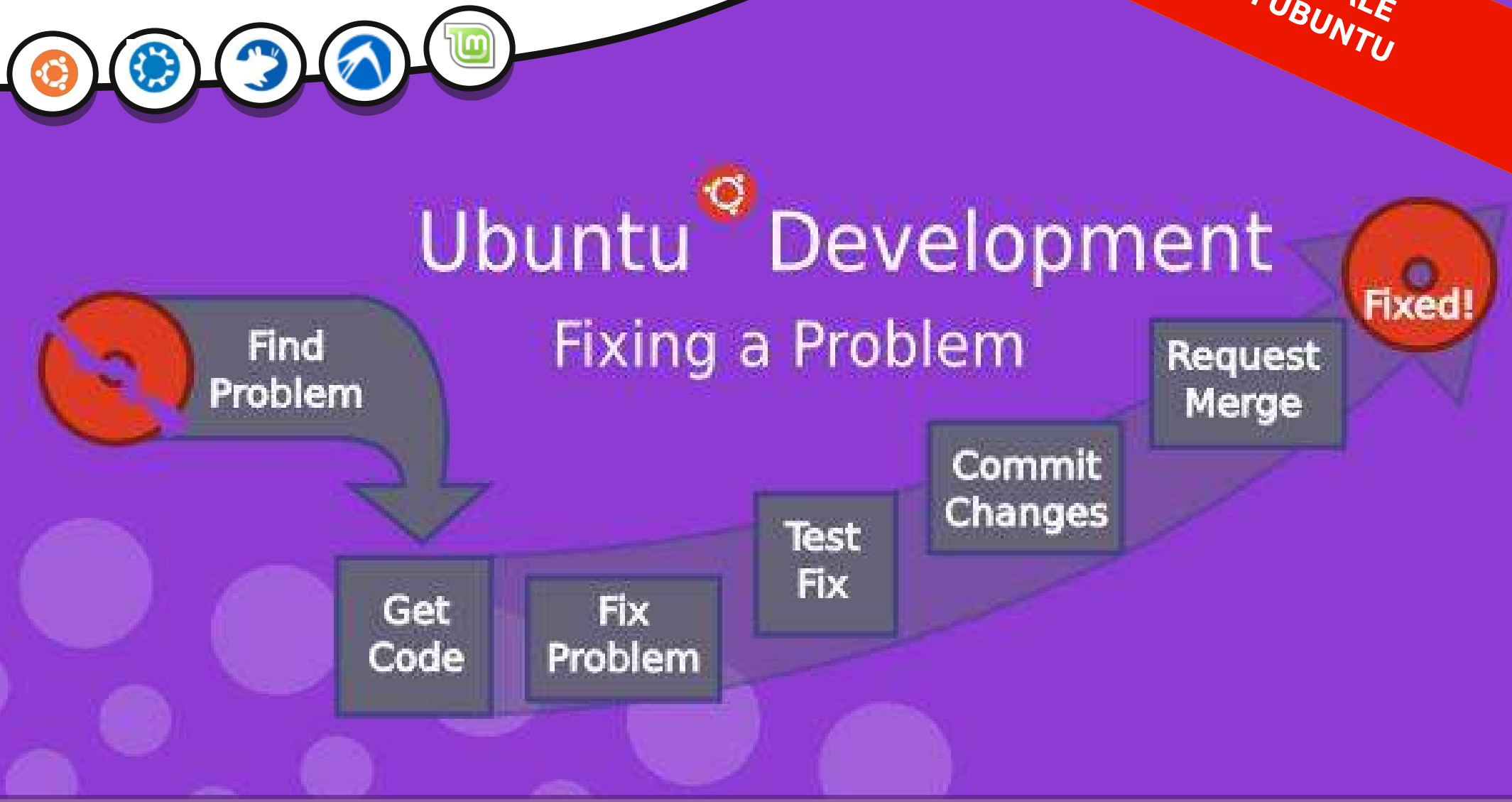
# Full Circle

LA RIVISTA INDIPENDENTE PER LA COMUNITÀ LINUX UBUNTU

EDIZIONE SPECIALE SVILUPPO DI UBUNTU

EDIZIONE SPECIALE
SVILUPPO DI UBUNTU




## ARGOMENTO A SERIE SINGOLA DAI NUMERI 49 - 52

full circle magazine non è affiliata né sostenuta da Canonical Ltd.

**Cos'è Full Circle**
Full Circle è una rivista gratuita e indipendente, dedicata alla famiglia Ubuntu dei sistemi operativi Linux.
Ogni mese pubblica utili articoli tecnici e articoli inviati dai lettori.

Full Circle ha anche un podcast di supporto, il Full Circle Podcast, con gli stessi argomenti della rivista e altre interessanti notizie.

**Si prega di notare che** questa edizione speciale viene fornita senza alcuna garanzia: né chi ha contribuito né la rivista Full Circle hanno alcuna responsabilità circa perdite di dati o danni che possano derivare ai computer o alle apparecchiature dei lettori dall'applicazione di quanto pubblicato.

# Full Circle

LA RIVISTA INDIPENDENTE PER LA COMUNITÀ LINUX UBUNTU

**Ecco a voi un altro Speciale monotematico!**

Come richiesto dai nostri lettori, stiamo assemblando in edizioni dedicate alcuni degli articoli pubblicati in serie.

Quella che avete davanti è la ristampa della serie '**Sviluppare con Ubuntu' parti 1-4, di Daniel Holbach** pubblicata nei numeri 49-52: niente di speciale, giusto quello che abbiamo già pubblicato.

Vi chiediamo, però, di badare alla data di pubblicazione: le versioni attuali di hardware e software potrebbero essere diverse rispetto ad allora.
Controllate il vostro hardware e il vostro software prima di provare quanto descritto nelle guide di queste edizioni speciali. Potreste avere versioni più recenti del software installato o disponibile nei repository delle vostre distribuzioni.

**Buon divertimento!**

**Come contattarci**

**Sito web:**
http://www.fullcirclemagazine.org/

**Forum:**
http://ubuntuforums.org/forumdisplay.php?f=270

**IRC:**#fullcirclemagazine su chat.freenode.net

**Gruppo editoriale**

Capo redattore: Ronnie Tucker
(aka: RonnieTucker)
ronnie@fullcirclemagazine.org

Webmaster: Rob Kerfia
(aka: admin / linuxgeekery-
admin@fullcirclemagazine.org

Podcaster: Robin Catling
(aka RobinCatling)
podcast@fullcirclemagazine.org

Manager delle comunicazioni:
Robert Clipsham
(aka: mrmonday) -
mrmonday@fullcirclemagazine.org

SOME RIGHTS RESERVED

Gli articoli contenuti in questa rivista sono stati rilasciati sotto la licenza Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 3.0. Ciò significa che potete adattare, copiare, distribuire e inviare gli articoli ma solo sotto le seguenti condizioni: dovete attribuire il lavoro all'autore originale in una qualche forma (almeno un nome, un'email o un indirizzo Internet) e a questa rivista col suo nome ("Full Circle Magazine") e con suo indirizzo Internet www.fullcirclemagazine.org (ma non attribuire il/gli articolo/i in alcun modo che lasci intendere che gli autori e la rivista abbiano esplicitamente autorizzato voi o l'uso che fate dell'opera). Se alterate, trasformate o create un'opera su questo lavoro dovete distribuire il lavoro risultante con la stessa licenza o una simile o compatibile.
**Full Circle magazine è completamente indipendente da Canonical, lo sponsor dei progetti di Ubuntu, e i punti di vista e le opinioni espresse nella rivista non sono in alcun modo da attribuire o approvati da Canonical.**

# Introduzione allo sviluppo di Ubuntu

Scritto da Daniel Holbach

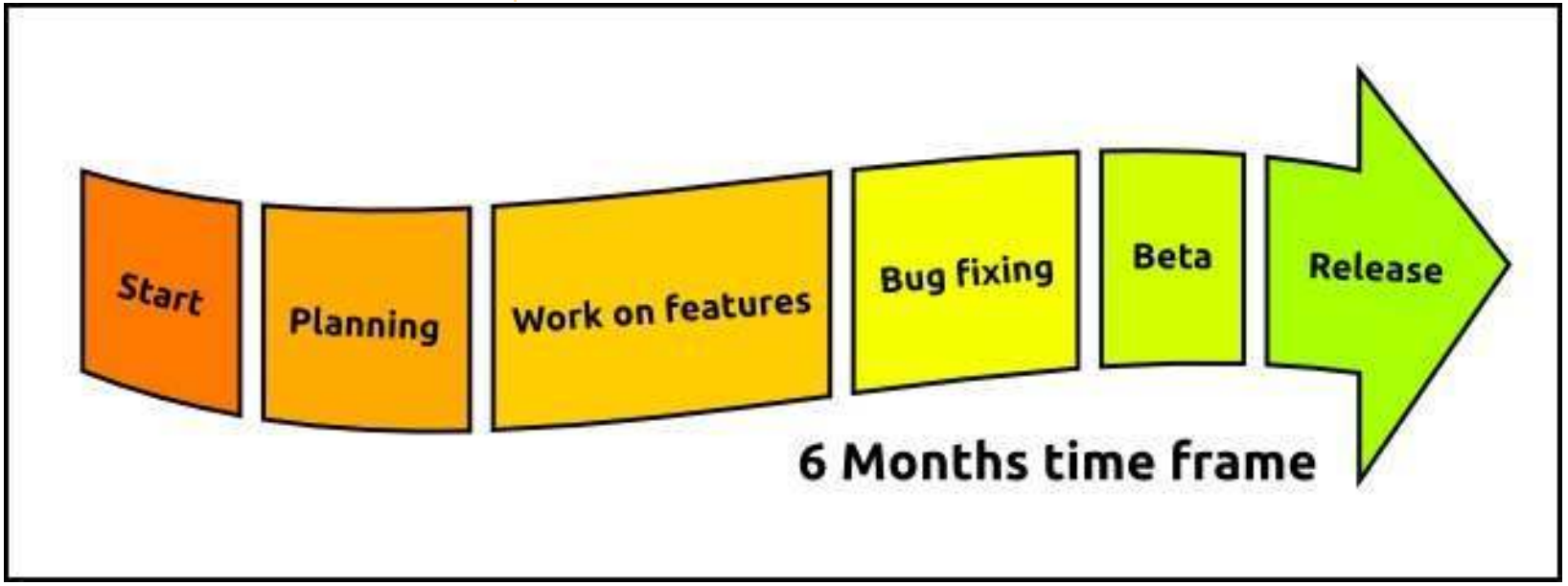
Ubuntu è costituito da migliaia di componenti diversi, scritti in diversi linguaggi di programmazione. Ogni componente, sia esso una libreria software, uno strumento, o un'applicazione grafica, è disponibile come pacchetto sorgente. I pacchetti sorgente nella maggior parte dei casi sono costituiti da due parti: il codice sorgente reale e i metadati. I metadati includono la dipendenze del pacchetto, il copyright, le informazioni sulle licenze e le istruzioni su come compilare il pacchetto. Una volta che questo pacchetto sorgente viene compilato, il processo di compilazione fornisce dei pacchetti binari che sono i file .deb che gli utenti possono installare.

Ogni volta che una nuova versione di un'applicazione è rilasciata o quando qualcuno fa una modifica al codice sorgente che passa in Ubuntu, il pacchetto sorgente deve essere caricato nel computer per essere compilato. I pacchetti binari finali sono poi distribuiti nell'archivio nei vari mirror in diversi paesi. Gli indirizzi URL presenti in /etc/apt/sources.list puntano ad un archivio o un mirror.

Ogni giorno vengono implementate immagini di CD delle differenti derivate di Ubuntu. Ubuntu Desktop, Ubuntu Server, Kubuntu ed altri, specificano un elenco dei pacchetti richiesti da montare nel CD. Queste immagini vengono poi usate per le prove di installazione e quindi fornire un feedback per l'ulteriore pianificazione di rilascio.

Lo sviluppo di Ubuntu dipende moltissimo dalla corrente fase del ciclo di rilascio. Rilasciamo una nuova versione di Ubuntu ogni sei mesi e questo è reso possibile solo perché abbiamo stabilito rigorose date di freeze. Per ogni data di freeze che viene raggiunta, gli sviluppatori sono tenuti ad apportare meno modifiche e devono essere anche meno intrusive. Il Feature Freeze è la prima grande data di freeze dopo la prima metà del ciclo di sviluppo. In questa fase le caratteristiche devono essere in gran parte attuate. Il resto del ciclo dovrebbe essere focalizzato sulla correzione di bug. Dopo questo, l'interfaccia utente, la documentazione, il kernel, ecc, sono congelati e viene messa fuori la versione beta che riceve un sacco di test. Dalla versione beta in poi, vengono fissati solo i bug critici e viene rilasciata la release candidate; se non presenta gravi problemi, diventa la release finale.

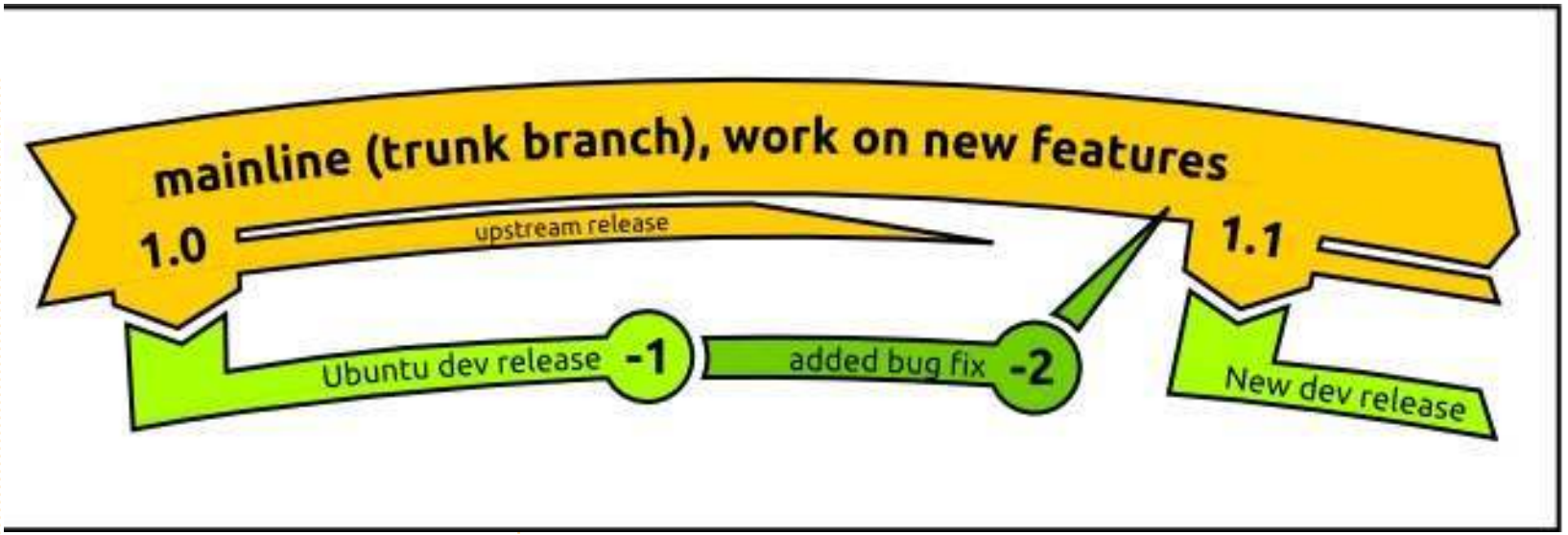
Migliaia di pacchetti sorgenti, miliardi di righe di codice e centinaia di collaboratori, richiedono un sacco di comunicazione e pianificazione per mantenere alti gli standard di qualità. All'inizio di ogni ciclo di rilascio, abbiamo l'Ubuntu Developer Summit dove gli sviluppatori e i collaboratori si riuniscono per pianificare le caratteristiche del prossimo rilascio. Ogni funzione è discussa dalle parti interessate e viene scritta una specifica che contiene informazioni dettagliate



circa la sua ipotesi, l'attuazione, i necessari cambiamenti in altri luoghi, come testarlo e così via. Questo è svolto tutto in maniera aperta e trasparente, quindi anche se non si può partecipare all'evento di persona, è possibile partecipare da remoto e ascoltare uno Streamcast, chattare con gli assistenti, e sottoscrivere cambiamenti delle specifiche, in questo modo si è sempre aggiornati.

Però non tutti i singoli cambiamenti possono essere discussi in una riunione, soprattutto perché Ubuntu si basa sulle modifiche che sono state apportate in altri progetti. Questo è il motivo per il quale i collaboratori in Ubuntu rimangono costantemente in contatto. La maggior parte delle squadre o dei progetti usano mailing list dedicate per evitare confusione estranea. Per un ulteriore coordinamento immediato, sviluppatori e collaboratori fanno uso della Internet Relay Chat (IRC). Tutte le discussioni sono aperte e pubbliche.

Un altro strumento importante per quanto riguarda la comunicazione è il report dei bug. Ogni volta che viene trovato un difetto in un pacchetto o un pezzo di infrastruttura, viene registrata una segnalazione di bug in Launchpad. Tutte le informazioni sono raccolte in




tale relazione compresa la sua importanza, lo stato e il cessionario, aggiornati quando necessario. Questo lo rende uno strumento efficace per monitorare i bug in un pacchetto o progetto e organizzare il carico di lavoro.

La maggior parte del software disponibile attraverso Ubuntu non è scritto dagli sviluppatori stessi di Ubuntu. La maggior parte di essa è scritto dagli sviluppatori di altri progetti Open Source e viene poi integrato in Ubuntu. Questi progetti si chiamano "Upstream", perché il loro codice sorgente fluisce in Ubuntu, dove noi provvediamo "solo" ad integrarlo. Il rapporto verso gli "Upstream" è di fondamentale importanza per Ubuntu. Non è solo codice che dagli "Upstream" fluisce verso Ubuntu, ma da Ubuntu (e altre distribuzioni) verso "Upstream" fluiscono anche gli utenti, le segnalazioni di bug e le patch.

Il più importante Upstream per Ubuntu è Debian. Debian è la distribuzione su cui è basato Ubuntu e molte decisioni di progettazione relative all'infrastruttura del packaging vengono fatte lì. Tradizionalmente, Debian ha avuto sempre dei manutentori dedicati o squadre di manutenzione per ogni singolo pacchetto. In Ubuntu ci sono squadre che hanno un interesse anche per un sottoinsieme di pacchetti e naturalmente ogni sviluppatore ha una speciale area di competenza, ma generalmente la partecipazione (e i diritti di upload) è aperta a tutti coloro che dimostrano capacità e volontà.

Essere in Ubuntu come un collaboratore nuovo non è così scoraggiante come sembra e può essere un'esperienza molto gratificante. Non è solamente imparare qualcosa di nuovo ed eccitante, ma anche una condivisione della soluzione e la risoluzione di un problema per milioni di utenti là fuori.

Lo sviluppo Open Source funziona in un ambiente distribuito con obiettivi diversi e diverse aree di interesse. Per esempio, ci potrebbe essere il caso che un particolare Upstream possa essere interessato a lavorare ad una nuova grande caratteristica, mentre Ubuntu, a causa della stretta pianificazione del

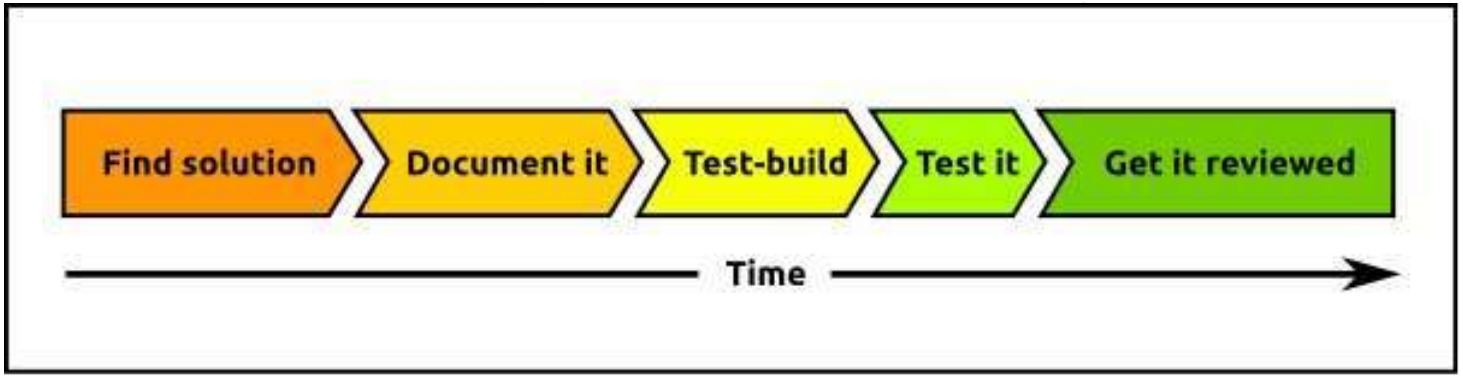
rilascio, potrebbe essere interessato a lavorare alla distribuzione della versione solo con un ulteriore bug fix. È per questo che facciamo uso di "Sviluppo distribuito", dove il codice è stato lavorato in vari rami che si fondono insieme, dopo le dovute revisioni del codice e sufficienti discussioni.

Nell'esempio citato sopra, avrebbe senso spedire Ubuntu con la versione esistente del progetto, aggiungere i bugfix, passarlo Upstream per il loro prossimo rilascio e rispedirlo (se adatto) nella prossima release di Ubuntu. Sarebbe il miglior compromesso e una situazione in cui tutti vincono.

Per risolvere un bug di Ubuntu si deve prima ottenere il codice sorgente per il pacchetto, poi lavorare sulla correzione, documentarla affinché sia facile da capire per gli altri sviluppatori e gli utenti, quindi creare il pacchetto per testarlo. Dopo averlo provato, si può facilmente proporre il cambiamento da inserire nell'attuale versione di rilascio in sviluppo di Ubuntu. Uno sviluppatore con diritto di upload lo recensirà per voi e poi potrebbe essere integrato in Ubuntu.

Quando si cerca di trovare una soluzione di solito è una buona idea controllare con Upstream e vedere se il problema (o una possibile soluzione) è già noto e, in caso contrario, fare il vostro meglio per rendere la soluzione uno sforzo condiviso. Un ulteriore passo potrebbe essere quello di prelevare la modifica applicata a una vecchia, ma ancora supportata, versione di Ubuntu e di inoltrarla Upstream.

I più importanti requisiti per il successo nello sviluppo di Ubuntu e l'avere talento per "far si che le cose funzionino ancora una volta", non avere paura a leggere la documentazione e fare domande, visto che si è un giocatore in una squadra, e godere di alcuni lavori da detective. Ottimi posti per fare le vostre domande sono ubuntu-motu-mentors@lists.ubuntu.com e **#ubuntu-motu su irc.freenode.net**. Potrai facilmente trovare un sacco di nuovi amici e persone con la tua stessa passione: fare il mondo un posto migliore rendendo migliore il software Open Source.




## UN APPELLO A FAVORE DEL GRUPPO PODCAST

Come avrete sentito nell'episodio #15 del podcast, stiamo cercando argomenti per questa sezione della rivista.

Invece di lasciar parlare noi a ruota libera su qualsiasi cosa ci colpisca, perché non ci proponete un argomento e guardate l'esplosione nucleare che ne deriva all'orizzonte? È altamente improbabile che tre di noi siano d'accordo.
Oppure, un pensiero ancora più radicale, inviaci un parere attraverso un contributo!

Puoi postare commenti e opinioni nella pagina del podcast su fullcirclemagazine.org, nella nostra sezione Ubuntu Forums e via mail a podcast@fullcirclemagazine.org. Puoi mandarci anche un commento registrando una clip audio di non più di 30 secondi e inviandola allo stesso indirizzo.

**I commenti e l'audio possono essere modificati per la lunghezza. Per favore ricorda che questa è una rivista a conduzione familiare.**

Sarebbe una gran cosa avere collaboratori che vengano in redazione ed esprimano un'opinione di persona.

Robin

# Lo sviluppo di Ubuntu Parte 2- Preparativi

Scritto da Daniel Holbach

Ci sono un certo numero di cose che dovete fare per iniziare a sviluppare per Ubuntu. Questo articolo è stato sviluppato per impostare il vostro computer in modo che possiate iniziare a lavorare con i pacchetti e caricare i vostri pacchetti su Launchpad. Ecco cosa vedremo:

• Installazione di software relativo al packaging. Questo include:
  • utility specifiche di Ubuntu per il packaging
  • Software di crittografia cosicché il tuo lavoro potrà essere verificato come fatto da voi
  • software di crittografia aggiuntivo in modo da poter trasferire file in modo sicuro
• Creazione e configurazione del tuo account su Launchpad
• Impostazione dell'ambiente di sviluppo per aiutarti a creare dei pacchetti in locale, interagire con altri sviluppatori e proporre le modifiche su Launchpad.

Nota: Si consiglia di fare un lavoro di pacchettizzazione utilizzando l'attuale versione di sviluppo di Ubuntu. Questo consentirà di verificare i cambiamenti nello stesso ambiente in cui tali modifiche saranno effettivamente applicate e utilizzate.

Non preoccupatevi però, la pagina wiki della release di sviluppo di Ubuntu (https://wiki.ubuntu.com/UsingDevelopmentReleases) mostra una varietà di modi per utilizzare in modo sicuro la release in sviluppo.

## Installare il software di base per il packaging

Ci sono una serie di strumenti che renderanno la vostra vita da sviluppatore di Ubuntu molto più facile. Incontreremo questi strumenti più avanti in questa guida. Per installare la maggior parte degli strumenti necessari, eseguire questo comando:

```
sudo apt-get install gnupg pbuilder ubuntu-dev-tools bzr-builddeb apt-file
```

Questo comando installerà i seguenti software:

**gnupg** - GNU Privacy Guard contiene strumenti necessari per creare una chiave crittografica con cui si firmeranno i file che si desidera caricare su Launchpad.

**pbuilder** - uno strumento per compilare un pacchetto riproducibile in un ambiente pulito e isolato.

**ubuntu-dev-tools** (e devscripts, una dipendenza diretta) - una raccolta di strumenti che rendono le attività di packaging molto più facile.

**bzr-builddeb** (e bzr, una dipendenza) - strumenti per il controllo distribuito della versione che rendono facile la collaborazione ed il lavoro per molti sviluppatori sullo stesso codice permettendo una facile fusione dei rispettivi lavori.

**apt-file** fornisce un modo semplice per trovare il pacchetto binario che contiene un dato file.

**apt-cache** (parte del pacchetto apt) fornisce ancora più informazioni ancora sui pacchetti su Ubuntu.

## Crea la tua chiave GPG

GPG sta per GNU Privacy Guard e implementa lo standard OpenPGP che permette di firmare e crittografare i messaggi e file. Questo è utile per diversi scopi. Nel nostro caso è importante che sia possibile firmare i file con la chiave in modo che possano essere identificati come qualcosa a cui hai lavorato. Se si carica un pacchetto sorgente in Launchpad, il pacchetto verrà accettato solo se si può assolutamente determinare chi ha caricato il pacchetto.

Per generare una nuova chiave GPG, eseguire:

```
gpg - gen-key
```

GPG in primo luogo chiede che tipo di chiave si vuol generare. La scelta predefinita (RSA e DSA) va bene. Accanto vi chiederà informazioni riguardo la dimensione della chiave. Il valore predefinito (attualmente 2048) va bene, ma 4096 è più sicuro. Successivamente, vi verrà chiesto se si desidera impostare una scadenza per la chiave. Un metodo sicuro è l'opzione "0", ciò significa che la chiave non scadrà mai. Le ultime domande riguarderanno il vostro

nome e indirizzo email. Basta scegliere quelle che si intende utilizzare per lo sviluppo di Ubuntu e sarà possibile aggiungere ulteriori indirizzi di posta elettronica in seguito. L'aggiunta di un commento non è necessario. A questo punto si dovrà impostare una passphrase. Sceglietene una sicura.

Ora GPG creerà per voi una chiave; il processo può richiedere un po' di tempo perché l'applicazione ha bisogno di byte casuali, quindi se date al vostro sistema del lavoro da fare, sarà bene. Muovete il cursore sullo schermo!

Una volta fatto questo, si otterrà un messaggio simile al seguente:

```
pub 4096R/43CDE61D 2010/12/06
Key fingerprint = 5C28 FB08
91C0 0144 2CF3 37AC 6F0B F90F
43CD E61D
uid Daniel Holbach
<dh@mailempfang.de>
sub 4096R/51FBE68C 2010/12/06
```

In questo caso 43CDE61D è l'ID della chiave.

Successivamente, è necessario caricare la parte pubblica della vostra chiave a un server di chiavi in modo che il mondo può identificare come di tua provenienza i tuoi messaggi e i tuoi file. Per farlo, digitate:

```
gpg - send-keys <ID <key
```

Questo invierà la vostra chiave a un server di chiavi, ma automaticamente una rete di keyserver sincronizzeranno tra di loro la chiave. Una volta che questa procedura è terminata, la chiave pubblica firmata sarà pronta a verificare i vostri contributi in tutto il mondo.

## Crea la tua chiave SSH

SSH è l'acronimo di Secure Shell ed è un protocollo che consente di scambiare dati in modo sicuro in rete. È comune l'uso di SSH per accedere e aprire una shell su un altro computer e utilizzarla per il trasferimento sicuro dei file. Per i nostri scopi, useremo principalmente SSH per comunicare in sicurezza con Launchpad.

Per generare una chiave SSH, digitare:

```
ssh-keygen-t rsa
```

Il nome del file predefinito di solito va bene, quindi si può semplicemente lasciare così com'è. Per motivi di sicurezza è altamente raccomando l'utilizzo di una passphrase.

## Impostare pbuilder

Pbuilder consente di creare pacchetti a livello locale sulla vostra macchina. Il suo uso serve ad un paio di scopi:

- La compilazione sarà effettuata in un ambiente minimale e pulito. Questo aiuta a fare in modo che i file compilati siano riproducibili, ma senza modificare il sistema locale.
- Non vi è alcun bisogno di installare tutte le dipendenze necessarie a livello locale.
- È possibile impostare più istanze per i vari rilasci di Debian e Ubuntu.

Impostare pbuilder è molto facile. Aprire con un editor di testo ~/.pbuilderrc e aggiungere la seguente riga:

```
COMPONENTS="main universe
multiverse restricted"
```

Questo farà sì che le dipendenze di compilazione siano soddisfatte con tutti i componenti. Quindi eseguire:

```
pbuilder-dist <release> create
```

dove <release> è, ad esempio, natty, maverick, lucid, o, nel caso di Debian, potrebbe essere sid. Questo richiederà un po' perché dovrà scaricare tutti i pacchetti necessari per una "installazione minima". Questi però saranno memorizzati nella cache.

## Impostazioni per lavorare con Launchpad

Con una configurazione locale di base, il prossimo passo sarà quello di configurare il sistema per lavorare con Launchpad. Questa sezione si concentrerà sui seguenti argomenti:

- Cos'è Launchpad e come creare un account su Launchpad
- Caricare la tua chiave GPG ed SSH su Launchpad
- Configurazione di Bazaar per lavorare con Launchpad
- Configurazione di Bash per lavorare con Bazaar

## Informazioni su Launchpad

Launchpad è il pezzo centrale

delle infrastrutture che utilizziamo in Ubuntu. Non solo archivia i nostri pacchetti e il nostro codice, ma anche cose come traduzioni, le segnalazioni e le informazioni sulle persone che lavorano su Ubuntu e la loro squadra di appartenenza. Potrete anche usare Launchpad per pubblicare le correzioni proposte e permettere ad altri sviluppatori di Ubuntu di revisionarle e sponsorizzarle.

Avrete bisogno di registrarvi su Launchpad e di fornire una quantità minima di informazioni. Questo vi permetterà di scaricare e caricare il codice, inviare segnalazioni di bug, e altro ancora.

## Crea un account Launchpad

Se non si dispone già di un account su Launchpad, è possibile crearne facilmente uno (all'indirizzo: https://launchpad.net/+login). Se si ha un account su Launchpad ma non si ricorda il proprio id Launchpad, è possibile scoprirlo andando su https://launchpad.net/people/+me, controllare la parte dopo la ~ nell'URL.

Il processo di registrazione Launchpad vi chiederà di scegliere un nome visualizzato. Si incoraggia l'uso del nome reale in modo che i colleghi sviluppatori di Ubuntu saranno in grado di conoscervi meglio.

Quando si registra un nuovo account, Launchpad invierà una mail con un link che è necessario aprire nel browser al fine di verificare l'indirizzo email. Se non la si riceve, controllare nella cartella spam.

La nuova pagina di aiuto per l'account (https://help.launchpad.net/YourAccount/NewAccount) su Launchpad ha più informazioni sul processo e le impostazioni aggiuntive che si possono cambiare.

## Carica la tua chiave GPG per Launchpad

Per conoscere la propria impronta digitale GPG, eseguire:

```
gpg --fingerprint
<email@address.com>
```

e verrà visualizzato qualcosa del tipo:

```
pub 4096R/43CDE61D 2010/12/06
Key fingerprint = 5C28 FB08
91C0 0144 2CF3 37AC 6F0B F90F
43CD E61D
uid Daniel Holbach
<dh@mailempfang.de>
sub 4096R/51FBE68C 2010/12/06
```

Si vada quindi su https://launchpad.net/people/+me/+editpgpkeys e copiare la parte che riguarda la vostra "impronta digitale" nella casella di testo. Nel caso di sopra, sarebbe 5C28 FB08 91C0 0144 2CF3 37AC 6F0B F90F 43CD E61D. Ora fate clic su "Import Key".

Launchpad utilizzerà l'impronta digitale per controllare il server delle chiavi di Ubuntu per recuperare la vostra chiave e, in caso di successo, inviare una e-mail criptata che chiede di confermare l'importazione delle chiavi. Controllare il proprio account di posta elettronica e leggere le e-mail che Launchpad ha inviato. Se il vostro client di posta elettronica supporta la crittografia OpenPGP, vi verrà chiesto di inserire la password scelta in fase di generazione della chiave GPG. Immettere la password, quindi fare clic sul link per confermare che la chiave sià la vostra.

Launchpad cripta l'e-mail, utilizzando la chiave pubblica, in modo che si possa essere sicuri che la chiave è la vostra. Se il vostro software di posta elettronica non supporta la crittografia OpenPGP, copiare il contenuto della e-mail criptata, digitare nel vostro terminale gpg, quindi incollare il contenuto dell'e-mail nella finestra del terminale.

Ritornando sul sito di Launchpad, utilizzare il pulsante "Conferma" e Launchpad completerà l'importazione della chiave OpenPGP.

Per ulteriori informazioni: https://help.launchpad.net/YourAccount/ImportingYourPGPKey

## Caricare la propria chiave SSH su Launchpad

Aprite la pagina https://launchpad.net/people/+me/+editsshkeys in un browser Web, nonché le pagine ~/.ssh /id_rsa.pub in un editor di testo. Questa è la parte pubblica della vostra chiave SSH, quindi la condivisione in Launchpad è sicura. Copiare il contenuto del file e incollarlo nella casella di testo nella pagina web che dice "Add an SSH key". Ora fate clic su "Import Public Key".

Per ulteriori informazioni su questo processo, visitare la pagina relativa alla creazione di una coppia di chiavi SSH (https://help.launchpad.net/YourAccount/CreatingAnSSHKeyPair) su Launchpad.

## Configurare Bazaar

Bazar è lo strumento che utilizziamo per memorizzare le modifiche al codice in modo logico, per scambiarsi proposte di modifiche e unirle, anche se lo sviluppo è fatto contemporaneamente. Per far sapere a Bazaar chi sei, basta eseguire:

```
bzr whoami "Bob Dobbs <subgenius@example.com>"
```

```
bzr launchpad-login SubGenio
```

whoami dirà a Bazar il nome e l'indirizzo di posta elettronica che dovrà utilizzare per i messaggi di commit. Con launchpad-login si imposta l'ID Launchpad. In questo modo il codice che si pubblica in Launchpad sarà associato a voi.

Nota: Se non riuscite a ricordare l'ID, andare su https://launchpad.net/people/+me e vedere dove si viene reindirizzati. La parte dopo il "~" nell'URL è il vostro ID Launchpad.)

## Configura la tua shell

Come Bazar, anche gli strumenti di packaging per Debian/Ubuntu hanno necessità di conoscervi. È sufficiente aprire il file ~/.bashrc in un editor di testo e aggiungere qualcosa di simile in fondo:

```
export DEBFULLNAME="Bob Dobbs"

export DEBEMAIL="subgenius@example.com"
```

Ora salvate il file e riavviate il terminale o eseguite:

```
source ~/.bashrc
```

(Se si utilizza una shell diversa da quella predefinita (che è bash), per favore editate il file di configurazione relativa a quella shell.)

**IL PROSSIMO MESE**: Correzione di un bug

## UN APPELLO A NOME DELLA FESTA DEL PODCAST

Come sentirete nella puntata #15 del podcast, chiederemo opinioni tematiche per quella sezione dello spettacolo.

Piuttosto che aspettare che noi sbraitiamo su qualsiasi cosa ci passi per la testa, perché non ispirarci con un argomento e prestare poi attenzione al fungo atomico oltre l'orizzonte! È altamente improbabile che tutti e tre noi siamo d'accordo.

Oppure, un pensiero ancora più radicale, inviateci un opinione per mezzo di un contributo.

Potete pubblicare commenti e opinioni sulla pagina del podcast presso fullcirclemagazine.org, nella nostra sezione Forum di Ubuntu, o scrivere a podcast@fullcirclemagazine.org. Potete inoltre inviarci un commento tramite registrazione di una sequenza audio di non più di 30 secondi e inviarla allo stesso indirizzo. **Commenti e audio potranno essere modificati per la loro lunghezza. Ricordatevi che questo è uno spettacolo per le famiglie.**

Sarebbe bello avere collaboratori che intervengono allo spettacolo ed esprimano un parere di persona.

**Robin**

# Sviluppo di Ubuntu, parte 3 - Correzione dei Bug

Scritto da Daniel Holbach

Se avete seguito le istruzioni per ottenere i preparativi per lo sviluppo di Ubuntu, dovresti avere tutto a posto e pronto a partire.

Come si può vedere nell'immagine qui a destra, non ci sono sorprese in fase di correzione dei bug in Ubuntu: si trova un problema, si ottiene il codice, si lavora alla correzione, si testa, si caricano le modifiche in Launchpad e si chiede che vengano revisionate e unite. In questa guida eseguiremo tutti i passi necessari uno ad uno.

## Trovare il problema

Ci sono un sacco di modi diversi per trovare le cose su cui lavorare. Potrebbe essere un bug report che avete incontrato voi stessi (che vi dà una buona occasione per testare la correzione), o un problema notato altrove, magari in un altro bug report.

Harvest è dove teniamo traccia delle varie liste TODO che riguardano lo sviluppo di Ubuntu. Esso elenca i bug che sono stati risolti in Upstream o già in Debian, elenca piccoli bug (li chiamiamo 'Bitesize'), e così via. Dategli un'occhiata e trovate il vostro primo bug su cui lavorare.


## Capire cosa sistemare

Se non si conosce il pacchetto sorgente contenente il codice che presenta il problema, ma si conosce il percorso del programma interessato nel proprio sistema, è possibile scoprire il pacchetto sorgente che dovrete lavorare.

Diciamo che hai trovato un bug in Tomboy, una nota applicazione per il desktop. L'applicazione Tomboy può essere avviata eseguendo /usr/bin/tomboy sulla riga di comando. Per trovare il pacchetto binario che contiene questa applicazione, utilizzare questo comando:

```
apt-file find /usr/bin/tomboy
```

Il comando mostrerà:

```
tomboy: /usr/bin/tomboy
```

Si noti che la parte che precede i due punti è il nome del pacchetto binario. Capita spesso che il pacchetto di origine e il pacchetto binario abbiano nomi diversi. Questo è molto comune quando un pacchetto sorgente singolo è usato per costruire più pacchetti binari diversi. Per trovare il pacchetto sorgente per un particolare pacchetto binario, digitare:

```
apt-cache show tomboy | grep ^Source:
```

In questo caso, non viene stampato nulla, nel senso che tomboy è anche il nome del pacchetto binario. Un esempio in cui i nomi dei pacchetti sorgente e binario sono diversi è python-vigra. Mentre questo è il nome del pacchetto binario, il pacchetto sorgente è in realtà libvigraimpex e può essere trovato con questo comando (e il suo risultato):

```
apt-cache show python-vigra | grep ^Source:
```

```
Source: libvigraimpex
```

## Ottenere il codice

Quando si conosce il pacchetto sorgente su cui lavorare, si desidera ottenere una copia del codice sul sistema, in modo che si possa eseguire il debug. Questo è fatto 'ramificando' il ramo del pacchetto sorgente corrispondente dal pacchetto sorgente. Launchpad mantiene rami del pacchetto sorgente per tutti i pacchetti in Ubuntu. Una volta che si ottiene un pezzo di sorgente in locale, è possibile analizzare i bug, creare una correzione e caricare il fix proposto di Launchpad, sotto forma di un ramo Bazaar. Quando si è soddisfatti della correzione, è possibile presentare una 'proposta di unione', che chiederà ad altri sviluppatori di Ubuntu di esaminare e approvare la modifica. Se sono d'accordo con le modifiche apportate, uno sviluppatore di Ubuntu caricherà la nuova versione del pacchetto per Ubuntu in modo che tutti possano ottenere benefici dalla vostra eccellente correzione e si ottiene un po' di credito. Ora siete sulla buona strada per diventare uno sviluppatore di Ubuntu! Adesso, nelle sezioni che seguono, verranno descritte le specifiche su come ramificare il codice, fornire il proprio fix e chiedere una revisione.

## Lavorare su una correzione

Ci sono interi libri scritti sulla ricerca di bug, sulla correzione, sul test, ecc. Se siete completamente nuovi alla programmazione, cercate di correggere prima i bug più facili, come ovviamente gli errori di battitura. Cercate di mantenere le modifiche in forma minima e documentate le modifiche e le ipotesi in modo chiaro.

Prima di lavorare da solo su una correzione, assicuratevi di indagare se nessun altro ne ha già prodotta una o sta attualmente lavorando su una correzione. Buone fonti da controllare sono:

• Upstream (e Debian) bug tracker (bug aperti e chiusi),

• nella cronologia delle revisioni Upstream (o le release più recenti) potrebbero avere risolto il problema,

• bug o caricamenti di Debian o altre distribuzioni.

Se si trova una patch per risolvere il problema, diciamo, collegato a un bug report, eseguire questo comando nella directory della sorgente che dovrebbe applicare la patch:

```
patch -p1 <../bugfix.patch
```

Fare riferimento alla patch (1) del manpage per le opzioni e gli argomenti come -dry-run,-p <num>, ecc

## La correzione di test

Per costruire una pacchettizzazione per i test con le modifiche, eseguire questi comandi:

```
bzr bd -- -S -us -uc
```

```
pbuilder-dist <release> build
.. /<pacchetto>_<versione>.dsc
```

Questo creerà un pacchetto sorgente dal contenuto ramo (-us-uc serve solo ad omettere il passo della firma del pacchetto di origine) e pbuilder-dist costruirà il pacchetto dai sorgenti per qualsiasi versione scelta.

Una volta che la pacchettizzazione riesce, installate il pacchetto da ~/pbuilder/<release>_result/ (usando sudo dpkg-i <package>_<versione>. Deb). Quindi verificare se il bug è stato risolto.

## Documentare la correzione

È molto importante documentare il cambiamento in modo sufficiente affinché gli sviluppatori che guarderanno il codice in futuro non dovranno indovinare quale è stato il vostro ragionamento e quali fossero i vostri presupposti. Ogni pacchetto sorgente di Debian e di Ubuntu include il Debian/changelog, in cui vengono tracciati i cambiamenti di ogni pacchetto caricato.

Il modo più semplice di aggiornare il changelog è quello di eseguire:

```
dch-i
```

Questo aggiungerà una voce nella copia del changelog per voi e lancerà un editor dove sarà possibile riempire gli spazi vuoti. Un esempio di questo potrebbe essere:

```
specialpackage (1.2-3ubuntu4)
natty; urgency = low
  * Debian/control: description
to include frobnicator (LP:
#123456)
 - Emma Adams
<emma.adams@isp.com> Sat, 17
lug 2010 02:53:39 0200
```

dch dovrebbe già compilare la prima e l'ultima linea di tale voce del changelog per voi. La linea 1 è costituita dal nome del pacchetto sorgente, il numero di versione, in quale release di Ubuntu è caricato, l'urgenza (che quasi sempre è 'low'). L'ultima riga contiene sempre il nome, l'indirizzo email e la data (in formato

RFC 5322), del cambiamento.

Tolta quella di mezzo, concentriamoci sul vero changelog stesso: è molto importante documentare:

• dove è stato fatto il cambiamento
• cosa è stato cambiato
• dove è avvenuta la discussione sul cambiamento

Nel nostro (molto scarso) esempio, l'ultimo punto è definito da (LP: #123456) che si riferisce a Launchpad bug 123456. Segnalazioni di bug, le discussioni in mailing list o le caratteristiche tecniche sono di solito una buona informazione razionale da fornire per un cambiamento. Come bonus, se si utilizza il metodo LP: #notazione<numero> Launchpad per i bug, il bug sarà automaticamente chiuso quando il pacchetto viene caricato in Ubuntu.

## Consegnare la correzione

Con la voce del changelog scritto e salvato, si può semplicemente eseguire:

```
debcommit
```

e il cambiamento sarà consegnato (localmente) con la voce del changelog, come un messaggio di commit.

Per inviarlo su Launchpad, come il nome del ramo, è necessario attenersi al seguente nomenclatura:

```
lp:
~<yourlpid>/ubuntu/<release>/<p
acchetto>/<branchname>
```

Questo potrebbe, ad esempio, essere

```
lp:
~emmaadams/ubuntu/natty/special
package/fix-for-123456
```

Quindi, se eseguite soltanto

```
bzr push
lp:
~emmaadams/ubuntu/natty/special
package/fix-for-123456
```

```
bzr lp-open
```

dovrebbe essere tutto a posto. Il comando push dovrebbe inviarlo a Launchpad e il secondo comando aprirà la pagina del ramo in questione nel vostro browser. Lì, trovate il link "(+) Proporre per la fusione" e fate clic per richiedere la recensione del cambiamento da qualcuno e inserirlo in Ubuntu.

Il prossimo mese: una panoramica della struttura della directory di Debian.


Below Zero

Zero Downtime

**Below Zero** is a Co-located Server Hosting specialist in the UK.

Uniquely we only provide rack space and bandwidth. This makes our service more reliable, more flexible, more focused and more competitively priced. We concentrate solely on the hosting of Co-located Servers and their associated systems, within Scotland's Data Centres.

At the heart of our networking infrastructure is state-of-the-art BGP4 routing that offers optimal data delivery and automatic multihomed failover between our outstanding providers. Customers may rest assured that we only use the highest quality of bandwidth; our policy is to pay more for the best of breed providers and because we buy in bulk this doesn't impact our extremely competitive pricing.

**At Below Zero we help you to achieve Zero Downtime.**

www.zerodowntime.co.uk

# Lo Sviluppo di Ubuntu Parte 4 - debian/

Scritto da Daniel Holbach

In questo articolo spiegherò brevemente i diversi file importanti per la pacchettizzazione in Ubuntu contenuti nella directory debian/. I più importanti sono il changelog, il controllo, il diritto d'autore, e le regole. Questi sono necessari per tutti i pacchetti. Un certo numero di altri file in debian/ possono essere utilizzati al fine di personalizzare e configurare il comportamento del pacchetto stesso. Alcuni di questi file sono discussi in questo articolo, ma questo non vuole essere un elenco completo.

## Il Changelog

Questo file è, come suggerisce il nome, un elenco delle modifiche apportate in ogni versione. Ha un formato specifico che dà il nome del pacchetto, versione, distribuzione, cambiamenti, e chi ha apportato le modifiche in un dato momento. Se si dispone di una chiave GPG (vedi: Come Impostare) assicuratevi di utilizzare lo stesso nome e indirizzo di posta elettronica nel changelog, come riportato nella vostra chiave.

Quello che segue è un modello di un changelog:

```
package (versione)
distribution; urgency=urgency

 * dettagli delle modifiche
  - più dettagli
 * ulteriori dettagli

-- nome del manutentore
<indirizzo>[due spazi]  data
```

Il formato (soprattutto la data) è importante. La data deve essere in formato RFC 5322, che può essere ottenuto utilizzando il comando date-R. Per comodità, il comando dch può essere utilizzato per modificare il changelog. Aggiornerà la data automaticamente. I punti minori nell'elenco puntato sono indicati da un trattino "-", mentre i punti principali utilizzano un asterisco "*". Se partite a pacchettizzare da zero, dch--create (DCH è nel pacchetto devscripts) crea un debian/ changelog standard al posto vostro.

Ecco un esempio di file changelog per 'hello':

```
hello (2.6-0ubuntu1) natty;
urgency=low

  * New upstream release with
lots of bug fixes and feature
improvements.

-- Jane Doe
<packager@example.com>  Thu,
21 Apr 2011 11:12:00 -0400
```

Si noti che la versione aggiunta al changelog è la -0ubuntu1 è la revisione della distribuzione, utilizzata in modo che il pacchetto possa essere aggiornato (per correggere i bug, per esempio) con i nuovi arrivi all'interno della stessa versione di rilascio dei sorgenti.

Ubuntu e Debian hanno degli schemi di versione del pacchetto leggermente diversi per evitare che gli stessi pacchetti possano andare in conflitto con la stessa versione sorgente. Se un pacchetto Debian è stato cambiato in Ubuntu, avrà la dicitura ubuntuX (dove X è il numero di revisione Ubuntu) aggiunto alla fine della versione Debian. Quindi, se il pacchetto Debian hello 2,6-1 è stato cambiato da Ubuntu, la stringa della versione diverrà 2,6-1ubuntu1. Se un pacchetto per l'applicazione non esiste in Debian, allora la revisione Debian è 0 (es. 2,6-0ubuntu1).

Per ulteriori informazioni, vedere la sezione changelog (punto 4.4) del Debian Policy Manual.

## Il file di controllo

Il file di controllo contiene le informazioni che il gestore di pacchetti (come apt-get, synaptic e adept) utilizza, le dipendenze build-time, le informazioni del manutentore e molto altro ancora.

Per il pacchetto di Ubuntu hello, il file di controllo sarà simile a:

```
Source: hello
Section: devel
Priority: optional
Maintainer: Ubuntu Developers
<ubuntu-devel-
discuss@lists.ubuntu.com>
XSBC-Original-Maintainer: Jane
Doe <packager@example.com>
Standards-Version: 3.9.1
Build-Depends: debhelper (>=
7)
Bzr-Vcs: lp:ubuntu/hello
Homepage:
http://www.gnu.org/software/he
llo/

Package: hello
Architecture: any
```

```
Depends: ${shlibs:Depends}
Description: The classic
greeting, and a good example
The GNU hello program produces
a familiar, friendly greeting.
It allows non-programmers to
use a classic computer science
tool which would otherwise be
unavailable to them.
Seriously, though: this is an
example of how to do a Debian
package. It is the Debian
version of the GNU Project's
`hello world' program (which
is itself an example for the
GNU Project).
```

Il primo paragrafo descrive il pacchetto sorgente compreso l'elenco dei pacchetti richiesti per creare il pacchetto dai sorgenti nel campo Build-Depends. Esso contiene inoltre alcune meta-informazioni come il nome del manutentore, la versione della Debian Policy a cui il pacchetto è conforme, la posizione del repository di controllo della versione di pacchettizzazione, e la home page Upstream.

Si noti che, in Ubuntu, abbiamo impostato il campo Maintainer con un indirizzo generale, perché chiunque può modificare qualsiasi pacchetto (questa cosa si differenzia da Debian dove i cambiamenti ai pacchetti sono ristretti a un individuo o un gruppo). I pacchetti in Ubuntu dovrebbero in genere avere il campo Maintainer impostato su Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>. Se il campo Maintainer viene modificato, il vecchio valore deve essere salvato nel campo XSBC-Original-Maintainer. Questo può essere fatto automaticamente con lo script update-maintainer, disponibile nel pacchetto ubuntu-dev-tools. Per ulteriori informazioni, vedere le specifiche Debian Maintainer Field sul wiki di Ubuntu.

Ogni ulteriore paragrafo descrive un pacchetto binario da compilare.

Per ulteriori informazioni, vedere la sezione control file (capitolo 5) del Debian Policy Manual.

## Copyright File

Questo file fornisce le informazioni sul copyright sia per la sorgente Upstream sia per il pacchetto. Ubuntu e Debian Policy (sezione 12.5) richiedono che ogni pacchetto abbia una copia letterale del diritto d'autore e le informazioni di licenza in /usr/share/doc/$(nome_pacchetto)/copyright.

In generale, le informazioni sul copyright si trovano nel file COPYING nella directory dei sorgenti del programma. Questo file dovrebbe includere informazioni quali il nome dell'autore e chi ha provveduto al packaging, l'URL da cui la fonte proviene, una linea di copyright con l'anno e chi detiene il copyright e il testo dello stesso autore. Un modello di esempio potrebbe essere:

```
Format:
http://svn.debian.org/wsvn/dep
/web/deps/dep5.mdwn?op=file&re
v=166
Upstream-Name: Hello
Source:
ftp://ftp.example.com/pub/game
s

Files: *
Copyright: Copyright 1998 John
Doe <jdoe@example.com>
License: GPL-2+
This program is free software;
you can redistribute it and/or
modify it under the terms of
the GNU General Public
License as published by the
Free Software Foundation;
either version 2 of the
License, or (at your option)
any later version.
.
This program is distributed in
the hope that it will be
useful, but WITHOUT ANY
WARRANTY; without even the
implied warranty of
MERCHANTABILITY or FITNESS FOR
A PARTICULAR PURPOSE. See the
GNU General Public License for
more details.
.
You should have received a
copy of the GNU General Public
License along with this
package; if not, write to the
Free Software Foundation,
Inc., 51 Franklin St, Fifth
Floor, Boston, MA  02110-1301
USA
.
On Debian systems, the full
text of the GNU General Public
License version 2 can be found
in the file
`/usr/share/common-
licenses/GPL-2'.

Files: debian/*
Copyright: Copyright 1998 Jane
Doe <packager@example.com>
License: GPL-2+
```

Questo esempio segue la DEP-5: Machine-parseable debian/copyright proposal. Siete invitati a utilizzare questo tipo di formato.

## Il file rules

L'ultimo file che dobbiamo guardare è rules (regole). Questo fa tutto il lavoro di compilazione per il nostro pacchetto. Si tratta di un Makefile in cui ci sono gli obiettivi di compilazione e installazione dell'applicazione e quindi poi creare il file .deb dai file installati. Esso ha anche l'obiettivo di pulire tutti i file

usati per la compilazione e quindi si finirà con l'avere solo di nuovo un pacchetto sorgente.

Ecco una versione semplificata del file rules creato da dh_make (che può essere trovato nel pacchetto dh-make):

```
#!/usr/bin/make -f
# -*- makefile -*-

# Uncomment this to turn on
verbose mode.
#export DH_VERBOSE=1

%:
	dh  $@
```

Andiamo nel dettaglio di questo file. Quello che fa è passare ogni obiettivo di compilazione che viene richiamato da debian/rules come argomento in /usr/bin/dh, che quindi si richiamerà tutti i necessari comandi dh_*.

dh esegue una sequenza di comandi debhelper. Le sequenze supportate corrispondono agli obiettivi di un file debian/rules: "build", "clean", "install", "binary-arch", "binary-indep" e "binary". Per vedere quali comandi sono eseguiti in ciascun target, eseguire:

```
dh binary-arch --no-act
```

I comandi nella sequenza binary-indep sono passati con l'opzione "-i" per assicurarsi che funzionino solo su pacchetti binari indipendenti, mentre i comandi nella sequenza binary-arch sono passati con l'opzione "-a" per assicurarsi che funzionino solo su pacchetti dipendenti dall'architettura.

Ogni comando debhelper verrà registrato quando sarà eseguito con successo in debian/package.debhelper.log. (Che il comando dh_clean elimina.) Così dh può dire quali comandi sono già stati eseguiti, per quali pacchetti e saltare l'esecuzione di tali comandi. Ogni volta che Dh viene eseguito, esamina il registro e trova l'ultimo comando che viene registrato nella sequenza specificata. Si prosegue poi con il comando successivo nella sequenza. Le opzioni -until, -before, -after e -remaining possono sovrascrivere questo comportamento.

Se debian/rules contiene un obbiettivo con un nome come override_dh_command, quando si arriva a quel comando nella sequenza, dh eseguirà il comando dal file rules piuttosto che il comando vero e proprio. Il comando override quindi potrà essere allora eseguito sia con opzioni addizionali sia interamente diverso. (Si noti che per utilizzare questa funzione, si dovrebbe usare Build-Depend da debhelper 7.0.50 o superiore).

Date un'occhiata a /usr/share/doc/debhelper/examples/ e man dh per ulteriori esempi. Si veda anche la sezione rules (Sezione 4.9) del Debian Policy Manual.


## File aggiuntivi

## Il file di installazione

Il file di installazione è utilizzato da dh_install per installare i file nel pacchetto binario. Ha due casi di utilizzo standard:

- Installare i file nel pacchetto che non sono gestiti dal sistema di compilazione.
- Dividere un unico pacchetto sorgente di grandi dimensioni in pacchetti binari multipli.

Nel primo caso il file di installazione dovrebbe avere una riga per ogni file installato in cui sono specificati sia il file sia le directory di installazione. Per esempio il file di installazione seguente installerebbe lo script foo nella directory principale del pacchetto sorgente di usr/bin e un file sul desktop nella directory debian in usr/share/applications:

```
foo usr/bin
debian/bar.desktop
usr/share/applications
```

Quando un pacchetto sorgente produce pacchetti binari multipli, dh

installerà i file in debian/tmp piuttosto che direttamente in debian/<pacchetto>. I file installati in debian/tmp possono poi essere trasferiti in pacchetti binari separati utilizzando più file $package_name.install. Questo spesso è fatto per dividere grandi quantità di dati indipendenti dall'architettura fuori dall'architettura di dipendenza dei pacchetti e dentro la stessa architettura. In questo caso solo il nome dei file (o directory) sono necessari per l'installazione senza la directory di installazione. Per esempio foo.install contenente solo i file dipendenti dall'architettura potrebbe essere simile a:

```
usr/bin/
usr/lib/foo/*.so
```

Mentre foo-common.install contenente solo il file indipendenti dall'architettura, potrebbe essere simile:

```
/usr/share/doc/
/usr/share/icons/
/usr/share/foo/
/usr/share/locale/
```

Questo creerebbe due pacchetti binari, foo e foo-common. Entrambi richiedono il loro punto proprio in debian/control.

Si veda dh_install man e la sezione del file di installazione (Sezione 5.11) della Guida della Debian New Maintainer per ulteriori dettagli.

## Il file watch

Il file debian/watch ci permette di controllare automaticamente nuove versioni Upstream con lo strumento uscan trovato nel pacchetto devscripts. La prima riga del file watch deve essere la versione formato (3, al momento della scrittura), mentre le righe che seguono contengono tutti gli URL da analizzare. Per esempio:

```
version=3
http://ftp.gnu.org/gnu/hello/hello-(.*).tar.gz
```

Eseguendo uscan nella directory radice delle sorgenti, verrà ora confrontato il numero di versione Upstream in debian/changelog con l'ultima versione disponibile Upstream. Se viene trovata una nuova versione verrà scaricata automaticamente. Per esempio:

```
$ uscan
hello: Newer version (2.7) available on remote site:

http://ftp.gnu.org/gnu/hello/hello-2.7.tar.gz
  (local version is 2.6)
hello: Successfully downloaded updated package hello-2.7.tar.gz
    and symlinked hello_2.7.orig.tar.gz to it
```

Per ulteriori informazioni, vedere il man di uscan e la sezione file watch (Sezione 4.11) del Debian Policy Manual.

Per un elenco dei pacchetti per i quali il file watch riporta che non sono in sincronia con Upstream, si veda Ubuntu External Health Status.

## Il Source/Format File

Questo file indica il formato del pacchetto dei sorgenti. Attualmente le impostazioni predefinite del formato del pacchetto sorgente è 1.0 se il file non esiste. Si consiglia di usare il formato sorgente 3.0 più recente. In questo caso, il file deve contenere una sola linea che indica il formato desiderato:

- 3.0 (nativo) per i pacchetti Debian nativo (nessuna versione Upstream) o
- 3.0 (quilt) per i pacchetti con un archivio separato Upstream

Se, per qualche motivo, si desidera continuare a utilizzare il vecchio formato, si prega di creare questo file e renderlo in 1.0 per essere espliciti sulla versione del pacchetto sorgente. Questo permette la futura rimozione del predefinito 1.0 per il formato di origine del pacchetto.

http://wiki.debian.org/Projects/DebSrc3.0 riassume le informazioni relative ed i vantaggi nel passare al formato di pacchetto sorgente 3.0.

Si veda man dpkg-source e la sezione source/format (Sezione 5.21) della Guida Debian New Maintainers, per ulteriori dettagli.

## Risorse aggiuntive

Oltre ai link al Debian Policy Manual in ciascuna sezione precedente, la Guida Debian New Maintainers contiene una descrizione più dettagliata di ogni file. Il capitolo 4, "File richiesti nella directory debian" discute ulteriormente del file controllo, changelog, copyright e rules. Il capitolo 5, "Altri file nella directory debian" discute dei file aggiuntivi che possono essere utilizzati.