.12.48-rt66 をダウンロード。linux-3.12.48 フォルダ内に入れる。

・カーネルにリアルタイム化パッチをあてる。

$ zcat　patch-3.12.48-rt66.patch.gz | patch -p1

・apparmor-2.10 ダウンロード。

/apparmor-2.10/kernel-patches/3.12

内の以下パッチを linux-3.12.48 フォルダ内に入れる。

0001-UBUNTU-SAUCE-AppArmor-basic-networking-rules.patch

0002-apparmor-Fix-quieting-of-audit-messages-for-network-.patch

0003-UBUNTU-SAUCE-apparmor-Add-the-ability-to-mediate-mou.patch

・セキュリティパッチ apparmor をあてる

patch -p1 < 0001-UBUNTU-SAUCE-AppArmor-basic-networking-rules.patch

patch -p1 < 0002-apparmor-Fix-quieting-of-audit-messages-for-network-.patch

patch -p1 < 0003-UBUNTU-SAUCE-apparmor-Add-the-ability-to-mediate-mou.patch

・config のコピーを行う。/boot 中にある使用中のカーネルの名前がついた.config のファイルを linux-3.12.48 フォルダ内に入れる。

・linux-3.12.48 フォルダ内で以下コマンド

$make oldconfig

・カーネルの設定を行う。linux-3.12.48 フォルダ内で以下コマンド

$make xconfig

Fig.1 に示すような UI が表示される。以下項目を設定

General setup TimerSubsystem の High Resolution Timer Support を選択。

Processor type and features の項目中の Preemption Model 内 Fully~(RT)を選択。

Power management and ACPI options の項目全てチェックを外す。

Security Options の項目で apparmor 該当部にチェックが入っていることを確認。

設定終了後保存して終了

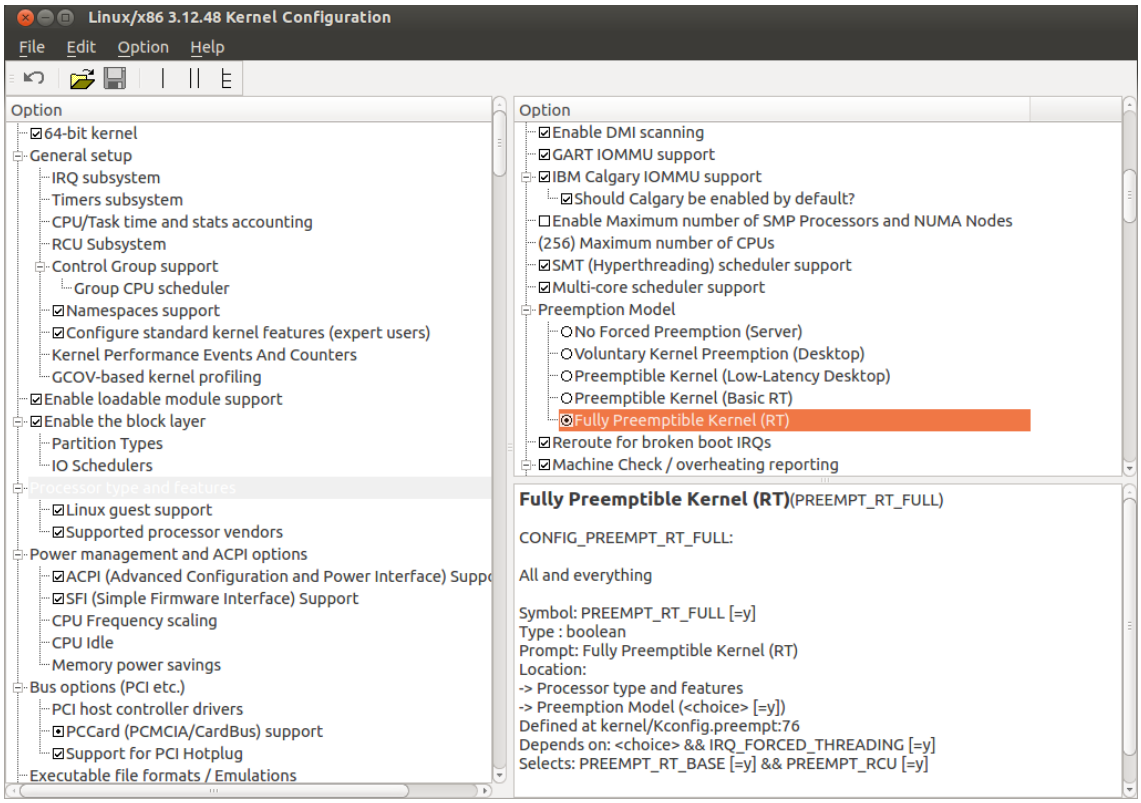


**Fig. 1 xconfig**

・linux-3.12.48 フォルダを/usr/src フォルダへ

カーネルのビルドを行う。/usr/src/ linux-3.12.48 に入り、以下コマンドを実行

$ sudo make-kpkg --initrd kernel_image kernel_headers –j8

末尾の –j8 はジョブ数である、並列実行によりビルド時間を短縮する。任意の数字に変更してよい。

また、--revision オプションでカーネル名を変更しておくとわかりやすい。

・ビルド終了

*インストール方法

・ビルド終了後 .deb ファイルが 2 つ作られる。それぞれ以下コマンドでインストールする。

$ sudo dpkg –i　ファイル名.deb

・再起動後インストールしたカーネルが起動時に選択できる。

## 4. 制御周期精度実験

上記によりリアルタイム化した OS 上で RTC のリアルタイム化（実行コンテキスト変更）を行い、制御周期の精度実験を行った。(RTC には実行コンテキストと呼ばれるファイルがあり、リアルタイム化を行う際はこのコンテキストの変更が必要になる。標準コンテキストのままでは OS をリアルタイム化しても RTC はリアルタイムには動かない。詳細は OpenRTM 公式サイト等参照のこと。)

制御周期を出力するのみの RTC を製作、1000HZ（0.001s 周期）で動作し実際の周期との差を確認した。比較対象として、リアルタイム化していない OS 上で標準コンテキストにより RTC を動作させた場合と、リアルタイム化した OS 上で標準コンテキストにより RTC を動作させた場合の実験も行った。実験結果を Fig. 2 に示す。グラフ中の青で表示されているものが、OS のリアルタイム化をせず、RTC も標準コンテキストの物を使用した際の実行結果である。緑で表示されているものが OS はリアルタイムだが、RTC は標準コンテキストの物を使用した際の実行結果である。赤で示しているものが、OS・RTC 共にリアルタイム化を行った際の実行結果である。リアルタイム化を行った赤のグラフが目標周期の 0.001s 付近から大きくぶれることなく動作しているのに対し、他のリアルタイム化がなされていないものは目標周期から大きくぶれがあることが分かる。各実行結果の平均値と標準偏差を Table 2 に示す。OS・RTC のリア

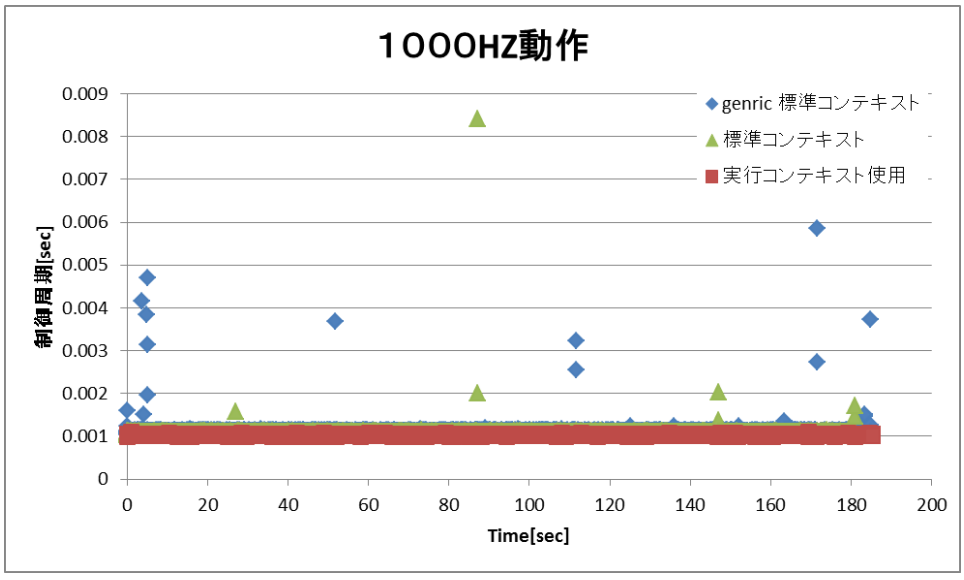


Fig. 2 Result

Table 2　Average and Standard Variation

| 項目 | 平均値[sec] | 標準偏差 |
|---|---|---|
| 理想値 | 0.001000 | 0 |
| generic＋標準コンテキスト | 0.001100 | 0.000023 |
| リアルタイム化＋標準コンテキスト | 0.001088 | 0.000020 |
| リアルタイム化＋実行コンテキスト変更 | 0.001035 | 0.000007 |

ルタイム化によって平均値、標準偏差共にリアルタイム化以前の環境より正確な制御周期を実現できていると言える。

改版履歴

| Ver | 改定日 | 内容 |
|---|---|---|
| 0.0 | 2016/3/30 | 新規作成 |
| | | |