# 第 9 回 東京エリア Debian 勉強会
# 事前資料

Debian 勉強会会場係 上川純一*

2005 年 10 月 15 日

* Debian Project Official Developer

# 目次

# 1 Introduction To Debian 勉強会

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？

目的として下記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を情報共有する場をつくる
- まとまっていない Debian を利用する際の情報をまとめて，ある程度の塊として出してみる

また，東京には Linux の勉強会はたくさんありますので，Debian に限定した勉強会にします．Linux の基本的な利用方法などが知りたい方は，他でがんばってください．Debian の勉強会ということで究極的には参加者全員が Debian Package をがりがりと作りながらスーパーハッカーになれるような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．次回は違うこと言ってるかもしれませんが，御容赦を．

## 1.1 講師紹介

- 樽石さん apt-listbugs を開発した人です．
- 上川純一 宴会の幹事です．

## 1.2 事前課題紹介

今回の事前課題は「Debian のバグシステムに物申す」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．

### 1.2.1 澤田さん

特に物申すことがないので勉強会に先駆けて apt-listbugs のコードを読んでみました。ふ〜む、osdn.debian.or.jp に置いてある `index.db-#{severity}.gz` を拾ってきて critical や grave なバグを取得、さらに`#{bug_number}.status` を拾ってバグ情報を取得しているのか･･･、ということは apt-listbugs のリクエストはすべて osdn.debian.or.jp に行っているということになり結構な負荷がかかっているのではないかと思う。問題ない程度なのか、問題があるけど何らかの回避方法を使っているのかが聞ければいいな〜と思う。index.db と status をミラーに置くようにして apt-listbugs は sources.list に書かれているミラーから index.db と status を取得するようにするなんてどうなんだろうか。

### 1.2.2 えとーさん

広い意味での Debian のバグシステムの話になるが、セキュリティの報告などが CVE に重点を置かれているが、CVE は US-CERT [*1] はあくまでアメリカの国内向けのセキュリティ情報を扱う組織が行なっているので例えば

[*1] US-CERT について http://www.us-cert.gov/aboutus.html
The United States Computer Emergency Readiness Team (US-CERT) is a partnership between the Department of Homeland Security and the public and private sectors. Established in 2003 to protect the nation's Internet infrastructure, US-CERT coordinates defense against and responses to cyber attacks across the nation.

JPCERT/CC との連携などには積極的ではない。JPCERT が現状だめだ、というだけでなく重点が US-CERT 発行のものなのでどうしても、差が出てしまいこまったことになるのはどうにかならないのかなぁ。バグシステムについてはドキュンバグのフィルタリングは当該メンテナがやらなければいけないが、他の人からも評価というか注目度の上げ下げを付けられるようにするとよいのでわないだろうか。

### 1.2.3 中島さん

バグ報告できるのか！けれど英語で報告しなくてはならないのは、けっこう大変だから報告はしない。というよりできない。めんどうだから。それよりも、まずバグがよくわからない。「これはバグなのか？」というのがあるわけだし、自分自身が間違ってるかもしれないわけで、なのでバグがあったら自動的に報告するというのがいいと思う。バグだ。と思う前に勝手にコンピュータがバグ報告を送信するのだ。これで使えば使うほどバグがなくなるはずだ。これがいい。

### 1.2.4 小林さん

個人的には、Debian のバグ追跡システム (BTS) にそんなに不満はありません。白い背景に黒い文字という、何の工夫もない見かけが少し悲しいと思っていましたが、最近ではスタイルシートが適用されて少しよくなったようにも思います。

初めてバグ報告したときに嬉しかったのは、reportbug パッケージの存在でした。好みの問題もあるでしょうが、MUA を開いてメールを書かなくてもコマンドラインからバグ報告ができるのは、非常に嬉しいことです。

ただ、つい数日前に久し振りにバグ報告をして初めて気付いたのですが、sarge の reportbug コマンドでバグを送信しようとしたときに、「12 bug reports found:」のようなものは出ても、その下に既存のバグ報告のリストが出てこなくそのままバグ報告内容の編集に移ります。woody のはエラー終了します。2005 年 7 月の機能追加か何かのせいでしょうか? リリースされたばかりの安定版の reportbug の機能の一部に明確な問題が出るのは残念です。

reportbug は、あくまで BTS のウェブインタフェースをパースし、コマンドラインからのバグ報告を可能にしてくれるツールだと思います。しかしそれだと、今回のようにウェブインタフェースが変化してしまった場合に使えなくて困るので、もう少し BTS と一体化した、ウェブインタフェースに依らない (異なるインタフェースを使った) reportbug の仕様、または reportbug によるパース結果が変わらないようなウェブインタフェースの仕様が好ましいと感じました。

まぁ、バグ報告するなら不安定版を使えということなのかもしれませんが……。

### 1.2.5 上川

最近 BTS の機能が改良して，いろいろと新しいことができるようになった．見ためもかわって，いままでとどこおっていた開発が再開したのはよいことだと思う．ただ，従来の出力を解析していたツールがうまくうごかない．自分のコードで動かなくなったのは，たとえば emacs で debian/changelog を編集するモードがあるのだが，そこで BTS からバグのタイトルと送信者の名前をとってきてバグを close するための changelog エントリを書くというものがある．それが最近どうも動いていないらしい．ウェブページを正規表現で解析していたような記憶がうっすらとあるので修正したら一瞬で直るんだろうなぁ，とおもいつつもまだ修正するにいたっておりません．

# 2　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．

## 2.1　2005 年 37 号

2005 年 9 月 13 日です．

問題 1. バグトラッキングシステムの見栄えで最近かわったのは何か

☐ A CSS を利用するようになった
☐ B DHTML になった
☐ C XHTML になった

問題 2. Debian UK で問題になったのは何か

☐ A メンバーが活動的でないこと
☐ B UK の経済状況がよろしくないこと
☐ C 商用利用をしようとした場合の Debian という名前の商標の利用の許可をする基準が不明確だったこと

問題 3. ソフトウェアを計測する，という論文で発表されたのは何か

☐ A Debian sarge には 2 億 3000 万行のソースコードが含まれている
☐ B Debian sarge の品質を計測した
☐ C Debian sarge の利用しやすさを計測した

問題 4. Joey Hess は testing に対して security 対応をすることを発表した．それに利用しているサーバはどれか

☐ A secure-testing.debian.net
☐ B security.debian.org
☐ C security.debuan.org

問題 5. /usr/doc をいまだにつかっているパッケージ数はどれくらいか

☐ A 100
☐ B 200
☐ C 500

問題 6. planet.debian.org をメーリングリスト経由で配布しようという意見に対して出た反論は

☐ A blog の内容は機密事項なので，メーリングリストで配布してほしくない
☐ B メーリングリストとして配布するとサーバの負荷が高くなる
☐ C blog の内容を永続的にメーリングリストのアーカイブとして保存されたくない

問題 7. /usr/share/doc/パッケージ名/examples/ にあるファイルに実行権限をつけることについてはどうするべきか

□ A サンプルは実行できるものは実行権限をつけるべき

□ B サンプルなんてかざりなので実行しなくてよい

□ C /usr/share 以下について実行権限をつけるのはこのましくなく，実行ファイルは bin におくべきだ.

問題 8. sponsors.debian.net が提供するサービスは何か

□ A 金銭的寄付をつのるフィッシングサイト

□ B 広告を配信し，広告収入を Debian プロジェクトの発展のために利用するサイト

□ C まだメンテナになっていない人が管理しているパッケージについてスポンサーが必要な状況をトラッキングするシステム

問題 9. 1.0beta3 のようなベータ版のバージョン番号が 1.0 のような最終版のバージョン番号より低い，と dpkg が判定してしまう．この状況に関してメンテナはどう対応するべきか

□ A 優先度の低い チルダ記号 ~を利用して， 1.0~beta3 のような名前にする．ただまだアーカイブシステムが対応していないので，今後の改善が必要.

□ B あきらめる

□ C ベータ版はパッケージ化しない

問題 10. ソースのみのパッケージのアップロードを可能にするという提案についての反論は何か

□ A バイナリが必要でなくなると，メンテナがテストをしなくなるのではないだろうか，という懸念がある

□ B ソースのみだとパッケージインフラが破綻する

□ C katie を改変するのが面倒

問題 11. BTS に任意のタグを追加できる機能が追加された，なんという機能か

□ A tagtag

□ B たぐるんです

□ C usertag

## 2.2 2005 年 38 号

2005 年 9 月 20 日です.

問題 12. David Moreno Garza が wnpp にて close したバグレポートの数は

□ A 729 のバグレポート

□ B 100 のバグレポート

□ C 123 のバグレポート

問題 13. International Conference on Open Source Systems に投稿された論文の中で説明されていた結果は

□ A 開発者は短期間でどんどん入れ替わる

□ B メンテナは実は幻想で，そんな人は存在しない

□ C 長いあいだアクティブに活動し，パッケージの数も多くメンテナンスする

問題 14. Frank Lichtenheld が発表したのは，non-free なドキュメントを削除する処理を開始するということだっ

た．状況をトラッキングするために彼が利用したインフラは．
□ A BTS のの usertags 機能で debian-release@lists.debian.org ユーザのタグとして管理
□ B Wiki ページ
□ C CVS 管理のテキストファイル

問題 15. Software freedom day 05 で Debian-women が行って，結果として良かったので今後も継続することになったのは
□ A debian-women-new IRC チャンネルがよい結果をもたらしたので，今後は debian-women チャンネルに新人を歓迎する時間帯というのをもうける
□ B CD をたくさん焼いたら人気だった
□ C Katie や BTS などをインストールしてユーザがいじれるように提供したら人気だったので，今後もやる

問題 16. init.d スクリプトは現在直列に実行されているが，今後，並列実行を実装する際に便利だろうと思われる LSB 規格の仕様は
□ A なんとなく並列に実行しても壊れないようにする仕様
□ B 気持の中だけでは並列な年頃
□ C init スクリプトの中で依存関係を記述できる仕様

問題 17. 新しいバージョンのパッケージにて問題が解決した場合の，バグレポートをクローズする方法でないのは何か
□ A changelog でバグ番号を記述しアップロードする
□ B バージョンヘッダを付けて，リクエストを -done アドレスに投げる
□ C btsclose コマンドを利用する

問題 18. Marc Brockschmidt が説明した，新規メンテナプロセスの Front Desk の変更とは
□ A 今後はより厳しい思想チェックを行う
□ B Debian にコントリビュートしていることが要件になり，何もしていない場合は，応募が取り消される
□ C 年齢制限を設けます

問題 19. security.debian.org で問題になったのは何か
□ A セキュリティーアップデートが遅い
□ B セキュリティーアップデートが嘘だった
□ C xfree86 のセキュリティーアップデートがあまりにも高いネットワーク負荷を発生させてしまい，security.debian.org がサーバとして機能しなくなってしまった．

## 2.3 2005 年 39 号

2005 年 9 月 27 日です．
問題 20. Ben Hutching が Debconf について報告したのは
□ A もう終ってしまった事は忘れる
□ B 忘れ物がありました
□ C DVD が入手可能になった

問題 21. wiki.debian.org への移行で特に手動の労力が必要だったのはどこか

□ A すでに wiki.debian.net から wiki.debian.org に移行してしまっているページがいくつかあったのでそれに対しての手動の対処
□ B kwiki から moinmoin へデータ形式の変更
□ C ドメイン名の登録

問題 22. init の時点では/が read-only でマウントされているが，その時点でデータを保存するのにはどうしたらよいか.
□ A メモリファイルシステムを/run にマウントする
□ B /mnt 以下にメモリファイルシステムをマウントする
□ C /を rw にマウントしなおす

問題 23. グラフィックライブラリ GLU の実装が Debian 内で複数ある理由はなぜか
□ A 一部のコードが一部のハードウェアでしか動かないという状況が続いているから
□ B 複数のパッケージをメンテナンスしているほうがかっこいいから
□ C メンテナの仲が悪いから

問題 24. Jeroen van Wolffelaar が提案したのは
□ A libc5 を消す
□ B libc6 を消す
□ C libc6.1 を消す

問題 25. piuparts であきらかになる問題は
□ A purge する際に，essential ではないパッケージに依存して，動作しないパッケージ
□ B インストールしても動かないパッケージ
□ C 使ってみて使いにくいパッケージ

## 2.4 2005 年 40 号

2005 年 10 月 4 日です.
問題 26. DPL チームが今後検討する予定の問題について記録する媒体として選択したのは
□ A BTS
□ B IRC bot
□ C Wiki

問題 27. tetex 3.0 はどういう状況になっているか
□ A 今後も入る見通しがない
□ B うごかなくて困っている
□ C experimental にアップロードされ，ライブラリのフリーズが完了したら unstable に入る

問題 28. Debian で配布する IA64 アーキテクチャ向けのカーネルについて Dann Frazier が SMP じゃないカーネルのサポートを削除しようとした，何故か
□ A SMP じゃないといやだから
□ B 時代は SMP です
□ C IA64 で，SMP でないシステムがほとんどなく，あまりテストされていない

問題 29. Wolfgang Borgert によると planet.debian.org と，メーリングリストの利用方法の違いは
□ A メーリングリストは古い技術なので今後はつかわなくなる
□ B blog はフレームされないで意見を述べることのできるメディアだが，議論するのはメーリングリストでして欲しい
□ C planet.debian.org は安定していないので使わないで欲しい

問題 30. pbuttonsd は/dev/input/eventXX を利用しているが，どういう問題があったか
□ A makedev が，最大 32 あるうちの 4 個しかデバイスファイルをつくっていなかったので，/dev を静的に管理しているユーザは一部の機能を利用できていなかった.
□ B USB 接続ではうまく認識できなかった
□ C 電源ボタンがおされたらアプリケーションがハングした.

## 2.5 2005 年 41 号

2005 年 10 月 11 日です.
問題 31. Debian security で改善したのは
□ A バックエンドとフロントエンドのサーバを分割し，負荷に強い構成に変更した
□ B 特定のユーザが負荷をかけられないようにスロットリングした
□ C セキュリティーパッチをリリースしないことでサーバに負荷がかからないようにした

問題 32. Carlos Parra Camargo が報告したのは何か
□ A Wiki が悪意をもったユーザにより書き換えられていたので前のバージョンを復活させた
□ B Wiki がおもしろくないので改善しよう
□ C Wiki サーバがダウンしている

問題 33. mozilla 1.7.8 に対してのセキュリティーアップデートはどういう形でリリースされたか
□ A 1.7.10 にバージョン 1.7.8 という名前をつけてリリースした
□ B セキュリティーパッチをバックポートした
□ C セキュリティーホールのある機能を全て disable にした

問題 34. 複数の chroot で同じユーザ情報を利用するのに利用できる方法でないのは
□ A FUSE の shadow etc
□ B LDAP
□ C rm /etc/passwd

問題 35. ソースコードにローカルに適用したパッチをパッケージのアップグレード後も維持するためにはどうしたら一番楽か
□ A 自分でがんばる
□ B apt-src を利用する
□ C パッチはあてない

問題 36. Jurij Smakov がリリースした文書は何か

□ A Debian Users Handlebook: Debian ユーザをどうあつかえばよいのか，が書いてある
□ B Debian Developers Handlebook: Debian Developer をどう扱えば良いか，が書いてある．
□ C Debian Linux Kernel Handbook: Debian でカーネルがどうビルドされているのか，が書いてある

# 3 最近の Debian 関連のミーティング報告

上川純一

## 3.1 東京エリア Debian 勉強会 8 回目報告

前回開催した第 8 回目の勉強会の報告をします.

9 月の第 8 回東京エリア debian 勉強会報告．debconf の使い方や実装についてあつく議論がかわされました．今回の参加人数は登録者が 16 名くらいで，実際に参加したのが 14 名くらいでした.

DWN quiz に関しては，今回も満点をとった方がちらほら.

鵜飼さんが debconf について説明しました．開発をする際に，.config スクリプトを書くのに必要な事項などについて説明しました．backup するための実装や，いろいろと細かい挙動の話しになるとけっこうすごい実装になっているのでみんなびっくりしていたのではないでしょうか.

やまねさんが debian JP のウェブページをどうしたいのかということについて説明しました．ぜひコミットしていただきたいところです．誰にむかってウェブをつくっていくのか，という議論もでてきましたが，新しく開発者がどんどん参加できるような情報として，入り口としては必要なウェブページだろう，という意見がでていました.

# 4 apt-listbugs の生い立ちと実装

樽石

この文書は 2005 年 3 月に北京で開催された Asia Debian Mini-Conf in Beijin で発表した資料の日本語訳です。

## 4.1 はじめに

Debian は 24 時間 365 日、多くの開発者によって日夜開発されている自由なオペレーティングシステムです。わたしたちはこのような最新のスナップショットを Debian のアップグレードフレームワークを使うことで簡単に共有することができ、実際に、このフレームワークを利用して多くの人が最新の Debian スナップショットを利用しています。

一方で、この最新スナップショットはときどき壊れることがあります。apt-listbugs は、このような壊れたスナップショットからあなたのコンピュータを守るための仕組みです。

この短い文書では、apt-listbugs がどのようにコンピュータを守るのか、apt-listbugs の生い立ち、実装について簡単に説明します。また apt-listbugs の抱えている現在の問題点について紹介します。

## 4.2 仕組み

Debian には　Debian バグ追跡システム (BTS) として知られる、Debian に関する全ての問題が利用可能な中央管理型バグ追跡システムがあります。Debian のパッケージインストールフレームワークである APT は apt-get や aptitude 等を利用してインストールやアップグレードを行う直前に BTS からこれらの問題を取得するために apt-listbugs を呼び出します。apt-listbugs はどのパッケージがインストールされるのか、またアップグレードされるのかを自動認識し、問題となるバグを見付けた場合はその事をユーザに通知し、インストールを継続するか警告を出します。

このアプローチは、主にふたつの問題、ひとつはユーザの視点、もうひとつは開発者の視点を解決します。

### 4.2.1 ユーザの視点

ユーザの視点は apt-listbugs のメインの目的です。APT フレームワークは任意のパッケージを簡単にアップグレードすることができます。複雑な作業を一切行わずに単に 'apt-get upgrade' と実行するだけです。これはすばらしい機能ですが、一方でこのアプローチは壊れたパッケージさえも簡単にインストールさせてしまいます。多くの人が apt-get upgrade を利用しているため、結果として世界中の Debian マシンが一瞬のうちに壊れてしまうことになります。

実際は、壊れたパッケージの情報は BTS に既に存在している可能性が非常に高いのですがこの情報が BTS にあるかどうかを毎回チェックしないといけないとしたら apt-get upgrade による簡単なアップグレード作業は面倒な作業に一変してしまいます。これが apt-listbugs が必要な理由です。apt-listbugs はそのような情報が BTS にあるかどうかをあなたの代わりに毎回自動で行います。

### 4.2.2 開発者の視点

ふたつ目の目的は Debian 開発者のためのものです。基本的には、BTS は Debian パッケージに関するあらゆる情報知るために利用できるすばらしいものです。もし自分のパッケージにバグがあった場合、他のユーザが問題を BTS に報告してくれます。開発者はいつでも、どこでも、この情報にアクセスすることができます。

しかし、仮に自分の環境ではたまたま発生しなかった致命的な問題が最新のパッケージに含まれてしまったとします。こうなると、多くのユーザは問題の深刻さを考慮して急いでバグ報告をしようとします。それゆえ、たったひとつのバグに対して非常にたくさんのバグ報告を受け取ることになってしまい、開発者はバグ報告の分類という本質的でない作業に時間をとられ、本当に直したい致命的な問題の解決に長い時間を要してしまうことになります。もし、ユーザがバグをレポートする前に似たようなレポートを閲覧することができれば、重複したレポートを送ることはなくなるでしょう。

## 4.3 ヒストリ

apt-listbugs はもともとユーザの視点をなんとかしたいという個人的な目的で数年前に開発されました。ちょうどその時は、大学院を修了するために修士論文を書かなければいけない時期でした。ところが、何を思ったか〆切の一週間前に apt-get upgrade をしてしまったのです。結果、NFS が動かなくなり、この問題を修正しなければいけなくなりました。このバグレポートをきちんとチェックしていれば、このような問題にはあわなかったのですが、バグレポートを毎回チェックするわけにはいきませんでした。なぜならいつインストールされるかわからない致命的な問題のために毎回バグレポートをチェックするのは非常に大変な作業だからです。

### 4.3.1 CGI 問題

apt-listbugs の最初のバージョンは Debian にすぐにアップロードすることはできませんでした。その理由は apt-listbugs は BTS の CGI 出力を利用していたからです。CGI スクリプトはスクリプト言語で書かれていたため、もしこのパッケージをアップロードすると、BTS サーバが非常に高負荷になってしまうことは容易に想像できます。そこで、この apt-listbugs を利用したら同時にアクセスできるユーザ数は何人程度なのか知るために CGI のパフォーマンスを測定することにしました。結果は、1 パッケージのレポートを取得するのにかかる時間は約 1 秒でした。BTS サーバは 2 台の CPU を備えていましたので、結局 1 秒に 2 つのバグレポートしか取得できません。そのため、もし 10 個のパッケージをアップグレードしようとしたら、レポートの取得に 5 秒かかってしまいます。これは非常に大きな問題です。特に、BTS サーバは Debian のマスタサーバでしたから、もし Debian ユーザ全員がそのサーバからバグレポートを取得したら、これはほとんど DoS アタックのような状態に陥ってしまいます。

### 4.3.2 強制キャッシュ

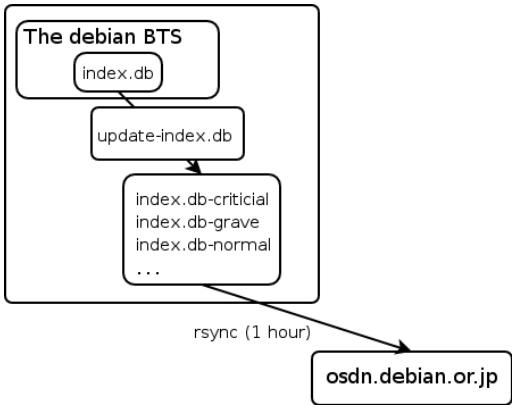
最初に考えた解決法は CGI 出力のキャッシュを行うプロクシサーバを利用することでした。プロキシサーバが http の No-Cache 属性を無視すれば、静的なデータを利用することができます。もちろん TTL の問題はありますが、DoS になってしまうよりは良いでしょう。

このサーバを用意したことで、とりあえず Debian に apt-listbugs をアップロードすることができましたが、はじめは experimental にアップロードして様子を見ていました。

### 4.3.3 index.db パーサ

はじめのバージョンの apt-listbugs はパーサに潜在的な問題を抱えていました。それはパーサが CGI 出力を利用していたことです。CGI のフォーマットはドキュメント化されていません。また、CGI スクリプト自体遅いという問題もあります。そのため、別の新しいパーサが必要でした。これは index.db パーサと呼ばれるパーサです。index.db パーサは BTS の内部データベースである index.db ファイルを直接パースするパーサです。index.db ファイル自体もドキュメント化はされていないのですが、このファイルは静的に生成されるものですので、CGI よりはずっと良い解です。

実際には、このパーサは index.db ファイルを critical や grave といった重要度毎に分割した index.db ファイルを利用します。




apt-listbugs には LDAP プロトコルを利用した実験的な LDAP パーサも存在しますが、このパーサは CGI パーサの代わりにするには意味がないものでした。その理由は当時の BTS LDAP サーバは内部的に tcl スクリプトを毎回よんでいるものだったからです。速度に変化はありませんでした。

> Andreas Barth 氏は BTS 用のネイティブな LDAP インタフェースを作成しました。そのため、現在ではこのインタフェースを利用してバグを取得することが可能になっています。詳細は `http://people.debian.org/~aba/bts2/ldap/` を参照してください。ただ、このインタフェースを利用するインタフェースはまだ実装されていません。

#### 4.3.4 SpamAssasin と apt-listbugs

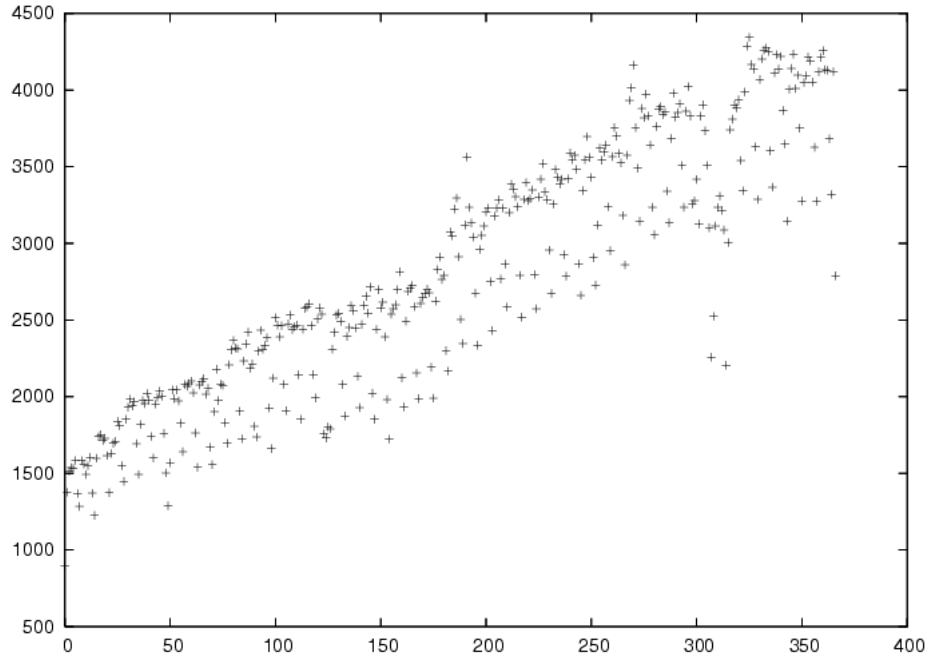
Index.db パーサを利用することで、apt-listbugs は動的なデータをサーバから取得することはなくなりました。そこで、この時点で一時的な解決であった強制キャッシュサーバを停止しました。

しかし、これは別の問題を生み出したのです。ちょうどその頃、BTS に対して非常に多くの SPAM メールが送られるようになっており、spamassasin が BTS サーバの CPU を非常に多く使うようになっていたのです。そして CPU 大量消費の原因が apt-listbugs にあるという勘違いされてしまったのです (Bug#207415)。

よく考えてみると、1 日で 46,000 個の静的データが負荷をこれほどあげるはずはありません。秒に換算すればわずか 0.5 個のだからです。通常の Web サイトはもっと多くの要求を処理できています。データサイズに関してはどうでしょうか。ひとつのデータは約 100 バイトですから、これも 50B/s 程度です。いずれにしてもこの問題の対策として apt-listbugs はサーバへの直接アクセスを禁止されてしまいました。

とはいえ、この問題が apt-listbugs によるものであったとしてもなかったとしても apt-listbugs がマスタサーバに直接アクセスするのはあまり良い考えではありません。そこでこの問題をきっかけに osdn.debian.or.jp にミラーサーバを構築しました。

現在、1 日に 4500 システムが apt-listbugs を使用しています。以下のグラフはどれくらいの Debian システムが apt-listbugs を利用しているかを表しています。X 軸は osdn.debian.or.jp にミラーサイトを構築した日からの日数、Y 軸は 1 日に osdn.debian.or.jp にアクセスする IP アドレスの数です。




## 4.4 実装

apt-listbugs は オブジェクト指向スクリプト言語のひとつである Ruby で記述されています。主に以下のクラスから構成されています。

```
* Acquire
 * HTTP
 * File
* Parser
 * CGI
 * Index
 * DSA
 * ReleaseCritical
* Bug
* Factory
 * PackageFactory
 * StatusFactory
 * BugsFactory
* Viewer
 * SimpleViewer
 * RSSViewer
```

### 4.4.1 Acquire

Acquire はデータを取得するクラスです。このクラスはサブクラスに対するキャッシュ機構を実装しています。実際の取得メソッド実装されておらず、サブクラスで実装します。例えば、HTTP サブクラスはデータを http から取得し、File サブクラスはデータをファイルシステムから取得します。

#### 4.4.2 Parser

Parser クラスは Acquire クラスで取得されたデータをパースし、バグ情報を保持する Bugs クラスを生成します。例えば以下の ruby コードは CGI 出力から apt-listbugs のバグ情を取得します。

```
acq = Debian::BTS::Acquire::HTTP.new
cgi_parser = Debian::BTS::Parser::CGI.new(acq)
bugs = cgi_parser.parse("apt-listbugs")
bugs.each do |bug|
  p bug
end
```

- CGI

CGI パーサは bugs.debian.org から取得した CGI 出力をパースします。

- Index

Index パーサは BTS システムの生データである index.db ファイルをパースします。apt-listbugs のデフォルトパーサです。

- DSA

DSA パーサは Debian セキュリティアドバイザリのページをパースします。

- ReleaseCritical

ReleaseCritical は Release-Critical ページをパースします。

#### 4.4.3 Bug

Bug クラスは ID、題名、重要度、タグ等のバグ情報を保持します。

これらのクラスは汎用的に作成されています。そのため、apt-listbugs 以外のプログラムからでもこれらのクラスを利用することが可能です。

#### 4.4.4 Factory

Factory クラスは「仕様書」からインスタンスを生成します。実際の作成はサブクラスにより行われます。このクラスは進捗表示等の Factory に汎用的なメソッドのみを実装しています。

- PackageFactory

PackageFactory は .deb ファイルの一覧からパッケージ情報を生成します。

```
# creating new packages database
new_pkgs = Factory::PackageFactory.create(pkgnames) do |msg, val|
  config.frontend.progress(msg, val) if config.quiet == false
end
Factory::PackageFactory.delete_ignore_pkgs(new_pkgs)
```

この処理が行われている間は、apt-listbugs から以下のメッセージが出力されます。

**パッケージフィールドを読み込んでいます**...

- StatusFactory

StatusFactory は指定したパッケージ名に対するの現在のシステムの情報を取得します。この処理中は以下のメッセージが出力されます。

**パッケージ状態を読み込んでいます**...

- BugsFactory

BugsFactory は Bug クラスで表現されるバグ情報を生成します。基本的に、この Factory はパーサクラスの単なるラッパーですが、パーサクラスは一度にひとつのパッケージのみを処理するのに対し、BugsFactory は複数のパッケージを処理できます。

**バグレポートを取得しています...**

### 4.4.5 Viewer

Viewer クラスはバグを表示するビューアを提供します。

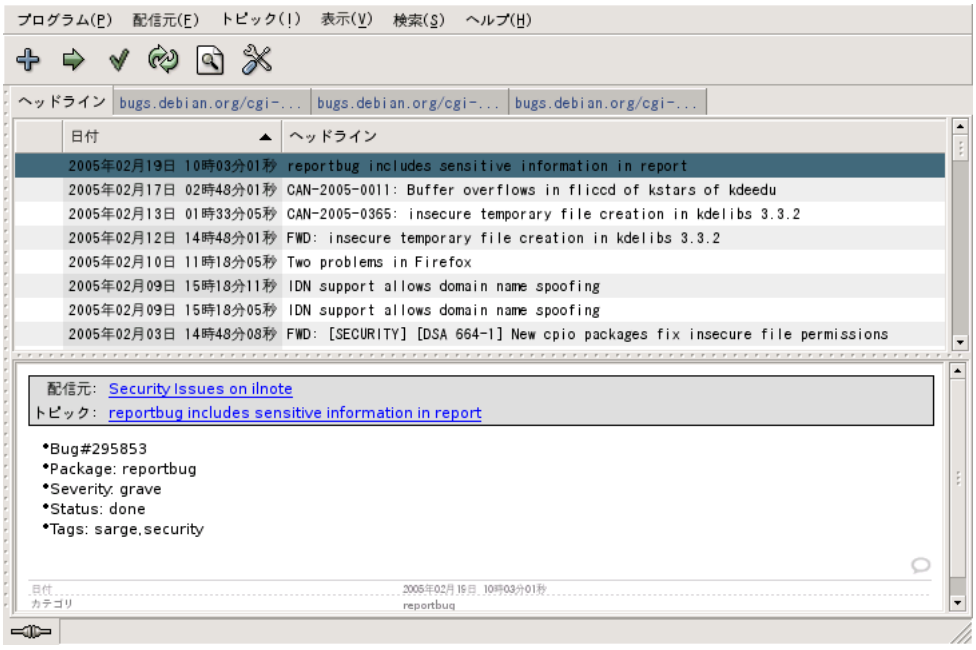
- SimpleViewer

SimpleViewer は以下のような処理を行う組み込みの対話的ビューアです。

```
critical bugs of cron (3.0pl1-86 -> 3.0pl1-87) <done>
 #282722 - Network install of Debian Woody Alpha -  cron corrupt on debian servers ?
grave bugs of strace (4.5.8-1.2 -> 4.5.9-1) <done>
 #294172 - strace - builds no s390 binary
grave bugs of gaim (1:1.1.2-3 -> 1:1.1.3-1) <done>
 #295904 - gaim: 1.1.3-1:  dies with SIGABRT on startup
grave bugs of reportbug (3.7.1 -> 3.8) <done>
 #295853 - reportbug includes sensitive information in report
grave bugs of cdbs (0.4.26-4 -> 0.4.27-1) <open>
 #295884 - cdbs: Changes control file to add new build dependency.
grave bugs of libdb4.3 (4.3.27-1 -> 4.3.27-2) <open>
 #294163 - libdb4.3: build failed on hppa
Summary:
 strace(1 bug), gaim(1 bug), cron(1 bug), cdbs(1 bug), libdb4.3(1 bug), reportbug(1 bug)
Are you sure you want to install/upgrade the above packages? [Y/n/?/...]
```

- RSSViewer

RSSViewer はバグ情報を RSS (Really Simple Syndication) 形式で出力します。



## 4.5 問題

現在、apt-listbugs が抱える一番の問題点は、apt-get upgrade を行うたびに apt-listbugs が大量のバグを表示してしまうことです。しかもほとんど役に立たない多くのレポートを読まなければいけません。それはそれほど重要でもない問題を重要だとタグ付けされた報告が非常に多いことが原因です。apt-listbugs は changelog ファイル中の ”closes: Bug#XXXX” という形式の文字列を処理することで不要なレポートを表示しないようにしていますが、上記のようなレポートはパッケージ保守担当者によって手動でクローズされてしまうため、表示から取り除くことはできません。

バグには open, done 等の状態があるのですが、これを使うことはできません。それはバグの状態というのは終局的なシステムだからです。バグ報告は通常そのバグを修正した新しいパッケージが Debian にアップロードされるとすぐにクローズされます。この段階では、バグ状態は変更されますが、そのバグを修正したパッケージはまだ利用できません。ミラーサイト使っている場合、この遅延時間もっと長くなるでしょう。

最初のバージョンの apt-listbugs はレポート文書中に存在する “Version” タグを利用しており、これによりいくつかのバグをフィルタすることができていました。この情報を利用すると

- 既にバグが存在している
- インストールしようとしているバージョンとは関係がない

といったフィルタリングが可能でした。しかし、現在は apt-listbugs が CGI にアクセスできないため、この情報を利用することができません。おそらく Since や Fixed-In といったような汎用的なタグ情報が必要であると考えています。

## 4.6 Tips

apt-listbugs はいろいろな利用方法があるためここでいくつか紹介します。

### 4.6.1 RSS によるセキュリティ問題

```
dpkg --get-selections | grep -v deinstall | awk -F' ' '{print $1;}' | \
   xargs -n $num /usr/sbin/apt-listbugs -y -q -T security --title \
   "Security Issues on `hostname --fqdn`" rss
```

このスクリプトは /usr/share/doc/apt-listbugs/security にあります。

### 4.6.2 作業中のパッケージで解決していないバグを見る

```
grep -e ^Package:  < debian/control  | cut -d' ' -f2- | \
  xargs /usr/sbin/apt-listbugs -S outstanding,open \
    critical,grave,serious,important,normal,wishlist,minor list
```

このスクリプトは /usr/share/doc/apt-listbugs/deblistbugs にあります。

## 4.7 まとめ

apt-listbugs はまだいろいろ問題を抱えていますが、基本的な概念は非常におもしろいものです。現在の興味は apt-listbugs に webrick 等を使って組み込み http サーバを導入することです。このフロントエンドを利用すると apt-listbugs の管理が非常に便利になります。また、bts2ldap を利用した LDAP バックエンドも必要であると考え

ています。

## 4.8 参考

- Debian - http://debian.org/
- Debian Bug Tracking System - http://bugs.debian.org/
- http://lists.debian.org/debian-devel/2003/08/msg02376.html

# 5 debbugs の内部構造

上川

## 5.1 はじめに

この文書は Anthony Towns が フィンランドの debconf 5 で発表した内容を日本にて展開するための資料です．Anthony Towns の作成した英語の資料を省略して抜粋しています．また，それ以降に変更した事項について追記しています．

Debian Bug Tracking System (BTS) は，ほぼ Debian に特化したバグ報告の管理のためのシステムです．他のプロジェクトでも利用されていることもありますが，Debian でバグがパッケージベースで厳格に分類できることなどの特性が反映されているため，Debian プロジェクトのワークフローで使いやすいように作られています．[*2]

規模としては，55000 以上の現在アクティブなバグ報告，231000 のアーカイブされたバグ報告を現在保持していて，毎週 1000 以上の新規のバグ報告が追加されています．ウェブインタフェースは追加された報告をすぐに反映しており，過去，ダウンタイムもほとんど発生していません．

Anthony Towns によると下記がバグトラッキングシステムの要件です．

- インタフェース： 開発者がメールで操作できるようになっており，誰でもウェブで閲覧できるようになっている．
- パッケージベース： バグ報告をパッケージ別に高速に管理する必要がある
- スケーラビリティー: 大量のバグ報告に対応できる必要がある
- 即時性： 現在のバグの状態をすぐに報告してくれる必要があり，バグの状態が変更されたらすぐに反映される必要がある
- 安定性： 継続して動作する必要がある．新規の機能がどんどん追加されたとしても．
- 公開： 議論の内容に Debian コミュニティー全体として参加できるように，永続的な公開記録として保存される必要がある．

## 5.2 データ形式

バグデータベースのスプールの形式は下記です．リレーショナルデータベースなどは利用していません，スプールディレクトリ以下にほとんどのデータが格納されています．

各バグについて，ファイルはそれぞれ４個あります．サマリーファイルはメタデータを保存します．ログファイルは，そのバグに対して流れたメールを全て保存します．

status ファイルは互換性のためだけに存在しています．report ファイルは，最初のバグ報告のメールで，バグが close されるときに送信されるものです．

- /org/bugs.debian.org/spool
  - incoming/
    - T.*
    - S[BMQFDU RC] *.*

[*2] Debian のインフラと統合されており，changelog にバグ番号を記述してパッケージをアップロードしたらバグが修正されたと記録されるようになっていたりします．

    - R[BMQFDU RC] *.*
    - I[BMQFDU RC] *.*
    - G[BMQFDU RC] *.*
    - P[BMQFDU RC] *.*
- db-h/
    - 00/
        - ..
        - 314200.log
        - 314200.report
        - 314200.status
        - 314200.summary
    - ..
    - 99/
- archive/
    - 00/
    - ..
    - 99/
- index.db – index.db.realtime へのシンボリックリンク
- index.archive – index.archive.realtime へのシンボリックリンク
- nextnumber

#### 5.2.1 incoming

incoming に来たメールは処理中，名前を変えます.

- T receive によってうけとられた
- S SPAM 確認待ち
- R SPAM 確認中
- I SPAM チェック通った
- G service か process スクリプトを通った
- P process 中

また，ファイル名の二つ目の文字はどこのメールアドレスにメールが送信されてきたものなのかということを示します．ファイル名ののこりは，バグ番号と，一意な ID です．一意な ID を決定するのに現在は時間とプロセス番号を利用しています.

- B: 通常のバグ報告．submit@ 1234@
- M: -maintonly メーリングリストに投げない
- Q: BTS に登録しない．-quiet
- F: アップストリームにフォーワード -forwarded
- D: バグ終了 -done
- U: サブミッターにメール -submitter
- R: ユーザのリクエスト用インタフェース request@
- C: デベロッパーの制御用インタフェース control@

### 5.2.2 Status と Summary

status ファイルの中身は行ベースです．無い行については空行とみなします．このファイルは今後なくしていこうとしています．

- バグ報告者のメールアドレス
- 時間 (秒)
- サブジェクト
- 元のメールのメッセージ ID
- バグがアサインされているパッケージ
- タグ
- close した人のメールアドレス
- 上流のメールアドレスか URL(forward されたばあい)
- マージされているバグ番号
- severity

summary ファイルは RFC822 形式で，拡張可能になっています．現在 Format-Version: 2 と 3 の二つの形式があります．3 は，ヘッダについては RFC1522(MIME) のデコードされた形式になっています．

- Format-Version: このファイル形式のバージョン
- Submitter: バグ報告者のメールアドレス
- Date: 時間 (秒)
- Subject: サブジェクト
- Message-ID: 元のメールのメッセージ ID
- Package: バグがアサインされているパッケージ
- Tags: タグ
- Done: close した人のメールアドレス
- Forwarded-To: 上流のメールアドレスか URL(forward されたばあい)
- Merged-With: マージされているバグ番号
- Severity: severity
- Owner: バグの所有者

### 5.2.3 log ファイル

あらゆるメールが log ファイルには追記されていきます．また，メタデータも追記されていきます．残念ながら，メタデータは生の HTML で書かれており，またバージョンによって記述の仕方が変わっており，さらに悪いことに，古いバグの中にあるテキストは更新されていないため，機械的に処理することは難しくなっています．

また，コントロール情報は，行頭のエスケープコードにより切り替わります．メールの中にエスケープコードのような文字列が出て来たら，それは文字コード 030(8 進数) の文字を追加してエスケープします．

詳細は Debbugs::Log を見てください．

- kill-init: まだ一行も処理していません
- incoming-recv: 07: あとに go がくる，Received:行
- autocheck: 01: X-Debian-Bugs-..: までの無視されている行，autowait が次に来る
- html: 06: 生で表示すべき HTML
- recips: 02: メールの受取人，04 で分割されている
- go: 05: メールの文書

- go-nox: X: メールの文書，X ではじまる行
- kill-end: 03: メッセージの終り.
- autowait: go-nox があとにくる，空行まで無視されるその他の情報.

#### 5.2.4 Index ファイル

index ファイルは，pkgreport.cgi がどのパッケージにどのバグがわりあてられているかを確認するための情報です.

以前は，by-package.idx と by-severity.idx というのがあり，高速化に貢献するはずだったのですが，一年以上長い間生成されていなかったうえに，生成されていなかったことに誰も気づかなかったので必要ないんじゃないだろうか，ということです.

データ形式としては下記のようになります．パッケージ，バグ番号，時間，ステータス，メールアドレス，severity の順に書いた行が全てのバグに対して作成されています.

```
pbuilder 317998 1121196782 open [Junichi Uekawa <dancer@netfort.gr.jp>] normal
```

### 5.3 コード形式

debbugs は特に設計もされずに長い間パッチを累積してきました．ただ，明確にわかれている部分はあって，メールを処理するコアのインタフェースのスクリプトと，ウェブを表示するための CGI 部分とで分離できます.

設定ファイルは全て/etc/debbugs にあります.

#### 5.3.1 コアのスクリプト

メールを処理する部分があります.

- errorlib: ライブラリ
- receive: MTA からメールを受信する
- spamscan: 受信メールを SPAM チェックする
- processall: process と service にメールを分配する
- process: バグメールを処理する
- service: control@ と report@ メールを処理
- expire: close されてから 28 日過ぎたバグをエキスパイア処理する
- rebuild: index ファイルをリビルド

receive と rebuild 以外は cron から起動しています．15 分に一回しか動作しません.

#### 5.3.2 CGI スクリプト

CGI 関連は，errorlib 関数を活用している部分もありますが，ほぼ独立しています.

- bugreport.cgi: バグレポートを一つ表示
- pkgreport.cgi: パッケージやサブミッタなどでサマリを作成する
- pkgindex.cgi: パッケージや severity に対して数を表示
- common.pl: ライブラリとして利用

pkgreport.cgi はユーザが直接ウェブでたたくため，特に速度が重要視される部分なので，触る場合には注意してください.

#### 5.3.3 ハックするには

debbugs のソースは CVS にあります．また，Debian Developer であれば，ミラーが merkel.debian.org の/org/bugs.debian.org 以下にあります.

## 5.4 そして何がおきたか

Anthony Towns の発表でどういう結果がもたらされたか見てみましょう.

### 5.4.1 バージョントラッキング

バグがどのバージョンで発見され，どのバージョンで修正されたのかというのをトラッキングできるようになりました．従来は発見されたバージョンだけが Version ヘッダで分かるようになったのですが，それ以外の情報も保持するようになりました.

http://lists.debian.org/debian-devel-announce/2005/07/msg00010.html

バグ番号を保持して操作するための BTS のコマンドは下記です.

```
close バグ番号 バージョン
reassign バグ番号 パッケージ バージョン
found バグ番号 バージョン
```

また，katie が変更され，バグを close するメッセージには，下記のヘッダが付くようになりました．それを BTS が処理して close されたバージョンを把握できるようになりました.

```
Source-Version: バグ番号
```

CGI に対して，version を指定すると，そのバージョンでの状態がでてきます．329344[*3]は，0.4 では open だったが，0.5 では close になったというのが http://bugs.debian.org/cgi-bin/pkgreport.cgi?pkg=cowdancer&version=0.4 と http://bugs.debian.org/cgi-bin/pkgreport.cgi?pkg=cowdancer&version=0.5 二つのページを比較するとわかります.

/org/bugs.debian.org/spool/db-h/44/329344.summary を見ると，メタデータとして保存されているのがわかります.

```
Format-Version: 2
Found-In: cowdancer/0.4
Done: Junichi Uekawa <dancer@debian.org>
Subject: cowdancer: cow-shell does not start, gives error
Date: 1127295198
Submitter: Francesco Potorti‘ <Potorti@isti.cnr.it>
Fixed-In: cowdancer/0.5
Package: cowdancer
Message-Id: <E1EIOYD-0003lE-00@pot.isti.cnr.it>
Severity: grave
```

### 5.4.2 ユーザタグ

http://lists.debian.org/debian-devel-announce/2005/09/msg00002.html

request@bugs.debian.org に対して下記のようなメールをおくればタグが追加できます.

```
    user aj@azure.humbug.org.au
    usertag 18733 + good-reasons-to-run-for-dpl
    usertag 18733 + still-cant-believe-it-finally-got-fixed
    usertag 62529 + your-days-are-numbered
```

見る際には，users=でユーザを指定するとタグが見えるようになります.

http://bugs.debian.org/cgi-bin/pkgreport.cgi?pkg=dlisp;users=dancer@debian.org

また，tag でタグを指定して，users=でユーザを指定するとそのユーザで作成したタグを全て検索することができます.

http://bugs.debian.org/cgi-bin/pkgreport.cgi?tag=ignore-for-now;users=dancer@debian.org

### 5.4.3 バグ購読

バグ番号に対してメーリングリストのようにして利用することができるようになりました．http://lists.debian.org/debian-devel-announce/2005/07/msg00014.html

[*3] http://bugs.debian.org/329344

**バグ番号**-subscribe@bugs.debian.org にメールを出すと登録するようにメールがかえって来るので，それに返信すると，バグ番号に登録されます.

### 5.4.4 バグブロッカー

どのバグがどのバグによって邪魔されているのかというのをトラッキングするための機能が追加されました.

```
block 保留中のバグ番号 by 原因のバグ番号
unblock 保留中のバグ番号 by 原因のバグ番号
```

### 5.4.5 mindays maxdays

mindays, maxdays オプションが追加されました．バグ報告の報告されてからの日数で表示させるかさせないかを選択できるオプションです.

http://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=dancer@debian.org&maxdays=90 や http://bugs.debian.org/cgi-bin/pkgreport.cgi?maint=dancer@debian.org&mindays=90 として入力できます.

### 5.4.6 バグ検索システム

Google が master.debian.org にとって DoS になるような検索の仕方をしていたので，現在 BTS は google の検索対象にははいっていないので検索サービスが必要だろう，という話題が出ていました.

鵜飼さんが全文検索エンジンサービス (FABRE) を実装しましたが，まだ本格的に使われるような状態にはまだいたっていないようです．結構このサービスは負荷が高いのが問題になると思われます．http://fabre.debian.net/

### 5.4.7 debian-bugs.el はまだ動くのか

reportbug は vi ユーザを中心としたインタフェースになっていますが，Emacs を利用している人でも，debian-bugs.el を利用して debian BTS を操作することができます.

たとえば，debian-changelog-mode を使っている場合であれば，changelog のエントリーを自動生成するようになっています．メニューから Bugs close を選択するか，debian-changelog-close-bug でバグ番号を選択する (タブ補完がききます) と，下記のようなエントリーが作成できます.

```
dsh (0.25.6-2) unstable; urgency=low

  * Bug fix: "How to control dsh timeout time?", thanks to Junichi Uekawa
    (Closes: #281012).
  * Bug fix: "allow exclusion of host from list of hosts.", thanks to
    Junichi Uekawa (Closes: #289766).
  * Bug fix: "dsh: -c -i hangs if no input under current design", thanks
    to Charles Fry (Closes: #241531).
```

この機能は 207852[*4] で上川の出したパッチが発端で実装されましたが，気づいたら正規表現が巨大になっています.

ひさしぶりにソースコードをみたら，現在の実装は，正規表現をつかいまくって HTML を解析しているため，何かイレギュラーなことがあった場合には，動作しなくなります．該当する関数は emacs-goodies-el:/elisp/debian-el/debian-bug.el(debian-bug-build-bug-menu) です.

BTS のフォーマットが変わったので，何かうごかなくなっていないかと心配していましたが，特にうごかないということはないようです.

みてみると submitter のメールアドレスがうまく解析できなかった場合には'thanks to XXXX (closes: XXXX)' が追加されないという仕様になっています．たまにこれが発生するので，再現する条件をさがしてバグを直したいですね.

[*4] http://bugs.debian.org/207852

```
      (with-temp-buffer
        (message "Fetching bug list...")
        (call-process "wget" nil '(t t) nil "--quiet" "-O" "-"
                      (concat
                       "http://bugs.debian.org/cgi-bin/pkgreport.cgi?src="
                       package))
        (message "Fetching bug list...done")
        (goto-char (point-min))
        (while
            (re-search-forward
             "\\(<H2.*</a>\\(.+\\)</H2>\\)\\|\\(<li><a
href=\"\\(bugreport.cgi\\?bug=\\([0-9]+\\)\\)\">\\(#[0-9]+: \\(.+\\)\\)</a>\\)"
             nil t)
          (let ((type (match-string 2))
              ;;(URL (match-string 4))
                (bugnumber (match-string 5))
                (description (match-string 6))
                (shortdescription (match-string 7)))
            (cond
             (type
              (setq bugs-are-open-flag (not (string-match "resolved" type)))
              (save-excursion
                (set-buffer debian-bug-tmp-buffer)
                (insert "\"-\"\n\"" type "\"\n")))
             (t
              (setq bug-alist (cons (list bugnumber description) bug-alist))
              (when bugs-are-open-flag
                (when (and (re-search-forward
                            "Reported by: <a class=\"submitter\"
href=\"pkgreport.cgi\\?submitter=[^;]+;arch=source\">"
                           nil t)
                           (or (looking-at "&quot;\\(.*\\)&quot; &lt;")
                               (looking-at "\\(.*\\) &lt;")))
                  (setq shortdescription
                        (concat "Bug fix: \"" shortdescription
                                "\", thanks to "
                                (debian-bug-rfc2047-decode-string
                                 (match-string 1))
                                " (Closes: #" bugnumber ").")))
                (setq bug-open-alist
                      (cons
                       (list bugnumber shortdescription)
 bug-open-alist)))
```

実際に changelog に追加する部分は emacs-goodies-el:elisp/dpkg-dev-el/debian-changelog-mode.el(debian-changelog-close-bug) にあります.

# 6 実際の当日の内容

上川

当日のタイムテーブル

- 18:10- quiz
- 18:30- こたえあわせ
- 19:00- 休憩
- 19:10- たるさん
- 20:10- 上川
- 21:00- 宴会

## 6.1 たるさん

osdn.debian.or.jp の rsync もとはもともと master.debian.org．バグ報告があったので， merkel.debian.org に切替えた．

グラフは２００３年９月から２００４年１０月．今は一年たっているので，おそらく７０００ IP アドレス．

中国で発表したときに，何分もかかった．ミラーサーバ必要だ，という話になった．がいまだになにもできていない．

RSSViewer を使えば，自分のマシンに今入っているパッケージのセキュリティー関連のバグだけを見る，ということができる．実は便利かもしれない．

メンテナンスにあきてきたのでどうせならかきなおしたいなぁ...

# 7　グループワーク用メモ

# 8 次回

関西出張会議を 10 月 29 日に実施する予定です.
また，東京での次回は 11 月 12 日土曜日の夜を予定しています．内容は本日決定予定です.
参加者募集はまた後程.

**Debian 勉強会資料**

2005 年 10 月 15 日　　初版第 1 刷発行
東京エリア Debian 勉強会 （編集・印刷・発行）