Software Release Practice HOWTO

Eric S. Raymond <esr@thyrsus.com> 1.0, 21 novembre 1998

Questo HOWTO descrive le regole di una buona release per un progetto open-source per Linux. Seguendo queste regole, si faciliterà il più possibile gli utenti nel costruire il codice e usarlo, e gli altri sviluppatori nel capirlo e cooperare per migliorarlo. Questo documento è una lettura obbligata per i novelli sviluppatori. Gli sviluppatori esperti devono fare una revisione quando stanno rilasciando un nuovo progetto. Verrà riveduto periodicamente per mostrare le evoluzioni degli standard. La traduzione italiana è stata curata da Edoardo Rampazzo <Taedium@mclink.it>

Indice

1 Introduzione

1.1 Perché questo documento?

C'è un gran numero di buone regole tradizionali per un codice open-source che aiutano gli altri nel portarlo, usarlo e cooperare nel suo sviluppo. Alcune di queste convenzioni sono tradizionali nel mondo Unix e nei precursori di Linux; altre sono state sviluppate recentemente in risposta a particolari nuovi strumenti e nuove tecnologie come il World Wide Web.

Questo documento aiuterà ad acquisire queste regole. Lo stesso è organizzato in sezioni, ciascuna contenente un indice degli argomenti. Quest'ultimo servirà anche come controllo preliminare ai fini della distribuzione.

1.2 Nuove versioni di questo documento

Questo documento sarà postato mensilmente nel newsgroup `comp.os.linux.answers` . Il documento è archiviato su diversi siti FTP riguardanti Linux, incluso `sunsite.unc.edu` in `pub/Linux/docs/HOWTO`.

Si può prendere visione dell'ultima versione di questo HOWTO sul World Wide Web all'URL `http://sunsite.unc.edu/LDP/HOWTO/Software-Release-Practice.html` .

Si è autorizzati a spedire domande o commenti su questo HOWTO a Eric S. Raymond,

esr@snark.thyrsus.com `mailto:esr@snark.thyrsus.com` .

2 Buone regole per dare un nome a un progetto e a un archivio

Poiché il carico sui distributori di archivi, come Sunsite, il sito del PSA e CPAN, è in aumento, si tende sempre più a far processare le proposte da valutare in parte o completamente da dei programmi (piuttosto che interamente da una persona).

Questo rende più importante, per un progetto o un nome di un archivio, adattare bene i nomi a modelli regolari che il programma possa analizzare e capire.

2.1 Usare nomi nello stile GNU, con una radice e numerazione delle patch primarie.secondarie

È bene che tutti i file dell'archivio abbiano nomi sullo stile GNU: radice prefissante in caratteri tutti minuscoli, seguiti da un trattino, dal numero della versione, dall'estensione, e da altri suffissi.

Supponiamo che si abbia un progetto di nome 'foobar', alla versione 1, release 2, livello 3. Se si tratta di un solo archivio (presumibilmente i sorgenti), ecco come dovrebbero essere i nomi:

foobar-1.2.3.tar.gz

L'archivio dei sorgenti

foobar.lsm

Il file LSM (assumendo che si sia sottoposti a Sunsite).

per favore *non* usare:

foobar123.tar.gz

Questo sembrerà a molti programmi un archivio di un progetto chiamato 'foobar123' senza numero della versione.

foobar1.2.3.tar.gz

Questo sembrerà a molti programmi un archivio di un progetto chiamato 'foobar1' nella versione 2.3.

foobar-v1.2.3.tar.gz

Molti programmi lo interpreterebbero come un progetto chiamato 'foobar-v1'.

foo_bar-1.2.3.tar.gz

La sottolineatura è difficile da leggere, scrivere e ricordare

FooBar-1.2.3.tar.gz

A meno che non *si voglia* assomigliare ad una mezza cartuccia del marketing. Inoltre è difficile da leggere, scrivere, e ricordare.

Se si deve distinguere tra archivi di sorgenti e di binari, o tra tipi diversi di binari, o esprimere alcuni tipi di opzioni nel nome del file, si è pregati di considerarle come un' estensione del file e metterle *dopo* il numero della versione. Cioè, è meglio scrivere:

foobar-1.2.3.src.tar.gz

sorgenti

foobar-1.2.3.bin.tar.gz

binari, di tipo non precisato

foobar-1.2.3.bin.ELF.tar.gz

binari di tipo ELF

foobar-1.2.3.bin.ELF.static.tar.gz

binari ELF linkati staticamente

foobar-1.2.3.bin.SPARC.tar.gz

binari SPARC

Si è pregati di *non* usare nomi come 'foobar-ELF-1.2.3.tar.gz', perché i programmi hanno difficoltà a distingure i suffissi (come '-ELF') dalla radice del nome.

Un buon modello generale per un nome è costituito, nell'ordine, da:

1. prefisso del progetto
2. trattino
3. numero della versione
4. punto
5. src o bin (facoltativo)
6. punto o trattino (meglio un punto)
7. tipologia del binario e opzioni (facoltativo)
8. estensioni dell'archivio e del tipo di compressione

2.2 Scegliere con accuratezza un nome per il prefisso che sia unico e facile da digitare

La radice prefissante dovrebbe essere comune a tutti i file di un progetto, facile da leggere, scrivere e ricordare. Si è pregati quindi di non usare sottolineature. Ugualmente non usare caratteri maiuscoli e maiuscoletti senza una buona ragione – confondono l'ordine di ricerca dell'occhio umano e ricordano una mezza cartuccia del marketing che voglia fare l'intelligente.

Crea inoltre confusione il fatto che due progetti diversi hanno lo stesso nome. Si cerchi così di controllare eventuali conflitti prima di rilasciare la propria nuova release. Un buon luogo dove controllare è costituito dal file di indice di Sunsite `http://sunsite.unc.edu/pub/Linux` .

3 Buone regole di sviluppo

La maggior parte di queste regole sono volte ad assicurare la portabilità, non solo verso sistemi Linux ma ugualmente verso altri sistemi Unix. Essere portabile verso altri sistemi Unix non è solo un segno di professionalità e cortesia del programmatore, ma anche un'assicurazione preziosa contro eventuali futuri cambiamenti in Linux stesso.

Inoltre, altre persone *cercheranno* di compilare il loro codice su sistemi non-Linux; con la portabilità si ridurrà il numero di persone perplesse che invieranno fastidiosissime e-mail.

3.1 Scrivere in puro ANSI C o in un linguaggio portabile

Per la portabilità e la stabilità, si dovrebbe scrivere in ANSI C o in un altro linguaggio che sia garantito come portabile in quanto abbia un'implementazione multi-piattaforma.

Linguaggi di questo tipo includono Python, Perl, Tcl ed Emacs Lisp. Vecchi, semplici linguaggi shell non sono qualificati; ci sono troppe differenti implementazioni con sottili idiosincrasie, e l'ambiente shell è soggetto a interruzioni dovute alle configurazioni dell'utente come, ad esempio, quelle degli alias delle shell.

Java promette bene come linguaggio portabile, ma le implementazioni disponibili per Linux sono appena abbozzate e scarsamente integrate con Linux. La scelta di Java è per ora il minore dei mali, sebbene possa divenire più popolare con il tempo.

3.2 Seguire le buone regole della portabilità del linguaggio C

Se si scrive in C, si è liberi di usare pienamente le caratteristiche ANSI – prototipi di funzioni inclusi, che aiuteranno a individuare le inconsistenze tra i diversi moduli. I compilatori in vecchio stile K&R sono superati.

D'altra parte non si presupponga che alcune caratteristiche specifiche del GCC come l'opzione '-pipe' o le funzioni nidificate siano disponibili. Queste si torceranno contro a chiunque faccia un porting verso sistemi non-Linux o non-GCC.

3.3 Utilizzare autoconf/ automake/ autoheader

Se si scrive in C, usare autoconf/automake/autoheader per risolvere problemi di portabilità, sondare la configurazione del sistema, e confezionare i makefile. Chi installa dai sorgenti si aspetta, giustamente, di poter digitare configure; make e ottenere una struttura pulita.

3.4 Controllare lo stato del codice prima di rilasciarlo

Se si scrive in C, fare un test con '-Wall' ed eliminare gli errori almeno una volta prima di ciascuna release. Questo blocca un numero sorprendente di errori. Per una totale completezza si compili anche con '-pedantic'.

Se si scrive in Perl, controllare il codice con 'perl -c', 'perl -w', e 'perl -T' prima di ciascuna release (si veda la documentazione Perl).

4 Buone regole per creare una distribuzione

Le seguenti indicazioni descrivono come la propria distribuzione dovrebbe apparire quando qualcuno la rintraccia, la scarica e la decomprime.

4.1 Essere sicuri che gli archivi tar vengano sempre decompressi in una nuova singola directory

Il più fastidioso errore dei novelli sviluppatori consiste nel fare archivi tar che decomprimono i file e le directory della distribuzione nella directory corrente, magari sovrascrivendo file già esistenti. *Non bisogna farlo mai !*

Invece, si dev'essere sicuri che i file del proprio pacchetto abbiano tutti una porzione di directory chiamata come il progetto, affinché questi vengano decompressi in una singola directory direttamente *sotto* quella corrente.

Ecco un trucco per un makefile che agisce in questo modo, assumendo che la directory della propria distribuzione venga chiamata 'foobar' e che SRC contenga un elenco dei file della distribuzione stessa. Ciò richiede GNU tar 1.13

```
VERS=1.0
foobar-$(VERS).tar.gz:
        tar --prefix-name='foobar-$(VERS)/' -czf foobar-$(VERS).tar.gz $(SRC)
```

Se si ha un versione più vecchia di tar, si può provare una cosa del tipo:

```
foobar-$(VERS).tar.gz:
        @ls $(SRC) | sed s:^:foobar-$(VERS)/: >MANIFEST
        @(cd ..; ln -s foobar foobar-$(VERS))
        (cd ..; tar -czvf foobar/foobar-$(VERS).tar.gz `cat foobar/MANIFEST`)
        @(cd ..; rm foobar-$(VERS))
```

4.2 Fornire un README

Bisogna fornire un file chiamato README o READ.ME che funga da mappa per la propria distribuzione. Per una vecchia convenzione, questo è il primo file che gli intrepidi esploratori leggeranno dopo aver decompresso i sorgenti.

È bene che il README includa:

- Una breve descrizione del progetto.
- Un link al sito web del progetto (se ne ha uno).

- Note sull'ambiente di progettazione ed eventuali problemi di portabilità.
- Una mappa che descriva i file importanti e le sotto-directory.
- Istruzioni per l'installazione e la compilazione o un richiamo a un file che contenga le stesse (di solito INSTALL).
- Un elenco di coloro che hanno fatto e che mantengono il progetto o un richiamo ad un file che lo contenga (di solito CREDITS).
- Novità recenti sul progetto o un richiamo ad un file che contenga le stesse (di solito NEWS).

4.3 Rispettare e seguire le regole standard per la titolazione dei file

Prima ancora di leggere il README, l'intrepido esploratore avrà analizzato i nomi dei file nella directory della distribuzione appena decompressa. Quei nomi possono essi stessi fornire delle informazioni. Aderendo a standard fissi sulle regole di nominazione, si può dare all'esploratore degli indizi preziosi su ciò che è in procinto di guardare.

Ecco alcuni nomi standard di file e il loro significato. Non è detto che siano tutti necessari per tutte le distribuzioni.

README or READ.ME

Il file della mappa, da leggersi per primo

INSTALL

istruzioni per la configurazione, struttura e l'installazione

CREDITS

elenco di chi ha contribuito al progetto

NEWS

notizie recenti sul progetto

HISTORY

storia del progetto

COPYING

termini di licenza del progetto (convenzione GNU)

LICENSE

termini di licenza del progetto

MANIFEST

elenco dei file della distribuzione

FAQ

documento testo contenente le domande poste più di frequente (Frequently Asked Question) sul progetto

TAGS

tag-file generato per uso di Emacs o vi

N.B. Si accetta come convenzione generale che i nomi dei file scritti tutti in caratteri maiuscoli costituiscano matainformazioni leggibili sul pacchetto, e non sui suoi componenti.

5 Buone regole di comunicazione

Il proprio software non renderà il mondo migliore se nessuno sa che esiste. Inoltre, con una presenza visibile del progetto su Internet sarà più semplice trovare utenti e co-sviluppatori. Ecco i modi standard per farlo.

5.1 Riferire a c.o.l.a

Annunciare la nuova release a *comp.os.linux.announce* `news:comp.os.linux.announce` . Oltre a essere letto da moltissime persone, questo gruppo è una dei più grossi contenitori di siti web riguardanti le novità del settore, come *Freshmeat* `http://www.freshmeat.net` .

5.2 Riferire ad un newsgroup attinente al tema

Si trovi un gruppo USENET attinente alla propria applicazione, e si annunci lì la sua uscita. Si posti solo dove la *funzione* del codice è attinente, e si restringa il campo d'azione. Se (per esempio) si sta rilasciando un programma scritto in Perl che consulti un server IMAP, si dovrebbe sicuramente postare a comp.mail.imap. Ma non si dovrebbe probabilmente postare a comp.lang.perl a meno che il programma non sia anche un esempio istruttivo delle tecniche Perl all'avanguardia.

L'annuncio dovrebbe includere l'URL di un sito web del progetto.

5.3 Fornire un sito Web

È importante avere un sito web se si vogliono raccogliere utenti o un gruppo di sviluppatori del progetto. Caratteristiche standard di un sito sono:

- Lo statuto del progetto (perché esiste, a chi si rivolge, ecc.).
- Collegamenti per scaricare i sorgenti.
- Istruzioni per iscriversi alla/alle mailing list del progetto.
- Un elenco di FAQ (Frequently Asked Questions).
- Documentazione in HTML sul progetto.
- Link a progetti correlati e/o in competizione con il proprio.

Alcuni siti di progetti forniscono anche degli URL per l'accesso anonimo al master source tree.

5.4 Creare delle mailing list attinenti al progetto

È una pratica comune creare una lista privata di sviluppo attraverso cui i collaboratori del progetto possano comunicare e scambiarsi delle patch. Si potrebbe anche creare una lista di annunci e notizie per le persone che vogliano essere tenute informate sullo stato del progetto.

5.5 Distribuzione attraverso i più importanti archivi

Negli anni scorsi, l'archivio di *Sunsite* `http://www.sunsite/unc.edu/pub/Linux/`

è stato il più importante luogo di scambio per i programmi Linux.

Altri siti importanti sono:

- Il sito del *Python Software Activity* `http://www.python.org`

  (per programmi scritti in Python).
- Il *CPAN* `http://language.perl.com/CPAN` , Comprehensive Perl Archive Network, (per programmi scritti in Perl).

5.6 Fornire archivi RPM

La configurazione standard de-facto per i pacchetti di binari installabili è quella usata dal Red Hat Package Manager, RPM. È presente nella maggior parte delle più conosciute distribuzioni Linux, e supportato efficacemente da praticamente tutte le altre (eccettute Debian e Slackware).

Di conseguenza, è una buona idea per il sito del proprio progetto fornire tanto pacchetti RPM installabili quanto archivi tar.