第 24 回 東京エリア Debian 勉強会資料

東京エリア デビアン 勉強会

Debian 勉強会幹事 上川純一

2007 年 1 月 20 日

1 Introduction

上川純一

今月の Debian 勉強会へようこそ。これから Debian のあやしい世界に入るという方も、すでにどっぷりとつかっているという方も、月に一回 Debian について語りませんか？

目的として次の二つを考えています。

- メールではよみとれない、もしくはよみとってられないような情報について情報共有する場をつくる
- Debian を利用する際の情報をまとめて、ある程度の塊として整理するための場をつくる

Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作るスーパーハッカーになった姿を妄想しています。

Debian をこれからどうするという能動的な展開への土台としての空間を提供し、情報の共有をしたい、というのが目的です。

デビアン勉強会

目次

2 事前課題

上川純一

今回の事前課題は「今後、勉強会につかう施設を提案してください」と「２００７年の勉強会の各月のアジェンダを提案してください」というタイトルで 200-800 文字程度の文章を書いてください。というものでした。その課題に対して下記の内容を提出いただきました。

2.1 Ishihara さん

2.1.1 「今後、勉強会につかう施設を提案してください」

角筈地域センター `http://www2.odn.ne.jp/~hak35040/index.html` 公共の施設が安くて便利そうな気がします。ただ、電源や LAN とかそういう設備が厳しいのかもしれません。個人的にはこういうところでの勉強会もやってみたいかも。`http://www.tef.or.jp/oshima/index.html`

2.1.2 「２００７年の勉強会の各月のアジェンダを提案してください」

・古い PC 復活大作戦

1. 古いマシンを Debian で復活させる: 古いマシン独特の問題
2. デスクトップ機にしてみる: X つかうよ or X つかわないよ
3. サーバにしてみる: セキュリティの問題

みたいな感じで、ちょっと型落ちなマシンを復活させる、という実践的？な企画があってもいいかなぁと思います。1 年でサーバマシンが立てられるよ、セキュリティもちゃんと配慮できるよ、みたいな企画ですね。

あとは、Debian に関係する英語を読んで見ましょう企画とか。

・vs 英語

1. Debian の英語サイトを読んでみよう: キーワードの見つけ方、 英語の辞書の使い方、 どのドキュメントを読めばいいの？

やっぱりネックとなるのは英語なメッセージやドキュメントだと思います。ググれ、と突き放すのもひとつの勉強法なんですけど、ちょっとでも読み方がわかると、敷居が低くなるんじゃないのかなぁと思うのですが。どうなんでしょうか？

2.2 小林さん

2.2.1 『今後、勉強会につかう施設を提案してください』

これまで何回か「大学は利用できないのか」という話があったこともあり、大学所属のメンバーとして、勉強会に使える施設が大学内にないか検討してみました。

まず、情報教育のための施設である情報教育棟[*1]については、『情報教育棟会議室・セミナー室利用申込』[*2]の利用規定第 2 条 (1) に、教職員が学部、専攻（系)、学科、部会、センター、委員会、事務部が主宰する会合、とありました。学生がサークルのようなものの集まりとして使用するような使い方はできないようです。まぁ、得体の知れない人間にコンピュータやネットワークを自由に利用されても困りますので、仕方がないでしょうか。

次に、サークルなどで使われている東京大学駒場コミュニケーション・プラザ[*3]も検討してみました。こちらの利用規則[*4]には次のようにあります。

(中略)

例外的に適当と認められるために何をすればよいのかよくわかりませんが、それなりの手続きを経なければいけないようです。

大学という場はオープンで結構使いやすいかと思われがちですが、オープンすぎて勝手に使われると問題になるので案外色々と制限 (「主催者が教職員」とか「構成員が学内の人間」とか) がかかっていることが多いです。昨年 11 月の大阪電気通信大学での勉強会のように教員が誘致してくださったりする場合はやりやすいでしょうが、そうでない場合はなかなか使いにくい場ではないでしょうか。まだまだ探す余地はあると思うのでもう少し粘ろうかと思いますが……。

2.2.2 『2007 年の勉強会の各月のアジェンダを提案してください』

2 月 「今年、私は Debian にこのように関わろうと思います／○○を成し遂げます!!」宣言大会
3 月 Debian etch リリース記念インストール大会
4 月 Debian lenny に向けて日本語圏でやっておきたい事項のリストアップ
5 月 Debian を使っていて思う「このようなフリーなアプリケーションが足りない」
6 月 Debian Conference 7 現地報告
7 月 Debian Conference 7 報告
8 月 Debian でのウェブアプリケーションのあれこれ
9 月 Debian でのグラフィックアプリケーションのあれこれ
10 月 Debian の派生ディストリビューションのあれこれ
11 月 Debian と私
12 月 「今年、私は Debian に関して○○を成し遂げました／成し遂げられませんでした」報告・反省大会

2.3 前田さん

2.3.1 「今後、勉強会につかう施設を提案してください」

二箇所提案します。

1. 晴海区民館: 晴海トリトンにあります。ちょっと高いのが難点です。http://www.city.chuo.lg.jp/sisetugaido/syukaisisetu/syukaisisetu17/
2. 月島区民館宴会を月島のお好み焼きや、もんじゃ焼きに絞る場合はこちらの方が良いかも。(ただ、月島の店は閉店が早いです。23 時にしまっちゃいます) http://www.city.chuo.lg.jp/sisetugaido/syukaisisetu/syukaisisetu14/index.html

あまり現実的ではありませんが、私の住んでいるマンションの共有施設、という手もあります。利用料は安いのですが（300 円 or500 円/ 1 時間)、交通の便があまりよくありません。あと、近くに飲み屋どころか、飲食店がほとんどありません。（あっても２３時で閉店なので）だからといってウチで飲み会、というのは困るので…。

2.3.2 「２００７年の勉強会の各月のアジェンダを提案してください」

バッティングしそうなのもあるので、ネタだけ列挙します。

[*1] http://www.edu.c.u-tokyo.ac.jp/
[*2] http://www.edu.c.u-tokyo.ac.jp/etc/application.htm
[*3] http://www.com-pla.com/
[*4] http://www.com-pla.com/kitakan/riyo.html

1. Debian でだってできる 3D デスクトップ s　裏番組（Vista）をぶっ潰せ！
2. Etch リリース（仮）を祝って、Sarge → Etch のアップグレード苦労話
3. 新入生・新入社員を Debian ユーザにする企て
4. 打ち上げ花火で Debian のグルグルを実現できるか（趣旨違いますね）
5. Debian 勉強会@ Linux World 又は LC（って今年も開催するのか？）
6. 他のユーザ会とのコラボ（特にターゲットは無し）

2.4 岩松さん

2.4.1 「今後、勉強会につかう施設を提案してください」

自分の勤務地の最寄り駅である国分寺駅近辺の施設を調べてみました。いいところがありませんでした。

2.4.2 「２００７年の勉強会の各月のアジェンダを提案してください」

- 2 月 パッケージ依存関係を図にして印刷してみる: apt-cache dotty で出力された dotty ファイルをプリントアウトして満足する。
- 3 月 東京青少年オリンピックセンター 合宿 and OSC 2007 spring: 合宿をする。OSC に参加して本を売る。
- 4 月 dbs を調べてみる: あまりメジャーではない dbs について熱く語る。
- 5 月 defoma を解析して、熱く語る。: 山根さんが defoma を解析して、熱く語る。
- 6 月 debconf and OSC DO 2007: debconf 組がエジンバラに行っている間、残った人はみんな北海道で合宿。
- 7 月 apt-xxx を調べる: 実はみんなが知らない apt-xxx があるのではないか？調べて熱く語る。
- 8 月 ラインセンスを調べる and コミケ: Debian パッケージの ライセンスを全て調べてみる。コミケをがんばる。
- 9 月 OSC 2007 Tokyo Fall に参加: OSC に参加する。
- 10 月 Debian で 動画再生環境を作ってみる: main セクションだけで動画再生環境を作ったらどうなるか、調べる。
- 11 月 KOF に参加: KOF に参加して、○○[*5]さん実家に泊まる。
- 12 月 反省会 and コミケ: コミケをがんばる。反省会もちゃんとする。温泉合宿を決行する。

2.5 北原さん

2.5.1 今後、勉強会につかう施設を提案してください

現在と同じような勉強会（曜日・時間・人数・プロジェクター等の施設）が出来る施設を探してみました。現在利用している施設の使用料がわからないので、参加費と人数、資料代等を適当に勘案して、１万円以下を条件としました。

そうすると、民間の物件はほとんど不可能なので、公共の施設となりますが、区の施設では利用者の制限が厳しい（半数が区民である必要がある等）ので都の施設を調べてみました。

以前の勉強会で話題に挙がった、東京体育館、同じ団体で管理している東京武道館・駒沢オリンピック公園総合運動場・東京辰己国際水泳場（全て会議室あり）を除くとあまり良いのは見つかりませんでした。

その中でも「東京スポーツ文化館・BumB」が、利用条件が緩く、2,625 円からで利用できそうです（場所は夢の島）。　勉強会に「男女平等の推進に関する」テーマが盛り込めれば、「東京ウィメンズプラザ」（場所は神宮前）が利用できるかも？（笑）（3,300 円から）

夜でなくても良ければ「東京国際ユースホステル」の研修室が利用でるかも？（飯田橋の駅側、3,000 円）

あと、利用方法が調べきれていませんが、清澄庭園・蘆花恒春園でも集会所が借りられるようですが、ちょいと勉強会とは雰囲気が違うような気がする。

国関係も調べようと試みましたが、まったく引っかかりませんでした。（調べ方が悪かったかも）

あと、「東京スポーツ文化館・BumB」は宿泊施設があって最大 251 人（定員）泊まれます。　DebConf の検討対象にはなりませんかね？

[*5] 編集注：不適切な表現があったため、削除）

2.6 小室さん

2.6.1 今後、勉強会につかう施設を提案してください

練馬区在住なので、練馬区施設関連を中心に探しました。

貫井地区区民館[*6] 最寄：中村橋駅（西武池袋線）徒歩 5 分 1 時間 400 円

光が丘地区区民館[*7] 最寄：光が丘駅（大江戸線）1 時間:500 円

石神井公園区民交流センター[*8] 最寄：石神井公園（西武池袋線）会議室（1）：1200 円

練馬女性センター[*9] 最寄：石神井公園駅（西武池袋線）第 1 研修室：1400 円

後は学校開放というのをしていて、会議室がある小学校が夜間貸し出しをしている所があるようです。[*10] 夜間は 18 時から 21 時で、大体 600 円ぐらいです。

練馬まで来てくれるかっというのが最大の問題ですね …

2.6.2 2007 年の勉強会の各月の agenda

暗号化とは !? とか ‥ 自分で調べたら分かるよなぁという事もただあるので、なかなか思いつかないですね。「これだけは絶対やってはいけない操作」とかどうでしょう。後は個人的に Debian policy は勉強したいと思います。ちゃんとした回答になっていなくてすいません …

[*6] http://www.city.nerima.tokyo.jp/chiiki/chikukan/kakukan/nukui.html

[*7] http://www.city.nerima.tokyo.jp/chiiki/chikukan/kakukan/hikari.html

[*8] http://www.city.nerima.tokyo.jp/kouryu/

[*9] http://www.city.nerima.tokyo.jp/jinken/indexj.html

[*10] http://www.city.nerima.tokyo.jp/nerima_sg/manabi/kaihou.html

3 Debian Weekly News trivia quiz

上川純一

ところで、Debian Weekly News (DWN) は読んでいますか？Debian 界隈でおきていることについて書いている Debian Weekly News. 毎回読んでいるといろいろと分かって来ますが、一人で読んでいても、解説が少ないので、意味がわからないところもあるかも知れません。みんなで DWN を読んでみましょう。

漫然と読むだけではおもしろくないので、DWN の記事から出題した以下の質問にこたえてみてください。後で内容は解説します。

3.1 2006 年 42 号

`http://www.debian.org/News/weekly/2006/42/` にある 2006 年 12 月 26 日版です。

問題 1. Linux Conference Australia で開催される Debian 関係のイベントは今回で何回目か

☐ A 1

☐ B 3

☐ C 6

問題 2. 最近急上昇して Debian 内で 3 位人気のアーキテクチャになったアーキテクチャは？

☐ A ARM

☐ B PPC

☐ C AMD64

問題 3. etch がフリーズされたのはいつ？

☐ A 11/11

☐ B 12/11

☐ C 12/24

問題 4. Debian のインストール CD イメージはどれくらいの頻度で更新されているか？

☐ A 毎日

☐ B 毎メジャーリリース

☐ C 毎マイナーリリース

4 最近の Debian 関連のミーティング報告

上川純一

4.1 東京エリア Debian 勉強会 23 回目報告

東京エリア Debian 勉強会報告。12 月の第 23 回 Debian 勉強会を実施しました。Bug Squashing Party について報告するのと、一年間の Debian 勉強会を反省する会です。

今回の参加人数は 15 人でした。あけどさん、小室さん、小林さん、岩松さん、Henrich さん、前田さん、澤田さん、キタハラさん、青木さん、みつかさん、えとーさん、でんさん、野首さん、gotom さん、上川でした。

まず、事前課題の紹介をしました。来年のスタイルについてのみなさまの提案をいただきました。今 Debian の各種インタフェースがどうなっているのかドキュメントを作成している、それには価値がある、という話や、会場で実習する「ハンズオン」ができると、忘れないうちに体験できるのでよいよね、という話題、ネットワークの事前課題で出てきたような特定のトピックにまつわるバッドノウハウをたくさん集めるというのを今後もやりたいというのが出てきました。また、用語がわからないというかまったくついていけない場合があるのだが、そういうのをログをとっておいて後から質問できるようになっているとよいのではないかという提案などがありました。

岩松さんと gotom さんが、先日開催した Bug Squashing Party についての報告を実施しました。今回の BSP では、etch リリース直前ということもあり、難しいバグしか残っていないことで、日本のメンテナのバグを修正するという方向を出してみました、とのことです。BSP をやるまえに課題を洗い出してくださいというメールを出すのはよいのではないかという提案がありました。今後も開催したいので BSP ができる会場を探したいそうです。まだよい会場をみつけれていないのですが、例えば二ヶ月に一回くらい開催するために場所を探しているというメールを出したら誰か良い場所を教えてくれるのではないか、というのが希望的観測でした。

上川が今年一年のアクティビティーの結果をまとめて報告しました。ふりかえってみるといろいろとみえてきます。今年は結果として 7 回は通常運用の方法で開催していましたが、6 回は場所を外部にとって違う場所で開催しているということが判明しました。来年も半々くらいでやっていくのがよいんじゃないかと思います。1 月に実施内容を決定するようにしましょう、という話になりました。

宴会は荻窪 うさぎにて。素敵なお店でした。

4.2 Debian Conference 参加案内

Debian Conference7 が開催されます。参加者は 1 月末日までに登録してください。

場所	イギリス エジンバラ
日時	2007 年 6 月 17 日-23 日
ウェブページ	`http://debconf7.debconf.org`

5 apt-torrent

岩松 信洋

Debian パッケージのレポジトリを bittorrent ネットワーク上に置き、それを apt で取得するというものが apt-torrent です。

まだ Debian としては頒布されておらず、フランス人の方が一人でシコシコやっているようです。http://sianka.free.fr/ で開発が行われています。

5.1 bittorrent とは

bittorrent とは、P2P を用いたファイル転送用プロトコルとその通信を行うソフトウェアの事を指します。特徴としては、P2P でファイルの一部をお互いに送受信しあうというプロトコルになっているところです。

通常の P2P ソフトウェアではファイルを提供元に集まるようになっているが、このプロトコルを使うことにより、ネットワーク帯域のないピアもファイルの配布に協力できるようになっています。

このプロトコル上で Debian パッケージを頒布し、apt で取得するようにしたものが apt-torrent です。

5.2 使い方

5.2.1 ダウンロード

先にも書いたように Debian としては頒布されていません。

以下のサイトからダウンロードし、使用します。

http://sianka.free.fr/download/apt-torrent_0.5.0-1_all.deb

apt のソースとして利用できる apt-line も提供されており、以下の apt-line を/etc/apt/sources.list に追記して使用可能です。

- debhttp://sianka.free.fr/apt-torrenttestingmain
- deb-srchttp://sianka.free.fr/apt-torrenttestingmain
- debhttp://sianka.free.fr/apt-torrentunstablemain
- deb-srchttp://sianka.free.fr/apt-torrentunstablemain

5.2.2 設定

パッケージをインストールすると、自動的に /etc/apt/sources.list に追記されます。

以下が追加された apt-torrent 用の apt-line です。

```
### BEGIN APT-TORRENT SOURCE LIST
# Do not edit or remove the markers, they are used by the apt-torrent package

# Uncomment one of the following:
# deb http://127.0.0.1:6968/debian/ unstable main
# deb http://127.0.0.1:6968/debian/ testing main

### END APT-TORRENT SOURCE LIST
```

自分の環境に合わせて apt-line のコメントを外します。

5.2.3 apt-get update してみる

apt-torrent 用の apt-line だけ有効にし、apt-get update を実行してみます。

```
iwamatsu@chimagu:~$ LANG=C sudo apt-get update
Get:1 http://127.0.0.1 unstable Release.gpg [189B]
Get:2 http://127.0.0.1 unstable Release [776B]
Ign http://127.0.0.1 unstable/main Packages/DiffIndex
Get:3 http://127.0.0.1 unstable/main Packages [27.9kB]
Fetched 28.9kB in 3s (8118B/s)
Reading package lists... Done
```

5.2.4 何かパッケージをインストールしてみる

いつも使っている apt-get / aptutide / dselect で torrent ネットワークから Debian Package を取得することが可能です。

しかし、実はどのようなパッケージが torrent 上にあるのか分りません。ドキュメントを読む限り先の URI のサイトで公開されている apt 用のレポジトリ `debhttp://sianka.free.fr/debianunstablemain` にあるものが公開されているようです。

apt のフロントエンドである aptitude を使い、インストールしたときの例を以下に示します。

```
# aptitude install frozen-bubble
Reading Package Lists... Done
Building Dependency Tree
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
The following NEW packages will be automatically installed:
 fb-music-high frozen-bubble-data
The following packages have been kept back:
 galeon galeon-common
The following NEW packages will be installed:
 fb-music-high frozen-bubble frozen-bubble-data
0 packages upgraded, 3 newly installed, 0 to remove and 2 not upgraded.
Need to get 12.1MB/12.2MB of archives. After unpacking 17.3MB will be used.
Do you want to continue? [Y/n/?]
Writing extended state information... Done
Get:1 http://127.0.0.1 unstable/main fb-music-high 0.1.1 [6950kB]
Get:2 http://127.0.0.1 unstable/main frozen-bubble-data 1.0.0-6 [5155kB]
Fetched 12.1MB in 24s (488kB/s)
Selecting previously deselected package fb-music-high.
(Reading database ... 208931 files and directories currently installed.)
Unpacking fb-music-high (from .../fb-music-high_0.1.1_all.deb) ...
Selecting previously deselected package frozen-bubble-data.
Unpacking frozen-bubble-data (from .../frozen-bubble-data_1.0.0-6_all.deb) ...
Selecting previously deselected package frozen-bubble.
Unpacking frozen-bubble (from .../frozen-bubble_1.0.0-6_i386.deb) ...
Setting up fb-music-high (0.1.1) ...
Setting up frozen-bubble-data (1.0.0-6) ...
Setting up frozen-bubble (1.0.0-6) ...

Reading Package Lists... Done
Building Dependency Tree
Reading extended state information
Initializing package states... Done
Reading task descriptions... Done
```

2007/01/18 現在では正常にパッケージが取得できません。[*11]

5.3 仕組み

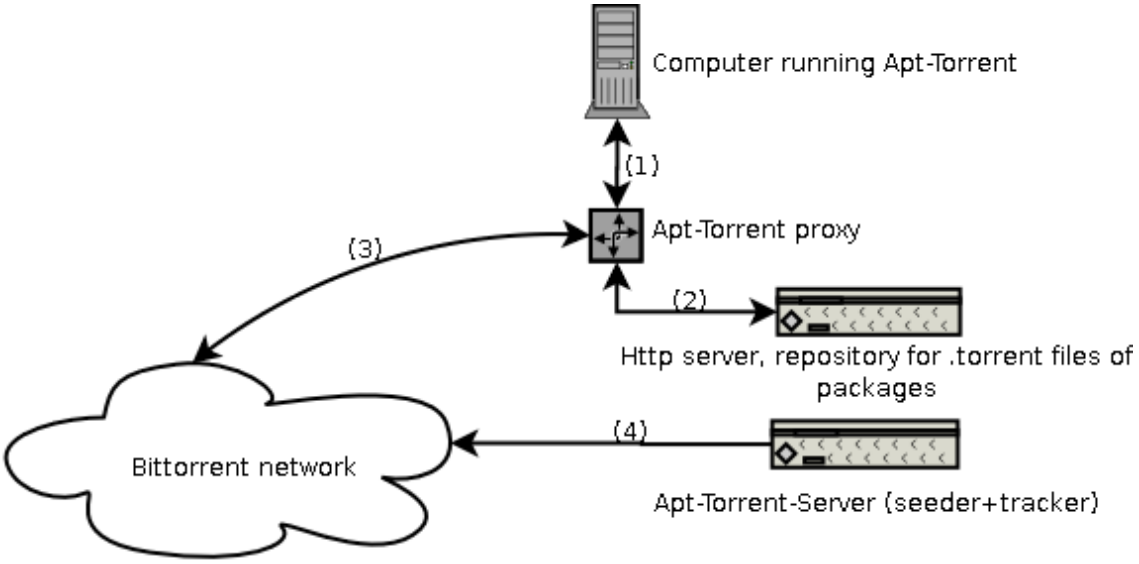
`apt-torrent` は torrent ネットワーク用のデーモンと、パッケージを取得するデーモンの二種類があります。プロセスを見ると、`apt-torrent` のプロセスがあることがわかります[*12]。

```
23404 ?        Ssl    0:00 /usr/bin/pike7.6 /usr/bin/apt-torrent
23411 ?        S      0:00 /usr/bin/pike7.6 /usr/bin/apt-torrent-httpd
23412 ?        S      0:00 /usr/bin/python /usr/bin/btlaunchmany /var/cache/apt-torrent --max_upload_rate -1 --parse_dir_interval 7
23644 pts/0    R+     0:00 ps ax
```

`apt-torrnet` は ファイルを取得するデーモン、`apt-torrent-httpd` はデータを管理する httpd server です。これら 2 つのデーモンが通信を行い、 `bittorrent` 網から Debian Package を取得します。

[*11] 作者に連絡取り中

[*12] btlaunchmany は複数の seed 管理用デーモン



簡単な流れは以下のようになります。

1. クライアントで `apt-get update` を行います。
   行うと、`apt-torrent-proxy` を介して、通信を行います。
2. `apt-torrent-proxy` を介して `apt-torrent` 用の http server と通信します。
   通信を行い、`/etc/apt/sources.list` に指定してある apt-line のサーバーから情報を取得を試みます。
3. bittorrent 網 にアクセスします。
4. `apt-torrent` server から一覧を取得します。
   `apt-torrent` は seeder[*13] と tracker[*14]を行っている `apt-torrent` にアクセスします。
5. `apt-get install xxxxxxxx` を行います `apt-torrent-proxy` を介して、seeder からデータの取得を試みます。取得したデータは Debian Package なので apt が後は勝手に処理をします。

`apt-torrent` 用 http server を介して bittorrent 網にアクセスしているので、http で公開されている apt-line からパッケージを取得しているのと変わらない動きをします。

5.4 apt との比較

今では、apt-get 時に http / ftp サーバーに負荷が集中していますが、`apt-torrent` を使用することによってひとつのパッケージを ピア同士で共有することが可能になります。`apt-torrnet` は http / ftp サーバーに負荷がかからなくなるひとつの方法になるのではないかと個人的に考えています。変化が激しい unstable / testing は無理としても、stable のパッケージを `apt-torrent` で頒布するのはいい方法だと思います。今後、開発者と連絡を取り合い、自宅でも `apt-torrent` server を立てて、実験してみようと模索しているところです。

5.5 その他

同じような機能を持った Winny のプロトコルを使い、`apt-winny` を実装してみると面白いと思いました。2ch で winny の Linux 版である Linny を開発しているようなので（コードはまだない。）期待したいと思います。

[*13] 対象ファイルのデータを全て持っているピア
[*14] bittorrent ファイルを管理するサーバー

6 Debian 勉強会 2007 年度計画

上川純一

Debian が今提供している付加価値は「＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿」です。2007 年に発生しそうな外部的なイベントは次の項目です

- Windows：（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
- Mac OS X：（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
- 他のディストリビューション：（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
- ハードウェア：（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
- ユーザの期待：（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
- （＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）

この状況を踏まえて Debian 勉強会が成し遂げるべきことは
「＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿」です。
これらを踏まえて 2007 年の Debian 勉強会のテーマは
「＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿」
とします。ここで、2007 年の 12ヶ月分のアジェンダを作成します。

1. 新年会（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
2. （＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
3. OSC?（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
4. OSC-Do?（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
5. Debconf プレゼンリハーサル（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
6. Debconf エジンバラ開催（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
7. Debconf 参加報告会（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
8. Debian 14 周年（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
9. （＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
10. OSC-Fall?（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
11. KOF?（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）
12. 忘年会（＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿）

この中で自分が主体として開催するのは「＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿」の回の内容です。

名前のブランディングとして、通常実施の勉強会を「＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿」と呼び、大衆向けの勉強会を「＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿」と呼びます。

7 パッチ管理ツール quilt の使い方

小林さん

本節ではパッチ管理ツール quilt[*15]を取り上げ、その基本操作を中心に説明します。quilt は Debian に特有な (つまり Debian-specific な) ソフトウェアではなく、Debian とは全く関係のない場所で開発されています。しかし、このツールが Debian パッケージの作成に有用であることを理解し、実際に作成に使用してもらえたら嬉しい限りです。

7.1 はじめに

7.1.1 パッチとは

開発者であれば当然のように作成したり読んだり当てたりしたことがある**パッチ**ですが、一般ユーザには見慣れないものだと思います。そこで、最初にパッチについて説明しておきます。

パッチとは、2 つのファイルの差分を含むファイルのことです。次のような 2 つのファイルがあるとしましょう。

```
nori1[3:57]%  cat file1.txt                                    saba:~/tmp/q
あいうえお
かきくけこ
さしすせそ
たちつてと
なにぬねの
はひふへほ
まみむめも
や　ゆ　よ
らりるれろ
わ　を　ん
nori1[3:58]%  cat file2.txt                                    saba:~/tmp/q
あいうえお
ばびぶべぼ
たちつてと
なにぬねの
まみむめも
や　ゆ　よ
らりるれろ
わ　を　ん
```

これらのファイルのどこが違っているかすぐにわかりますか? 一々目で比べる必要はありません。`diff` コマンドでこれらの差分をとることができます。

```
nori1[3:57]%  diff file1.txt file2.txt                         saba:~/tmp/q
2,3c2
< かきくけこ
< さしすせそ
---
> ばびぶべぼ
6d4
< はひふへほ
```

`file1.txt` と `file2.txt` の内容や行番号を考えれば、なんとなく意味がわかると思います。それでかまいません。もちろん、この出力を手で書けるようにならなくてかまいません。

この形式でもよいのですが、通常は `diff` コマンドに`-u` オプションをつけ、次のような **unified diff** という形式で出力させます。

[*15] `http://savannah.nongnu.org/projects/quilt`

```
nori1[3:57]%  diff -u file1.txt file2.txt                        saba:~/tmp/q
--- file1.txt   Thu Jan 18 03:57:14 2007
+++ file2.txt   Thu Jan 18 03:57:22 2007
@@ -1,9 +1,7 @@
 あいうえお
-かきくけこ
-さしすせそ
+ばびぶべぼ
 たちつてと
 なにぬねの
-はひふへほ
 まみむめも
 や ゆ よ
 らりるれろ
```

こちらのほうが、追加・削除された行がどちらのファイルに所属するのか、そしてファイルのどのような部分が異なるのかがわかりやすいでしょう。

パッチとは要は、次のようにしてこういった出力をファイルに収めたものです。

```
nori1[3:58]%  diff -u file1.txt file2.txt > file.diff            saba:~/tmp/q
```

ただし、全く縁のない 2 つのファイルであれば差分はわざわざファイルに収める必要はないでしょう。わざわざパッチを作成するのは、ファイルに変更を加える前後の差分を見たり、後で紹介するように機械的な処理で変更を加えられるようにしたりするためです。そこで、ここまでは file1.txt と file2.txt は無縁のファイルとしてきましたが、ここからは file1.txt を改変したものが file2.txt であると考えてください。つまり file1.txt が改変前のファイルで、file2.txt が改変後のファイル、パッチ file.diff には file1.txt を file2.txt にするための変更が含まれている、と考えてください。

さて、作成したパッチは、patch コマンドを用いて、その中に含む変更を該当ファイルに**適用する** (**当てる**) ことができます。

```
nori1[4:00]%  patch < file.diff                                  saba:~/tmp/q
patching file file1.txt
```

こうして file1.txt の内容は file2.txt の内容と同一になります。

```
nori1[4:01]%  cat file1.txt                                      saba:~/tmp/q
あいうえお
ばびぶべぼ
たちつてと
なにぬねの
まみむめも
や ゆ よ
らりるれろ
わ を ん
nori1[4:02]%  cmp file1.txt file2.txt                            saba:~/tmp/q
```

また、-R オプションを指定して逆向きに当てることもできます。

```
nori1[4:18]%  patch -R < file.diff                               saba:~/tmp/q
patching file file1.txt
nori1[4:19]%  cat file1.txt                                      saba:~/tmp/q
あいうえお
かきくけこ
さしすせそ
たちつてと
なにぬねの
はひふへほ
まみむめも
や ゆ よ
らりるれろ
わ を ん
```

file1.txt の内容は元に戻りました。

パッチは必ずしもうまく当たらないことを忘れてはなりません。次のように、file1.txt の 2 行目を手元で改変したとします。

```
nori1[5:29]%  cat file1.txt                                      saba:~/tmp/q
あいうえお
がぎぐげご
さしすせそ
たちつてと
なにぬねの
はひふへほ
まみむめも
や ゆ よ
らりるれろ
わ を ん
```

これに先程のパッチを当てようとすると次のようにエラーになります。

```
nori1[5:30]%  patch < file.diff                                    saba:~/tmp/q
patching file file1.txt
Hunk #1 FAILED at 1.
1 out of 1 hunk FAILED -- saving rejects to file file1.txt.rej
```

これは、手元で与えた変更とパッチが加えたい変更が競合しているからです。

もちろん、file1.txt の 2 行目を改変する代わりに、次のように最終行に 1 行加えたとします。

```
nori1[5:32]%  cat file1.txt                                        saba:~/tmp/q
あいうえお
かきくけこ
さしすせそ
たちつてと
なにぬねの
はひふへほ
まみむめも
や ゆ よ
らりるれろ
わ を ん
むふふふふ
```

この場合、手元で加えた変更はパッチが含む変更とは被らないので、パッチはうまく当たります[*16]。

```
nori1[5:36]%  patch < file.diff                                    saba:~/tmp/q
patching file file1.txt
nori1[5:52]%  cat file1.txt                                        saba:~/tmp/q
あいうえお
ばびぶべぼ
たちつてと
なにぬねの
まみむめも
や ゆ よ
らりるれろ
わ を ん
むふふふふ
```

これまでは 1 つのファイルの変更を 1 つのパッチに収めてきましたが、1 つのパッチに複数のファイルの変更を収めることも可能です。以下は、diff に-r オプションを渡して、内容の一部が異なる 2 つのファイルを含む 2 つのディレクトリの差分をパッチとして出力させた例です。

```
nori1[5:14]%  diff -ur skkdic.orig skkdic > skkdic.diff              saba:~/tmp
nori1[5:14]%  cat skkdic.diff                                       saba:~/tmp
diff -ur skkdic.orig/debian/changelog skkdic/debian/changelog
--- skkdic.orig/debian/changelog        Fri Dec 15 03:36:49 2006
+++ skkdic/debian/changelog     Mon Dec 25 17:39:10 2006
@@ -1,5 +1,7 @@
 skkdic (20061130-2~pre1) unstable; urgency=low

+  * debian/rules: Pass the '-k' option to dh_installchangelogs so that
+    all of ChangeLog* are installed equally.
   * debian/skkdic{,-cdb,-extra}.docs: Deleted since their settings are
     now moved into debian/rules.

diff -ur skkdic.orig/debian/rules skkdic/debian/rules
--- skkdic.orig/debian/rules    Fri Dec 15 03:36:49 2006
+++ skkdic/debian/rules Mon Dec 25 17:39:10 2006
@@ -2,6 +2,7 @@
 include /usr/share/cdbs/1/rules/debhelper.mk
 include /usr/share/cdbs/1/class/makefile.mk

+DEB_DH_INSTALLCHANGELOGS_ARGS := -k
 DEB_INSTALL_CHANGELOGS_ALL := ChangeLog
 DEB_INSTALL_DOCS_ALL := ChangeLog.* README*s/committers.txt
```

以上で簡単なパッチの説明は終わりです。こういったパッチは、修正した部分を確認したり他人と修正内容をやりとりしたりするのに非常に便利で、開発作業はこれなしではやっていけないと言っても過言ではないでしょう。実際に使うに当たっては patch(1) や diff(1) を参照することをお勧めします。

7.1.2 パッチ管理ツールとは

今回説明する quilt はパッチ管理ツールです。これは、複数のパッチを管理するためのものです。どのような場合に必要になるのでしょう?

先程パッチとは「修正した部分を確認したり他人と修正内容をやりとりしたりするのに非常に便利」だと書きました。その言葉に基づけば、一般にはパッチとは使い捨ての一時ファイルで短寿命です。管理する必要はありません。

しかし、以下のようなケースを考えてみてください。

[*16] ただしあくまで形式的にであって、意味的に適切かどうかは確認する必要があります。

- ある団体がソフトウェア foo を公開し、定期的にリリースしている。
- A さんは foo を改変して使ったり再配布したりする必要がある。
- A さんの改変内容には複数の論理的に異なる変更が混ざっており、foo に取り込まれるものもあれば取り込まれないものもある。
  - foo のバグの修正 1 (小さな変更なので次のマイナーリリースで適用される。)
  - foo のバグの修正 2 (大きめの変更なので次のメジャーリリースまで適用されない。)
  - foo のバグの修正 3 (バグを回避するための A さん独自の次善策で、開発元からのリリースでは適用されない。)
  - :
  - foo のバグの修正 N
  - A さんの環境に合わせるための Makefile の修正

A さんが全体の変更を 1 つのパッチとして管理するのは不可能に近いでしょう。foo の新しいリリースが出たときに、元のパッチを当てても (うまく当たる部分もあるでしょうが) 間違いなくうまく当たりません。しかも複数の変更が混ざっているのでそれを解決するのは非常に大変でしょう。

変更を複数の論理的なパッチに分割すれば、少しは管理が楽になるでしょう。1 つずつパッチを当て、うまく当たるものと当たらないものを区別できます。当たらないもののうち、既に開発元で加えられた修正のパッチは削除し、残すべきなのに他の変更の影響で部分的にうまく当たらないパッチはうまく当たるよう調整します。それらの作業は面倒かもしれませんが、1 つのパッチとして管理するのに比べれば楽でしょう。ただし、作業中に、現在どのパッチが当たっていてどれが当たっていないかを一々覚えておかなければならないのが大変です。

そこで登場するのがパッチ管理ツールです。パッチ管理ツールは、複数のパッチの取り扱いを楽にするためのツールです。具体的には、どのパッチが当たっておりどれが当たっていないかを管理できます。また、一度に複数のパッチを当てたり外したりできるので、全ての (あるいはある一群の) パッチがうまく当たるかどうかを概観するのも楽にできます。A さんがやっているような作業にはうってつけのツールです。

さて、実は Debian パッケージのメンテナ (保守担当者) はこの A さんのような作業をする必要があります。すなわち、開発元のリリースに複数のパッチを独自に当て、それらを管理する必要があるのです。というのも、一般に開発元と Debian とではリリースサイクルや品質の基準が異なるからです。管理すべきパッチは、ソフトウェアの規模が大きくなればなるほど増えるでしょう。そこで Debian パッケージメンテナはパッチ管理ツールを使用することが推奨されています。

ここでは quilt を使用したパッチ管理を紹介します。Debian でよく使われるパッチ管理ツールには、dpatch というものもあります。こちらは Debian パッケージメンテナンスに特化したツールですが、機能的には quilt とそんなに違いはありません。興味がある方は、2005 年 7 月勉強会[*17]の資料を参照してください。

7.2 インストール

quilt は、Debian では `quilt` パッケージに含まれており、`apt-get` や `aptitude` で普通にインストールできます。

7.3 quilt の基本操作

7.3.1 最初に知っておくべきこと

quilt では一般に、`quilt <コマンド>`という書式で様々なコマンドが利用できます。利用可能なコマンドの一覧は次のようにして参照できます。

[*17] `http://www.netfort.gr.jp/~dancer/column/2005-debianmeeting.html.ja#6th`

```
nori1[13:19]%  quilt --help                                              whale:~
使い方: quilt [--trace[=verbose]] [--quiltrc=XX] command [-h] ...
       quilt --version
コマンド一覧:
        add       files   import   previous  setup
        annotate  fold    mail     push      snapshot
        applied   fork    new      refresh   top
        delete    graph   next     remove    unapplied
        diff      grep    patches  rename    upgrade
        edit      header  pop      series

全コマンド共通オプション:

--trace
        コマンドを bash のトレースモード（-x）で実行。内部デバッグ用。

--quiltrc file
        ~/.quiltrc（存在しない場合は代わりに /etc/quiltrc）以外のコン
        フィギュレーションファイルを指定。内容の詳細については PDF のド
        キュメントを参照。

--version
        バージョン情報を出力して終了。
```

また各コマンドのヘルプメッセージは各コマンドに-h オプションを指定すれば表示できます。以下の例では quilt add のヘルプメッセージを表示させています。

```
nori1[13:45]%  quilt add -h                                              whale:~
Usage: quilt add [-P patch] {file} ...

Add one or more files to the topmost or named patch.  Files must be
added to the patch before being modified.  Files that are modified by
patches already applied on top of the specified patch cannot be added.

-P patch
        Patch to add files to.
```

7.3.2 パッチスタック

実際に quilt を使う前に、quilt のデータ構造を知っておきましょう。quilt は、あるツリーに対する一連のパッチを一列に並べてスタックとして管理します。

あるソースツリーに、以下のような 5 つのパッチを順に加えたいとします。

1. r1091-remove-trailing-garbage.diff
2. r1092-implement-distclean.diff
3. r1094-add-readme.diff
4. fix-ftbfs-on-m68k.diff
5. work-around-an-error-of-libtool.diff

このとき、これらのパッチは **patches ディレクトリ**に含まれていなければなりません。そして、パッチの整列順序を収めた次のような内容の **series ファイル**[*18]がパッチディレクトリに含まれている必要があります。

```
r1091-remove-trailing-garbage.diff
r1092-implement-distclean.diff
r1094-add-readme.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
```

つまりまとめると、上に挙げた 5 つのパッチを順に加えるようなパッチセットのデータは次のようなものです。

[*18] dpatch における 00list と同じ役割を果たします。

```
nori1[14:22]%  ls patches/*             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
patches/fix-ftbfs-on-m68k.diff
patches/r1091-remove-trailing-garbage.diff
patches/r1092-implement-distclean.diff
patches/r1094-add-readme.diff
patches/series
patches/work-around-an-error-of-libtool.diff
nori1[14:22]%  cat patches/series       whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
r1092-implement-distclean.diff
r1094-add-readme.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
```

quilt のパッチデータはこれが全てです。

なおこのデータ構造は、7.3.4 で後述するパッチ作成作業時に quilt が自動的に作成してくれるので、通常は手でいじる必要はありません。次のセクションでは、このような patches ディレクトリを既にもったツリーでパッチを当てたり外したりしてみましょう。

7.3.3 パッチスタックの操作

さて、では quilt を使ってみます。まずはスタック管理のための操作を眺めてみましょう。スタック管理に必要な操作は、もちろん push と pop です。

最初は何もパッチが当たっていないとします。この状態で `quilt push` を実行すると、1 番目のパッチ、つまり `r1091-remove-trailing-garbage.diff` が当たります。このパッチは `Makefile.in` に修正を加えるものです。

```
nori1[14:30]%  quilt push               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in

Now at patch r1091-remove-trailing-garbage.diff
```

きちんと当たったようですね。`quilt push` は、引数を与えなければ「次のパッチ」を当てるコマンドです。これで現在ツリーは、`r1091-remove-trailing-garbage.diff` だけが当たった状態になりました。

もう一度 `quilt push` を実行します。

```
nori1[15:08]%  quilt push               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1092-implement-distclean.diff
patching file Makefile.in

Now at patch r1092-implement-distclean.diff
```

次のパッチ `r1092-implement-distclean.diff` も当たりました。これを「`r1092-implement-distclean.diff` が**一番上に来た**状態」と言うことにしましょう。スタック (stack) とは英語で「積み重ねたもの」の意味です。push 操作ではその上にもの (quilt ではパッチ) を次々に載せていくので、「パッチ○○まで当てられた状態」は、言い換えれば「パッチ○○が一番上に来た状態」です。

push の逆は pop です。すなわち現在一番上にあるパッチを外すのは `quilt pop` です。

```
nori1[15:15]%  quilt pop                whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch r1092-implement-distclean.diff
Restoring Makefile.in

Now at patch r1091-remove-trailing-garbage.diff
nori1[15:15]%  quilt pop                whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
```

`quilt push` や `quilt pop` に引数としてパッチ名を与えると、そのパッチが一番上に来た状態になるように一連のパッチを当てたり外したりします。また引数として数値を与えると、その数だけパッチを当てたり外したりできます。`-a` をオプションとして指定すると全てのパッチを当てたり外したりできます。

```
nori1[15:17]%  quilt push 3             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in

Applying patch r1092-implement-distclean.diff
patching file Makefile.in

Applying patch r1094-add-readme.diff
patching file README

Now at patch r1094-add-readme.diff
nori1[15:17]%  quilt pop r1091-remove-trailing-garbage.diff
Removing patch r1094-add-readme.diff
Removing README

Removing patch r1092-implement-distclean.diff
Restoring Makefile.in

Now at patch r1091-remove-trailing-garbage.diff
nori1[15:18]%  quilt push -a            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1092-implement-distclean.diff
patching file Makefile.in

Applying patch r1094-add-readme.diff
patching file README

Applying patch fix-ftbfs-on-m68k.diff
patching file Makefile.in

Applying patch work-around-an-error-of-libtool.diff
patching file Makefile.in

Now at patch work-around-an-error-of-libtool.diff
nori1[15:18]%  quilt pop -a             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch work-around-an-error-of-libtool.diff
Restoring Makefile.in

Removing patch fix-ftbfs-on-m68k.diff
Restoring Makefile.in

Removing patch r1094-add-readme.diff
Removing README

Removing patch r1092-implement-distclean.diff
Restoring Makefile.in

Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
```

現在一番上にあるパッチは quilt top で確認できます。現在一番上にあるパッチの次のパッチと前のパッチはそれぞれ、quilt next と quilt previous で確認できます。現在当てられているパッチと当てられていないパッチはそれぞれ、quilt applied と quilt unapplied で確認できます。そして、全てのパッチ、つまり series ファイルの内容は quilt series で確認できます。

```
nori1[15:31]%  quilt top                whale:~/svnwc/deb/serf/trunk/serf-0.1.0
No patches applied
nori1[15:31]%  quilt next               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
nori1[15:31]%  quilt previous           whale:~/svnwc/deb/serf/trunk/serf-0.1.0
No patches applied
nori1[15:31]%  quilt applied            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
No patches applied
nori1[15:31]%  quilt unapplied          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
r1092-implement-distclean.diff
r1094-add-readme.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
nori1[15:32]%  quilt push 2             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in

Applying patch r1092-implement-distclean.diff
patching file Makefile.in

Now at patch r1092-implement-distclean.diff
nori1[15:32]%  quilt top                whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1092-implement-distclean.diff
nori1[15:32]%  quilt next               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1094-add-readme.diff
nori1[15:32]%  quilt previous           whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
nori1[15:32]%  quilt applied            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
r1092-implement-distclean.diff
nori1[15:33]%  quilt unapplied          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1094-add-readme.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
nori1[15:33]%  quilt series             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
r1092-implement-distclean.diff
r1094-add-readme.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
nori1[15:33]%  quilt pop -a             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch r1092-implement-distclean.diff
Restoring Makefile.in

Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
```

ある特定のファイルを変更するパッチの一覧は `quilt files` で参照可能です。逆に、ある特定のパッチが変更するファイルの一覧は `quilt patches` で参照可能です。

```
nori1[15:35]%  quilt files r1091-remove-trailing-garbage.diff
Patch r1091-remove-trailing-garbage.diff is not applied
nori1[15:35]%  quilt push r1091-remove-trailing-garbage.diff
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in

Now at patch r1091-remove-trailing-garbage.diff
nori1[15:35]%  quilt files              whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Makefile.in
nori1[15:36]%  quilt files r1091-remove-trailing-garbage.diff
Makefile.in
nori1[15:36]%  quilt pop -a             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
nori1[15:36]%  quilt patches Makefile.in
r1091-remove-trailing-garbage.diff
r1092-implement-distclean.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
```

これでパッチスタックの操作の説明は終わりです。パッチを含んだ patches ディレクトリが存在していれば、その中のパッチを自由自在に当てたり外したり、現在当たっているパッチを調べたりできるようになりました。しかしこれだけでは何の役にも立ちません。次のセクションではパッチを新たに追加する方法について説明します。

7.3.4 パッチの追加

いよいよ patches ディレクトリに新たなパッチを追加します。quilt では `quilt new` で新たなパッチに名前をつけて作成を開始し、適当な変更を加えた後、`quilt refresh` でパッチとして保存します。

新たなパッチを追加する場合、まずはパッチをスタックのどこに入れるか決めます[*19]。今回は (特に意味はあり

[*19] 7.4 にパッチの並べ方に関する簡単なアドバイスがあります。

ませんが) 1 つ目のパッチである r1091-remove-trailing-garbage.diff の後に入れましょう。パッチの名前は love-debian.diff とし、その内容は、次のような love-debian ターゲットを Makefile.in に加えるものにしましょう。

```
love-debian:
        echo "I love debian!!"
```

では作成を開始します。

```
nori1[17:29]%  quilt push r1091-remove-trailing-garbage.diff
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in

Now at patch r1091-remove-trailing-garbage.diff
nori1[17:29]%  quilt new love-debian.diff
Patch love-debian.diff is now on top
```

早速 Makefile.in を編集したいところですが、少し待ってください。実は quilt では、パッチに含めるファイルは予め quilt add で追加しておかなければなりません。

```
nori1[17:29]%  quilt add Makefile.in    whale:~/svnwc/deb/serf/trunk/serf-0.1.0
File Makefile.in added to patch love-debian.diff
```

その上でファイルの編集をし、その内容を quilt diff で確認します。

```
nori1[17:35]%  vim Makefile.in          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
[snip]
nori1[17:38]%  quilt diff               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Index: serf-0.1.0/Makefile.in
===================================================================
--- serf-0.1.0.orig/Makefile.in        2007-01-18 17:35:11.000000000 +0900
+++ serf-0.1.0/Makefile.in        2007-01-18 17:38:38.000000000 +0900
@@ -99,6 +99,9 @@
                $(INSTALL) -m 644 $$i $(DESTDIR)$(includedir); \
         done

+love-debian:
+        echo "I love debian!!"
+
 clean:
         rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)
```

最後に quilt refresh で保存します。

```
nori1[17:39]%  quilt refresh            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Refreshed patch love-debian.diff
```

さて、実は love-debian.diff は 1 番目のパッチの後に挿入しました。ということは、残りのパッチがうまく当たるか確認しておかなければなりません。

```
nori1[17:40]%  quilt push -a            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1092-implement-distclean.diff
patching file Makefile.in
Hunk \#1 succeeded at 104 (offset 3 lines).

Applying patch r1094-add-readme.diff
patching file README

Applying patch fix-ftbfs-on-m68k.diff
patching file Makefile.in

Applying patch work-around-an-error-of-libtool.diff
patching file Makefile.in

Now at patch work-around-an-error-of-libtool.diff
```

新たな行を挿入したので「Hunk #1 succeeded at 104 (offset 3 lines).」というメッセージが出ましたが、問題なく当たりました。

quilt series を実行すると、きちんと love-debian.diff が適切な位置に挿入されていることがわかります。

```
nori1[17:41]%  quilt series             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
love-debian.diff
r1092-implement-distclean.diff
r1094-add-readme.diff
fix-ftbfs-on-m68k.diff
work-around-an-error-of-libtool.diff
nori1[17:41]%  quilt pop -a             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch work-around-an-error-of-libtool.diff
Restoring Makefile.in

Removing patch fix-ftbfs-on-m68k.diff
Restoring Makefile.in

Removing patch r1094-add-readme.diff
Removing README

Removing patch r1092-implement-distclean.diff
Restoring Makefile.in

Removing patch love-debian.diff
Restoring Makefile.in

Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
```

さて、パッチを作成した後になって気付いたのですが、love-debian ターゲットが表示する文字列の中の「debian」は「Debian」と大文字にした方がよさそうです。このようなときはパッチを改変しましょう。quilt push ないし quilt pop で目的のパッチ (love-debian.diff) を一番上に持ってきて、エディタで Makefile.in に変更を施した上で quilt refresh で保存すれば OK です。Makefile.in は既にパッチ love-debian.diff の変更対象に含まれているので、エディタでの編集前に quilt add で追加する必要はありません。

```
nori1[17:54]%  quilt push love-debian.diff
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in

Applying patch love-debian.diff
patching file Makefile.in

Now at patch love-debian.diff
nori1[17:59]%  vim Makefile.in          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
[snip]
nori1[18:01]%  quilt diff               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Index: serf-0.1.0/Makefile.in
===================================================================
--- serf-0.1.0.orig/Makefile.in        2007-01-18 17:38:38.000000000 +0900
+++ serf-0.1.0/Makefile.in        2007-01-18 18:01:35.000000000 +0900
@@ -99,6 +99,9 @@
                  $(INSTALL) -m 644 $$i $(DESTDIR)$(includedir); \
         done

+love-debian:
+        echo "I love Debian!!"
+
 clean:
         rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)

nori1[18:01]%  quilt refresh            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Refreshed patch love-debian.diff
```

さて、このパッチを眺めているうちにまた気が変わって、design-guide.txt という別のファイルにも変更を加え、このパッチに含めようと思い立ちました。design-guide.txt は今のところ love-debian.diff の変更対象外なので、今度は、編集する前に quilt add で変更対象に追加しておかなければなりません。quilt add design-guide.txt を実行した上でエディタで design-guide.txt を開いてもよいのですが、面倒なのでここでは quilt edit コマンドを実行しましょう。このコマンドは、「引数のファイルを quilt add で追加した上で、環境変数 EDITOR で指定されたエディタでそのファイルを開く」という、2 つの操作をまとめて実行するためのものです。

```
nori1[18:19]%  quilt edit design-guide.txt
[snip]
nori1[18:21]%  quilt diff               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Index: serf-0.1.0/Makefile.in
===================================================================
--- serf-0.1.0.orig/Makefile.in        2007-01-18 17:55:20.000000000 +0900
+++ serf-0.1.0/Makefile.in        2007-01-18 18:01:35.000000000 +0900
@@ -99,6 +99,9 @@
                 $(INSTALL) -m 644 $$i $(DESTDIR)$(includedir); \
         done

+love-debian:
+        echo "I love Debian!!"
+
 clean:
         rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)

Index: serf-0.1.0/design-guide.txt
===================================================================
--- serf-0.1.0.orig/design-guide.txt        2007-01-18 18:19:30.000000000 +0900
+++ serf-0.1.0/design-guide.txt        2007-01-18 18:20:40.000000000 +0900
@@ -9,6 +9,7 @@
   4. Bucket Read Functions
   5. Versioning
   6. Bucket lifetimes
+  7. Love Debian


 ------------------------------------------------------------------------------
@@ -150,3 +151,10 @@


 ------------------------------------------------------------------------------
+
+7. LOVE DEBIAN
+
+Hey, please love Debian.
+
+
+------------------------------------------------------------------------------
nori1[18:22]%  quilt refresh            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Refreshed patch love-debian.diff
```

これで新たなパッチ `love-debian.diff` の内容は完成です。しかしこれだけだと後で何をしたいパッチだかわからなくなりそうです。最後に、パッチの説明を加えておきましょう。quilt では、パッチの内容を説明するヘッダを操作するコマンドは `quilt header` です。`-e` オプションを指定してコマンドを実行すると、環境変数 EDITOR のエディタでパッチのヘッダを編集できます。また、何もオプションを与えずに実行するとヘッダを表示できます。

```
nori1[18:37]%  quilt header             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
nori1[18:37]%  quilt header -e          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
[snip]
nori1[18:41]%  quilt header             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
A patch to show my love for Debian.
```

本セクションでは、新たにパッチを作成する手順を学びました。これで自由にパッチを追加・編集できます。次のセクションでは、開発元が加えた変更への対応方法を学びます。

7.3.5 開発元が加えた変更への対応

パッチをいじれるようになったところで、開発元が新しいリリースなどで加えた変更に対応してみましょう。この作業では、`quilt push` の`-f` オプションが役立ちます。

開発元が加えた変更をマージする前に、まずやっておかなければならないことがあります。それは、ツリーにパッチが適用されていない状態にすることです。既に説明したように、適用されているパッチを全て外すには `quilt pop -a` を実行します。

```
nori1[20:00]%  quilt pop -a             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch love-debian.diff
Restoring Makefile.in
Restoring design-guide.txt

Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
```

その上で、ツリーのファイルを、開発元が新しくリリースしたものに差し替えます (つまり開発元が加えた変更をマージします)。ここでは以下のような変更が入ったとします。

- `Makefile.in` への love ターゲットと clean-love ターゲットの追加 (`love-debian.diff` が変更する部分と同

じ部分を改変するので、競合)

- README ファイルの追加 (r1094-add-readme.diff の修正内容そのもの)

さあ、新しいツリーにパッチを当てようとするとどうなるでしょうか。とりあえず quilt push -a で一度に全て当てようとしてみましょう。

```
nori1[20:32]%  quilt push -a            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in
Hunk #1 succeeded at 103 with fuzz 2 (offset 3 lines).

Applying patch love-debian.diff
patching file Makefile.in
Hunk #1 FAILED at 99.
1 out of 1 hunk FAILED -- rejects in file Makefile.in
patching file design-guide.txt
Patch love-debian.diff does not apply (enforce with -f)
nori1[20:32]%  quilt top                whale:~/svnwc/deb/serf/trunk/serf-0.1.0
r1091-remove-trailing-garbage.diff
```

love-debian.diff を当てるのに失敗して、そこでパッチの適用が止まってしまいました。そこで、-f オプションを quilt push につけて、love-debian.diff を強制的に適用してみます。

```
nori1[20:33]%  quilt push -f            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch love-debian.diff
patching file Makefile.in
Hunk #1 FAILED at 99.
1 out of 1 hunk FAILED -- saving rejects to file Makefile.in.rej
patching file design-guide.txt
Applied patch love-debian.diff (forced; needs refresh)
```

Makefile.in がどうなったか眺めてみると、先程 love-debian ターゲットを加えた箇所に、次のように love ターゲットと clean-love ターゲットが追加されています。

```
nori1[20:37]%  cat Makefile.in          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
[snip]
                $(INSTALL) -m 644 $$i $(DESTDIR)$(includedir); \
        done
love:
        echo "Making love..."
clean-love:
        echo "Clearing my old love..."
clean:
        rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)
[snip]
```

quilt push -f からのメッセージにあるように、パッチ適用の拒否の理由は Makefile.in.rej に記されています。念のため覗いてみましょう。

```
***************
*** 99,104 ****
                $(INSTALL) -m 644 $$i $(DESTDIR)$(includedir); \
          done

  clean:
          rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)

--- 99,107 ----
                $(INSTALL) -m 644 $$i $(DESTDIR)$(includedir); \
          done

+ love-debian:
+         echo "I love Debian!!"
+
  clean:
          rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)
```

これは **context diff** と呼ばれる形式ですが、意味が掴めればかまいません。上で説明したように、love-debian ターゲットを加えるための、clean ターゲットの前の 3 行がうまく当たらなかったことがわかるでしょう。

強制的にパッチを適用した結果がどうなったかわかったところで、削除されてしまった大切な love-debian ターゲットを Makefile.in に再度追加した上で、quilt push -f からのメッセージにあるようにパッチを更新 (refresh) しましょう。

```
nori1[20:45]%  vim Makefile.in          whale:~/svnwc/deb/serf/trunk/serf-0.1.0
[snip]
nori1[20:54]%  quilt diff               whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Index: serf-0.1.0/Makefile.in
===================================================================
--- serf-0.1.0.orig/Makefile.in        2007-01-18 20:32:13.000000000 +0900
+++ serf-0.1.0/Makefile.in        2007-01-18 20:52:22.000000000 +0900
@@ -102,6 +102,10 @@
         echo "Making love..."
 clean-love:
         echo "Clearing my old love..."
+
+love-debian:
+        echo "I love Debian!!"
+
 clean:
         rm -f $(TARGET_LIB) $(OBJECTS) $(OBJECTS:.lo=.o) $(PROGRAMS) $(TEST_OBJECTS) $(TEST_OBJECTS:.lo=.o)

Index: serf-0.1.0/design-guide.txt
===================================================================
--- serf-0.1.0.orig/design-guide.txt        2007-01-18 20:31:56.000000000 +0900
+++ serf-0.1.0/design-guide.txt        2007-01-18 20:36:10.000000000 +0900
@@ -9,6 +9,7 @@
   4. Bucket Read Functions
   5. Versioning
   6. Bucket lifetimes
+  7. Love Debian


 ----------------------------------------------------------------------------
@@ -150,3 +151,10 @@


 ----------------------------------------------------------------------------
+
+7. LOVE DEBIAN
+
+Hey, please love Debian.
+
+
+----------------------------------------------------------------------------
nori1[20:54]%  quilt refresh            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Refreshed patch love-debian.diff
```

さあ、これで love-debian.diff の問題は解決しました。再び quilt push -a で残りのパッチを全て当てようとしてみましょう。

```
nori1[20:55]%  quilt push -a            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1092-implement-distclean.diff
patching file Makefile.in
Hunk #1 succeeded at 108 (offset 7 lines).

Applying patch r1094-add-readme.diff
The next patch would create the file README,
which already exists!  Applying it anyway.
patching file README
Patch attempted to create file README, which already exists.
Hunk #1 FAILED at 1.
1 out of 1 hunk FAILED -- rejects in file README
Patch r1094-add-readme.diff can be reverse-applied
```

今度は r1094-add-readme.diff を当てるのに失敗して、そこでパッチの適用が止まってしまいました。メッセージによれば、このパッチは逆に当てることができます。それはすなわち、このパッチが既に当たった状態であることを意味します。開発元で README を追加してくれたので、README を追加するための r1094-add-readme.diff は不要になったのでした。そこで、このパッチは削除することにします。

```
nori1[21:05]%  quilt delete r1094-add-readme.diff
Removed patch r1094-add-readme.diff
```

これで r1094-add-readme.diff の問題も解決です。再び quilt push -a で残りのパッチを全て当てようとしてみましょう。

```
nori1[21:05]%  quilt push -a            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch fix-ftbfs-on-m68k.diff
patching file Makefile.in

Applying patch work-around-an-error-of-libtool.diff
patching file Makefile.in

Now at patch work-around-an-error-of-libtool.diff
```

全てうまく当たりました。これで一連のパッチはうまく当たるようになりました。

```
nori1[21:06]%  quilt pop -a             whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Removing patch work-around-an-error-of-libtool.diff
Restoring Makefile.in

Removing patch fix-ftbfs-on-m68k.diff
Restoring Makefile.in

Removing patch r1092-implement-distclean.diff
Restoring Makefile.in

Removing patch love-debian.diff
Restoring Makefile.in
Restoring design-guide.txt

Removing patch r1091-remove-trailing-garbage.diff
Restoring Makefile.in

No patches applied
```

7.3.6 パッチの依存関係の可視化

quilt の quilt graph コマンドと Graphviz を使うと、適用されたパッチの依存関係を可視化できます。

最初に、全てのパッチを当てておきます。

```
nori1[22:18]%  quilt push -a            whale:~/svnwc/deb/serf/trunk/serf-0.1.0
Applying patch r1091-remove-trailing-garbage.diff
patching file Makefile.in
Hunk #1 succeeded at 103 with fuzz 2 (offset 3 lines).

Applying patch love-debian.diff
patching file Makefile.in
patching file design-guide.txt

Applying patch r1092-implement-distclean.diff
patching file Makefile.in
Hunk #1 succeeded at 108 (offset 7 lines).

Applying patch fix-ftbfs-on-m68k.diff
patching file Makefile.in

Applying patch work-around-an-error-of-libtool.diff
patching file Makefile.in

Now at patch work-around-an-error-of-libtool.diff
```

この状態で何も与えずに quilt graph を実行すると、Graphviz 用のソースファイルが出力されます。これは、同じファイルを変更する複数のパッチの間に依存関係がある、として計算した結果です。

```
nori1[22:18]%  quilt graph              whale:~/svnwc/deb/serf/trunk/serf-0.1.0
digraph dependencies {
        n0 [label="r1091-remove-trailing-garbage.diff"];
        n1 [label="love-debian.diff"];
        n2 [label="r1092-implement-distclean.diff"];
        n3 [label="fix-ftbfs-on-m68k.diff"];
        n4 [style=bold,label="work-around-an-error-of-libtool.diff"];
        n0 -> n1 [len="1.39"];
        n1 -> n2 [len="1.39"];
        n2 -> n3 [len="1.39"];
        n3 -> n4 [len="1.39"];
}
```

このソースを例えば graph.dot というファイルに保存すると、Graphviz を使って (例えば dot -Tpng graph.dot -o graph.png などと実行して) 様々な形式のファイルが生成できます。しかし、ps を出力するのであれば次のように直接出力するのが楽でしょう。

```
nori1[22:35]%  quilt graph -T ps > patchdep-1.eps
```

なお、依存関係の指定はありませんが、内部的に (/usr/share/quilt/graph で) Graphviz の dot コマンドを呼び出しているので、実行には graphviz パッケージがインストールされている必要があります (Bug#407469[*20])。

生成された図は図 1 のようになります。

他方で、同じファイルを変更するだけでなくその変更領域が被っている場合に依存関係がある、として計算すると次のようになります。変更領域の被りが 1 行の場合 (図 2) と 2 行 (図 3) の場合です。

```
nori1[22:44]%  quilt graph --lines=1 -T ps > patchdep-2.eps
nori1[22:44]%  quilt graph --lines=2 -T ps > patchdep-3.eps
```

[*20] http://bugs.debian.org/407469

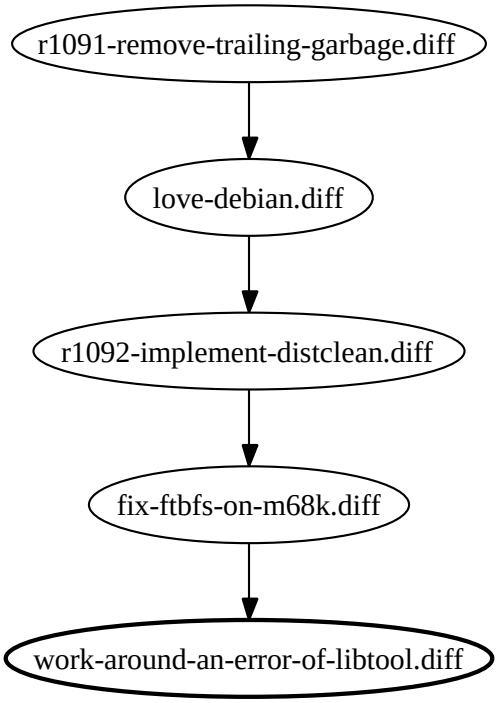


図 1　同じファイルを変更する複数のパッチの間に依存関係がある、として計算した依存関係

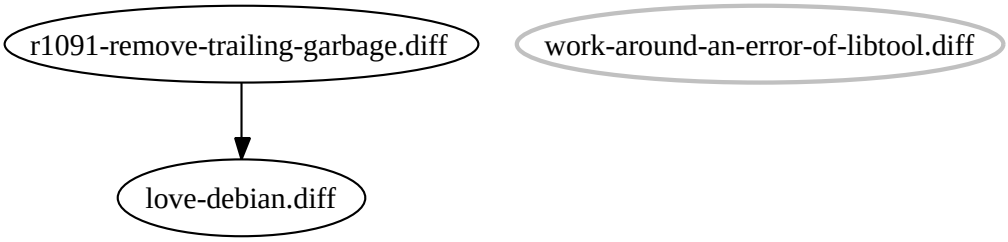


図 2　変更領域が 1 行被っている場合に依存関係がある、として計算した依存関係

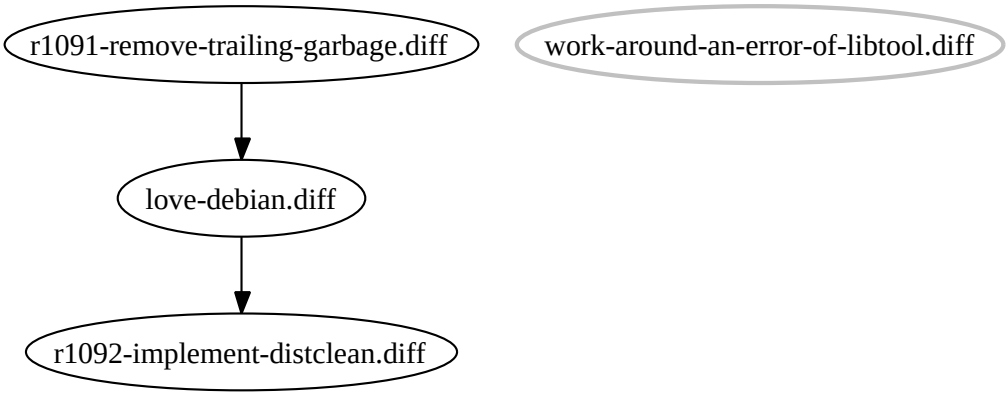


図 3　変更領域が 2 行被っている場合に依存関係がある、として計算した依存関係

7.3.7 環境変数

quilt では以下のような環境変数を指定できます。ここでは省略していますが、他にも diff 関連の環境変数がいくつかあります。

`QUILT_PATCHES` patches ディレクトリを指定します。指定されていない場合は `patches` がパッチファイル (および series ファイル) の格納に使用されます。

`QUILT_PC` .pc ディレクトリを指定します。.pc ディレクトリは、どのパッチが当てられたかを管理するためのもので、指定されていない場合は `.pc` となります。

7.4 Debian パッケージ作成時の quilt の使用

Debian パッケージ作成時のパッチの管理に quilt を使用するときは、以下を参考にしてください。

- quilt はデフォルトではパッチファイルを patches ディレクトリに入れます。しかし、Debian パッケージ作成用のパッチは、パッケージ作成に用いる他のファイルと同様 `debian` ディレクトリの下にまとめて入れておくことが推奨されています。7.3.7 で説明した環境変数 `QUILT_PATCHES` に `debian/patches` を設定しておくとよいでしょう。
- cdbs では quilt をパッチ管理に使用するのに便利なルールが用意されています。使用する場合は `debian/rules` 内で `/usr/share/cdbs/1/rules/patchsys-quilt.mk` を include してください。なお、cdbs のこのクラスは `patches` に `debian/patches` への symlink を作成します。
- やや一般的な話ですが、パッチを並べる順序は開発元に近い順にするとよいでしょう。つまり、開発元に既に取り込まれている変更が最初に適用され、開発元には取り込まれることのない Debian 独自の変更は最後に適用されるようにすることをお勧めします。

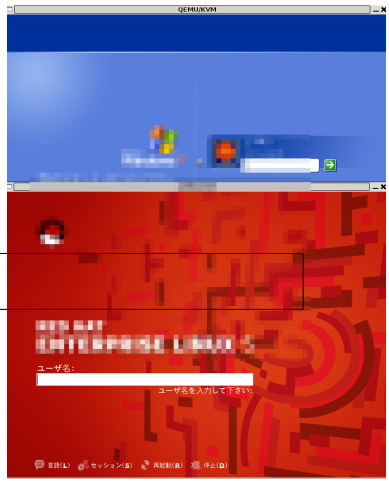
8 仮想マシンモニタ KVM

上川純一

KVM という仮想マシンモニタがあります。これは、 Intel VT, もしくは、AMD-V 対応のプロセッサ[*21] の仮想化対応機能を活用するための仕組です。2006 年末の時点では、kvm はデバイスドライバとして実装されており、`/dev/kvm` として実装されています。

8.1 使いかた

Linus Kernel 2.6.20 以降ではカーネル側の機構は標準で入っているようです。[*22] Debian で KVM を利用する際の手順を説明します。



`apt-get install kvm` でパッケージをインストールします。kvm コマンドがインストールされます。これが kvm 向けに変更された qemu です。

udev が作成してくれる `/dev/kvm` にアクセスできるようにします。デフォルトは `root:kvm 660`[*23] で、root のみがアクセスできる設定になっています。ユーザを kvm グループに追加すればよいでしょう。

```
# adduser dancer kvm
```

CPU の VT 機能はデフォルトで on になっていない場合があるので、 on にします。これはマシンによって違うようです。 通常 BIOS で設定します。 旧モデルの MacBook の場合は EFI 上で必要なコマンドを発行することになります。[*24]

以上で、kvm コマンドを利用して VT の恩恵にあずかることができるようになります。コマンドラインなどは qemu 互換、むしろ qemu そのものを利用しているので、qemu と同じように動かすことができます。

8.2 実行例

kvm を活用していろいろと試してみましょう。qemu 用のディスクイメージを作成し、適当なディストリビューションのディスクイメージを利用して、インストールします。おそらく、-monitor コマンドで stdio をモニタにするほうが便利でしょう。

[*21] Intel Core Duo や、Opteron Rev. F など
[*22] 2.6.20-rc1 で導入され、まだ 2.6.20 はリリースされていないためこの記事の内容は 1 月 19 日時点での予測です
[*23] バージョン 11-1 以降の場合
[*24] `vmx-var-set.efi` という名前で流通しているようです。

```
$ qemu-img create -f qcow hda 10G
$ kvm -hda hda -cdrom /home/iso/RHEL4-U4-i386-AS-disc1.iso
   -boot d -m 512 -monitor stdio
(qemu) change cdrom /home/iso/RHEL4-U4-i386-AS-disc2.iso
(qemu) change cdrom /home/iso/RHEL4-U4-i386-AS-disc3.iso
(qemu) change cdrom /home/iso/RHEL4-U4-i386-AS-disc4.iso
(qemu) change cdrom /home/iso/RHEL4-U4-i386-AS-disc5.iso
(qemu) change cdrom /home/iso/RHEL4-U4-i386-AS-disc1.iso
(qemu) eject cdrom
```

ここでは某 CDROM が 5 枚あるディストリビューションを例にとっていますが、CDROM が 5 枚あっても、qemu モニターにて CDROM を入れ換えながら GUI 操作を行うことで無事にインストール完了します。リブートするところでゲスト OS がカーネルパニックを起こしますが、そこは適当に kvm 自体を起動しなおせば問題ありません。

```
$ kvm -hda hda  -boot c -m 512 -monitor stdio -localtime  \
 -redir tcp:2222::22
```

8.3 ベンチマークしてみた

qemu を使った場合と kvm を使った場合の速度比較をしてみました。まず、ユーザ空間で完結する例として単純に while でループを回すだけのプログラムです。qemu の場合は 6s かかったものが、 kvm の場合は 0.6s で完了しました。

8.4 パフォーマンスの見えかた

kvm を実行しているホスト OS からどのように見えるのか確認してみましょう。kvm の仕組だと、システムコールを実行して処理を行うことになるのでしょう。

`mpstat -P ALL 1` の出力を一部みてみると、一つの CPU の システム時間がとられているように見えます。

```
00 時 28 分 51 秒  CPU   %user   %nice    %sys %iowait    %irq   %soft  %steal   %idle    intr/s
00 時 28 分 52 秒  all    0.00    0.00   50.00    0.50    0.00    0.50    0.00   49.00   1421.78
00 時 28 分 52 秒    0    0.00    0.00   92.08    0.00    0.00    0.99    0.00    6.93    430.69
00 時 28 分 52 秒    1    0.00    0.00    6.93    0.99    0.00    0.00    0.00   91.09    991.09
```

`top` コマンドで見た場合には、kvm は通常のプロセスと同様に CPU 時間を消費し、メモリを消費しているように見えます。

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3746 dancer    25   0  308m 252m 246m R   82 25.7 100:36.60 kvm
 2512 root      10  -5     0    0    0 S    0  0.0   0:11.32 kjournald
 2806 root      15   0  2556  932  804 S    0  0.1   1:09.36 syslogd
```

`iostat 1 /dev/dm-1` の出力を確認してみます。ゲスト OS のディスク IO によってホスト OS のディスク IO が発生しているのが見れます。

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.00    0.00   42.50   42.00    0.00   15.50

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
dm-1            805.94       102.97      6400.00        104       6464
```

8.5 他との比較

Xen, qemu, qemu+kqemu, openvz, vserver, user-mode-linux などと機能を比較してみましょう。と思ったらすでにやられていたので、この URL を参照してください。`http://virt.kernelnewbies.org/TechComparison`

デビアン勉強会

Debian 勉強会資料

2007 年 1 月 20 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）