

A large, light pink circular brushstroke graphic is centered in the background of the slide. It has a textured, hand-painted appearance with varying shades of pink and white.


東京エリア Debian 勉強会

資料

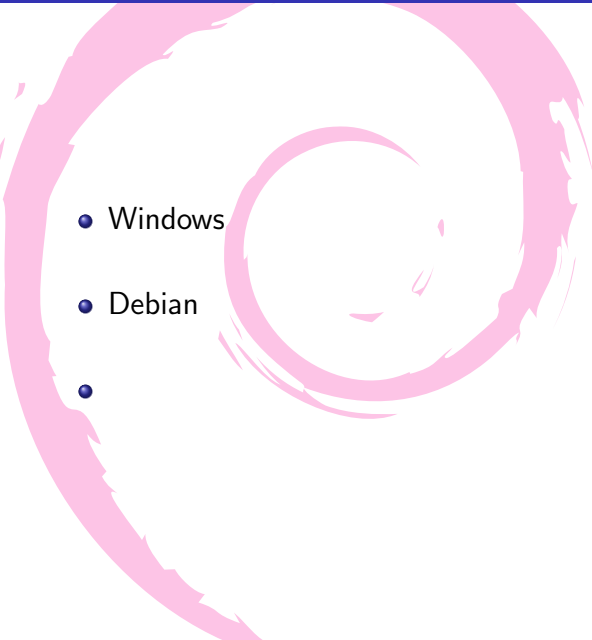
上川 純一 dancer@debian.org
IRC nick: dancerj

2007年4月21日

本日の agenda

- 注意事項
 - 飲食禁止
 - 政治/宗教/営利活動禁止
 - 最近事情 etch のリリースについて
 - 事前課題紹介
 - quiz
 - quilt
 - darcs
 - git
- 

OSC の agenda

- 仮想化友の会、Debian 勉強会の紹介
 - 事前課題紹介
 - 仮想化常識 Quiz
 - Windows から見える仮想化世界
 - Debian の仮想化技術紹介
 - 最後に
- 



etch リリース

- 4月8日 リリース
- 4月12日 宴会開催
- 4月XX日 日本語版リリースノート完成

事前課題

「私はバージョン管理システムをこのようにつかっています」もしくは「バージョン管理システムを使わずにこのようにしています」

青木さん

VCSはCVSとSVNを使っていますが、あまり複雑なパッチ管理ではなく単に共同開発の最新版管理レベルです。それとバックアップ的な意味合いで使います。ハンドマージより凝ったことといえばVIMDIFの利用ぐらいがせいぜいです。タグ管理はとくいではないです。DVCSをつかうべきかと思いつつ、勉強不足ですね。

Kentaro Waki さん

特に変わった使いかたはしていません。一応、プログラマなのでソースコードのバージョン管理に「subversion」使ってます。つい最近まで「CVS」使ってました。「subversion」で気に入ったのは、「ファイル名の変更」「ブランチの扱い」などです。また、「windows」ユーザと関係する場合に便利そうに見えます。まだ実際には触ってませんが、「TortoiseSVN」は「windows」ユーザに違和感の少ないUIっぽいので。「subversion」関連で、「Trac」の評判も良いようなので導入を検討しています。

森田尚さん

個人で余暇に書いているプログラムのソースコード管理と、仕事（出版社で書籍の企画・編集をしています）での原稿管理のために、Subversion リポジトリを Apache でホストして使っています。2002-2004 年ごろは主に CVS を使っていました。

自分の仕事では、協働するチームメンバが複数の組織と拠点に分散していることが多いので、いずれ分散型 VCS を導入したいと考えています。

数世代にわたるバックアップが欲しいだけのときなど、VCS を使うまでもない場合のために、ホームディレクトリその他を pdumpfs のバックアップ対象にしています。

「バージョン管理システムを使わなくて済むようにこのようにしています」何故バージョン管理システムが必要なのか？一つの物に複数によってたかって改変を加えるからである。であれば、使わなくて済むようにするアプローチとして

- 複数の改変を同時にしない
- 複数の改変が相互に影響しないようにする

のどちらかを行えば良いのである。前者は改変を一人が行うと言う事になるが現実的ではない後者を実現するためには、設計プロセスが完全でなければならない要するに、ソフトウェア開発であれば、モジュール化が適切に行われているなど、設計の瑕疵を無くせばバージョン管理システムは不要である。と言う事に行き着く。

実際の所、そういった状況で作られるソフトウェアはまず無い。多くの開発現場では、混沌の中から偶然の産物の様に作り出されたりする。この状況が存在する限り、やはりバージョン管理システムは必要とされ続けるのだろう。だが設計プロセスを注意深く行えば、設計プロセスの瑕疵を無くす手法が見つかれば、なんとか実現できるかも知れない。

今のソフトウェア開発プロジェクトでは、何とバージョン管理システムを使っていない。複数の人が同じファイルを編集しようとすることも実際にあり、その状況はかなり問題である。プロジェクト全体としてバージョン管理を導入する考えはないようである。そこで、自分のチーム内で独自にソースコードを管理する方式を考えているところである。使用するツールとしてはSubversionを考えている。チームの担当範囲のファイルは決まっているので、バージョン管理対象は明確ではあるが、開発環境がよく言えば特殊、悪く言えば洗練されていないために実際にきれいに適用するのは一筋縄ではいかない。こんな環境でもうまく運用できる方法を考えているところである。

その他、自分だけではあるがパッチ管理として quilt を使ってみたことがある。自分の変更点をパッチ (diff) にするのが自動的にできるのは便利だが、自分の作業中に他人が同じファイルを編集するということが起こると混乱してしまった。こういう用途には向かないようだ。

私はバージョン管理システムをこのようにつかっていたり、
いなかったりします（しました）

一年前まで：

バージョン管理システムを使ったことがありませんでした。
当時は、実験データ解析用のコードを書いていたが、
hoge_20051204.cc のようなファイル名だけで管理していま
した。（頻繁に書き直したりするわけじゃないので、それで
済んでいた）CVS とか名前は知っていたけど、覚えている
余裕がありませんでした。

一年前：

仕事で初めて VSS を使いました。チェックアウトするとロックされてしまうのが不便じゃね？と思いました。その後、VSS に上げる前のソースコードを自分のローカルマシン上で管理するために Subversion を使っていました。「達人プログラマー」で、ソースコード以外の普通のドキュメントもバージョン管理システムで管理しよう、という話を読んで、なるほど！と思いました。結局、今に至るまで実践はしていません。

今：

今度の現場では VSS を使わせてくれないらしいです。それほどコードを書く必要のないプロジェクトなのですが、台帳で管理するとか言う話です。フリーソフトのインストールも不可なので、svn を入れて使ったりする事も出来ませんが、一番つらいのは emacs(meadow) を使えない事なのですが、オフトピですね)

鈴木さん

バージョン管理は、cvs と subversion を使ったことがあります。cvs は、Web(tlec.linux.or.jp) の更新で利用しています。チェックアウトすればどこでも更新できるので便利です。最近余り更新してませんが。subversion は、ドキュメント作成を会社と家で更新できるような1日の区切りでチェックインしていました。使ったり使わなかったりで持続しないです。理由は、リリースするときに tar にまとめてバックアップしているので頻繁に subversion は使ってないです。

個人で管理しているサーバに、バージョン管理システムとして Subversion を、バグトラッキングシステムには trac を利用しています。特に変わった用途として使ってはならず、通常どおりソースコード管理およびバグトラッキングとして使用しています。主にクライアントマシンとして Windows を使用している関係から、TortoiseSVN を Subversion のクライアントとして利用しています。

新たにリリースされた Etch への移行を検討しており調査中ですが、trac-ja-resource パッケージがうまく動作してくれず、難航しています。

小室 文さん

バージョン管理は使っていません。会社ではバージョン管理を使うような人や案件はないです。プライベートでも管理する物も一緒に管理したい人もいないので、導入していません。

バージョン管理を導入する場合、使う人が対等な位置にいるのが前提な気がします。なので、他の案件の一部分の機能を追加するためにファイルを検証サーバにアップする際は(1)事後報告するか(2)案件管理者にファイルを送る事が多いです。

使ってみないと！と思いつつも必要に迫られていないのでよく理解していません。

回答：

個人的に作成しているプログラムはたいした量ではないので、「使わなくてすむように」というよりは、「使う必要がない」という状態です。

プログラムを構成するファイルは、修正前にファイル名にバージョン番号を付けて、全バージョンそのままの形で保存してあります。（若しくは、ディレクトリにバージョン番号を付けて丸ごとコピー。） 規模が小さいのでこれで十分管理できてしまいます。

「私はバージョン管理システムをこのようにつかっています」

内容が日々進化していくファイル(プログラム・ドキュメント・翻訳・図など)は何でもリポジトリに入れてバージョン管理下に置いています。バージョン管理システムのない生活はもう考えられなく、バージョン管理システムがあってこそ効率的な仕事ができると思うようになっていきます。

これまでは主に Subversion などの中央集権的なバージョン管理システムしか使ってきませんでした。最近、Debian のウェブサイトやリリースノートの日本語訳コーディネータとして働くようになってから、分散バージョン管理システムにも興味をもつようになりました。翻訳チーム全員にコミット権を与えるわけにはいかないというのはプロジェクトとしては仕方がないことだと思いますが、他方でコーディネータとしては、あらゆるコントリビュータの仕事はきちんと区別し、分割してコミットしたいのです。しかしそのような作業をコーディネーター一人でやるのは大変なので、分散バージョン管理システムを導入して、本家リポジトリへのコミット権はもってなくても自分のリポジトリにコミットできる翻訳者やレビューアにはどんどん自分で作業をしてもらったほうが、効率がいいのではないかと考えています。

えとーさん

自前のソースコードの管理や参加しているプロジェクトのソースコードの管理に利用しています。
設定ファイルやその他雑多なものはあまり利用できていないので、今後も勉強しながら便利に使っていきたいと思っています。

バージョン管理システムは、仕事で cvs をつかっています。ソースコードから word や excel の資料からメモに至るまですべてまとめて cvs で管理していました。今度配属されたプロジェクトでは Mercurial (OpenSolaris など採用されている分散 SCM) を使用することになり、今まで使ってきた cvs とは勝手が違うために戸惑っております。今回は分散 SCM が 2 種類紹介されるようですので、この機会に分散 SCM の有効な使い方を学べたらと思っています。よろしくお願ひします。

従来、自作プログラムは環境情報をまとめたメモと一緒にソースファイル一式を tar.bz2 で固めて蓄積してました。この方法ではソースを管理するという点では問題は無いのですがそれ以外への発展が無く、方々の作業で無駄が発生しました。代表的なのが、差分抽出により目的外の変更が入っていない事の確認です。このような事をサポートしてくれるツールとしてバージョン管理ツールの DIFF 機能を利用しています。

またバージョン管理業務を長く行っていると

- 「何故バージョンを更新したのか」とか
- 「どうして、このような作りになっているのか」

といった事が忘れてしまいがちです。このためバグトラッキングシステム (Trac) や自動ドキュメント化システム (Doxygen) 等との連携が今後の課題になっています。




今回のお題にはあがりませんでした。私は以下のようなシステムで個人的には作業をしています。

- バグトラッキング、仕様書管理 Trac
- バージョン管理 SubVersion/SVK

Debian 常識クイズ


Debian の常識、もちろん知ってますよね？ 知らないとはずかしいけど知らないとは言えないいろいろなこと、Debian Weekly News をベースに確認してみましょう。

問題1. ウェブアプリケーション関連のパッケージの静的コンテンツはどこにおくべきか？

-  A `/var/www` に置く
-  B `/usr/share/PACKAGE` に置く
-  C `/srv/XXX` に置く

問題1. ウェブアプリケーション関連のパッケージの静的コンテンツはどこにおくべきか？

答えは:

•  A /var/www に置く

•  B /usr/share に置く

•  C /srv/XX に置く



B

問題2. Debian Project の MIA アカウントに対して 実施する WaT とは何をするものか



A 今年の DPL 選挙に投票しなかった人に対して確認メールを送り反応がない人を引退プロセスに移行する



B 気に入らない人を強制退会させる



C あれ？ Debian Developer だらけの水泳大会

問題2. Debian Project の MIA アカウントに対して 実施する WaT とは何をするものか



A 今年の DPL 選挙に投票し
なかった人に対して確認メー
ルを送り反応が得られなかった
プロセス



B 気に入ら
させる






C あれ？[
らけの水泳

答えは:



A

問題3. etch リリースはどのような暗号鍵で署名されるか？

-  A オンライン鍵とオフライン鍵
-  B オフライン鍵のみ
-  C オンライン鍵のみ

問題3. etch リリースはどのような暗号鍵で署名されるか？



A オンライン鍵とオフライン鍵



B オフライン鍵






C オンライン鍵

答えは:



A

問題4. Frans PopがアナウンスしたBabelboxは何をするものか？

-  A いろいろな言葉を喋ってくれる
-  B フォントを複数表示
-  C 自動でくりかえし Debian Installer が稼働し、Gnome にしばらくログインしてくれるしくみ

問題4. Frans PopがアナウンスしたBabelboxは何をするものか？

答えは:



A いろいろな言葉を喋ってくれる



B フォン





C 自動で
Installerか
しばらく
しくみ






C

問題5. DPL 選挙の勝者は？

-  A Iwamatsu
-  B Sam Hocevar
-  C Anthony Towns

問題5. DPL 選挙の勝者は？

-  A Iwamatsu
-  B Sam Ho
-  C Anthony

答えは:



B

世界中に分散して開発しているオープンソース開発の現場では必須となる SCM。最近中央集権的な CVS・SVN から分散管理的なツールへの移行が進んでいます。最近の流行なんだけどあまりとりあげられていない現場のツール。その現状を特集してみます。

- quilt
- darcs
- git

Debian のソース管理

debian/changelog に変更履歴が記述されることになっている。
.dsc のバージョン毎の変更履歴は debdiff コマンドで確認できる。
ただし、各リリース単位でしか変更履歴の記録が存在しない。

```
$ ls -1 *dsc *.diff.gz *.orig.tar.gz
refit_0.7-1.diff.gz
refit_0.7-1.dsc
refit_0.7-2.diff.gz
refit_0.7-2.dsc
refit_0.7-3.diff.gz
refit_0.7-3.dsc
refit_0.7.orig.tar.gz
refit_0.8-1.diff.gz
refit_0.8-1.dsc
refit_0.8.orig.tar.gz
```

0.7-1, 0.7-2 の差分

```
$ debdiff refit_0.7-[1,2].dsc
diff -u refit-0.7/debian/README.Debian refit-0.7/debian/README.Debian
--- refit-0.7/debian/README.Debian
+++ refit-0.7/debian/README.Debian
@@ -6,4 +6,5 @@
-EFI files are available in /usr/share/refit/, bless refit.efi in a EFI
+EFI files are available in /usr/share/refit/, copy it to somewhere
+accessible from MacOSX, boot into Mac OS X, 'bless' refit.efi in a EFI
or HFSplus partition in order to use it.

- -- Junichi Uekawa <dancer@debian.org>, Sun,  2 Jul 2006 09:19:20 +0900
+ -- Junichi Uekawa <dancer@debian.org>, Sat,  8 Jul 2006 13:40:11 +0900
diff -u refit-0.7/debian/changelog refit-0.7/debian/changelog
--- refit-0.7/debian/changelog
+++ refit-0.7/debian/changelog
@@ -1,3 +1,10 @@
+refit (0.7-2) unstable; urgency=low
+
+ * README.Debian: update a little bit.
+ * remove gptsync.o from UNIX build so that it's not used in EFI build.
+
+ -- Junichi Uekawa <dancer@debian.org> Sat,  8 Jul 2006 13:42:08 +0900
+
refit (0.7-1) unstable; urgency=low

 * Initial release (Closes: #375999)
```


SCM ではないパッチの管理

Debian のソースパッケージ内部にパッチファイルを複数保持し、変更単位に分離して管理するシステム。パッチはパッケージの `diff.gz` に含まれる。

- dpatch
- dbs
- quilt

quilt by 小林さん



darcs by David Smith



git-buildpackage by 上川



git を日常的に使っている人
挙手

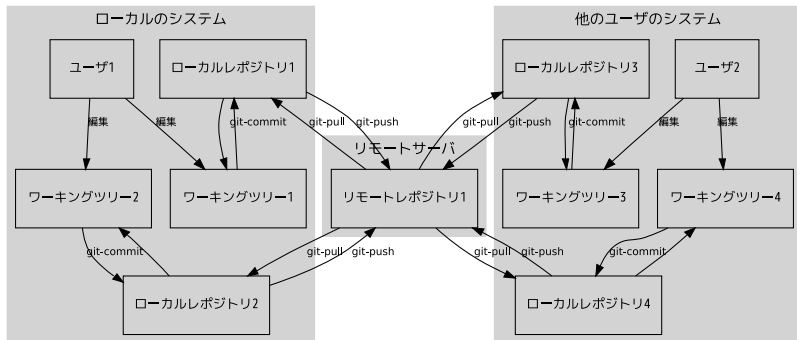


git の使いかたを知らない人
挙手



この会のメンバーは濃すぎるので、もう git の使い方は常識のようなので復習になってしまいますが

git のワークフロー



分散SCM、git の関連用語

用語	定義
ワーキングツリー	SCM で管理されている作業用のディレクトリで、ユーザが直接作業できるようにファイルがある場所
ローカルレポジトリ	SCM で管理されているデータワーキングツリーと同じ場所の.git ディレクトリに実体がある。直接ファイルを編集することはできない
リモートレポジトリ	SCM 管理されているデータで、ネットワーク上のどこかに存在しているもの。しばしば他人と共有している。直接ファイルを編集することはできない
コミット	ワーキングツリーからローカルレポジトリに情報を反映すること
プッシュ	ローカルレポジトリからリモートレポジトリに情報を反映すること
プル	リモートレポジトリからローカルレポジトリとワーキングツリーに情報を反映すること

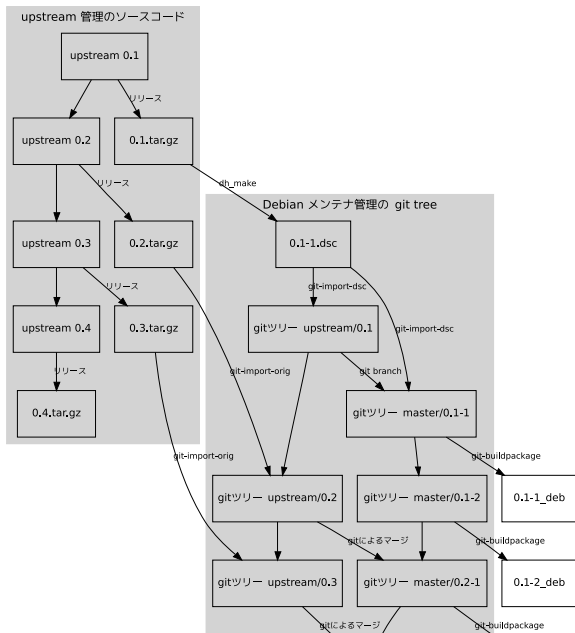
git のよくつかうコマンド

コマンド名	例	意味	cvs 相当
git-clone	<code>git-clone git://XXX/YYY</code>	リモートレポジトリをローカルにクローンし、ワーキングツリーをチェックアウトする	cvs login, cvs co
git-init-db	<code>git-init-db</code>	ローカルレポジトリ (.git ディレクトリ) を作成する	cvs init
git-pull	<code>git-pull git://XXX/YYY</code>	リモートレポジトリの変更をローカルにマージし、ワーキングツリーをアップデートする	cvs up
git-commit	<code>git-commit -a -m 'xxx'</code>	ローカルレポジトリに変更をコミットする	cvs ci の前半
git-push	<code>git-push git://XXX/YYY</code>	ローカルレポジトリをリモートレポジトリに送信する	cvs ci の後半
git-add	<code>git-add filename</code>	ファイルを次回コミットの際にローカルレポジトリに追加されるように登録する	cvs add
git-rm	<code>git-rm filename</code>	ファイルを次回コミットの際にローカルレポジトリから削除されるように登録する	cvs remove
git-status	<code>git-status</code>	ローカルレポジトリに対してワーキングツリーの状況を確認する	cvs status
git-diff	<code>git-diff</code>	ローカルレポジトリとワーキングツリーの差分を表示する	cvs diff

Debian パッケージを管理する際のワークフローをスムーズにするためのツール

- arch-buildpackage - tools for maintaining Debian packages using arch
- cvs-buildpackage - A set of Debian package scripts for CVS source trees.
- darcs-buildpackage - Suite to help with Debian packages in Darcs archives
- git-buildpackage - Suite to help with Debian packages in Git repositories
- hg-buildpackage - Suite to help with Debian packages in Mercurial archives
- svn-buildpackage - helper programs to maintain Debian packages with Subversion
- tla-buildpackage - Suite to help with Debian packages in Arch archives

git-buildpackage



git-buildpackage


- git-import-dsc
- git-import-orig
- git-buildpackage



最初の Debian パッケージをインポートし、Debian ブランチとアップストリームのブランチを作成します

新しいアップストリームバージョンがリリースされた場合にアップストリームブランチにインポートし、Debian ブランチにマージします。

git レポジトリからパッケージをビルドし、必要であればタグを付けます。



利点: git-cherry / git-cherry-pick コマンドなどを利用して、
まだマージされていない変更の一覧を確認できる

欠点: 一度コンフリクトがあるとその変更がマージのコミットとして記録される (quilt / dpatch との違い)

Debian のソースパッケージをみただけではSCMのレポジトリがどこにあるのかがわからない。

質問?



Debian 勉強会の今後の活動

- 5月：Debconf ネタ (superh, pbuilder-qemu) リハーサル?
SCM の活用について? その他との連系について?
- 6月：スコットランドで開催
- 7月：
- 8月：
- 8月：
- 8月：
- 8月：
- 12月：反省会