Guida del Committer

The FreeBSD Italian Documentation Project

Revisione: 43184

Diritto d'autore © 1999, 2000, 2001, 2002, 2003,
2004 The FreeBSD Italian Documentation Project

FreeBSD è un marchio registrato della FreeBSD Foundation.

CVSup è un marchio registrato di John D. Polstra.

IBM, AIX, OS/2, PowerPC, PS/2, S/390, e ThinkPad sono marchi della International Business Machines Corporation negli Stati Uniti, in altri paesi, o in entrambi.

Intel, Celeron, EtherExpress, i386, i486, Itanium, Pentium, e Xeon sono marchi o marchi registrati della Intel Corporation o delle sue sussidiarie negli Stati Uniti e in altri paesi.

Sparc, Sparc64, e UltraSPARC sono marchi SPARC International, Inc negli Stati Uniti e in altri paesi. I prodotti che portano il marchio SPARC sono basati su architettura sviluppata dalla Sun Microsystems, Inc.

Molti dei nomi identificativi usati dai produttori e dai venditori per distinguere i loro prodotti sono anche dei marchi. Quando questi nomi appaiono nel libro, e il FreeBSD Project è al corrente del marchio, vengono fatti seguire dal simbolo «TM» o «®».

2013-11-13 di hrs.

Sommario

Questo documento fornisce informazioni per la comunità dei committer di FreeBSD. Tutti i nuovi committer dovrebbero leggere questo documento prima di iniziare, e i committer già esistenti sono fortemente incoraggiati a riguardarselo di tanto in tanto.

Traduzione a cura di Alex Dupre <ale@FreeBSD.org >.

Indice

1. Dettagli Amministrativi 2
2. Tipi di Bit di Commit 3
3. Operazioni sul CVS 4

4. Convenzioni e Tradizioni 14
5. Licenza Preferita per i Nuovi File 16
6. Relazioni tra Sviluppatori 17
7. GNATS 18
8. Chi è Chi 20
9. Guida Rapida a SSH 22
10. Il Lungo Elenco di Regole dei Committer di FreeBSD 23
11. Supporto per Diverse Architetture 23
12. FAQ Specifiche sui Port 23
13. Benefici del Lavoro 23
14. Domande Generali 23

1. Dettagli Amministrativi

Host con il Repository Principale	`ncvs.FreeBSD.org`
Metodi di Accesso	ssh(1), solo protocollo 2
CVSROOT Principale	`ncvs.FreeBSD.org :/home/ncvs` (guarda anche la Sezione 3, «Operazioni sul CVS»).
CVS Repository Meisters <cvsadm@FreeBSD.org > Principali	Peter Wemm e Mark Murray, così come Josef Karthauser e Joe Marcus Clarke per i `ports/`
Mailing List	mailing list degli sviluppatori del doc/ di FreeBSD, mailing list del doc/ committer di FreeBSD; mailing list degli sviluppatori del ports/ di FreeBSD, mailing list del ports/ committer di FreeBSD; mailing list degli sviluppatori del src/ di FreeBSD, mailing list del src/ committer di FreeBSD. (Ogni repository di progetto ha le sue mailing list -developers e -committers. Gli archivi per queste liste possono essere trovati nei file `/home/mail/repository-name-developers-archive` e `/home/mail/repository-name-committers-archive` sul cluster di `FreeBSD.org` .)
Report mensili del Core Team	`/home/core/public/monthly-report` sul cluster di `FreeBSD.org` .
Tag CVS Degni di Nota	`RELENG_4` (4.X-STABLE), `RELENG_5` (5.X-STABLE), `HEAD` (-CURRENT)

È richiesto l'uso di ssh(1) o telnet(1) con Kerberos 5 per connettersi agli host del progetto. Per ssh(1) è permesso solo il protocollo 2. Questi sono generalmente più sicuri che un semplice telnet(1) o rlogin(1) visto che la negoziazione delle credenziali avverrà sempre in modo cifrato. Tutto il traffico è cifrato di default con ssh(1). Insieme a programmi di utilità come ssh-agent(1) e scp(1), anch'essi disponibili, ssh(1) è di gran lunga più conveniente. Se non sai nulla di ssh(1), guarda la Sezione 9, «Guida Rapida a SSH».

2. Tipi di Bit di Commit

Il repository CVS di FreeBSD ha un numero di componenti che, se combinati, supportano i sorgenti di base del sistema operativo, la documentazione, l'infrastruttura dei port delle applicazioni di terze parti, e vari programmi di utilità. Quando vengono assegnati i bit di commit di FreeBSD, vengono specificate le aree dell'albero dove il bit può essere usato. Solitamente, le aree associate a un bit corrispondono a quelle di chi ha autorizzato l'assegnamento del bit di commit. Ulteriori aree di autorità possono essere aggiunte in seguito: se occorrerà, il committer dovrà seguire le normali procedure di allocazione del bit di commit per quell'area dell'albero, chiedendo l'approvazione all'entità appropriata e possibilmente prendendo un mentore per quell'area per un po' di tempo.

Tipo di Committer	*Responsabile*	*Componenti dell'Albero*
src	core@	src/, doc/ soggetta ad appropriata revisione
doc	doceng@	doc/, www/, documentazione src/
ports	portmgr@	ports/

I bit di commit assegnati prima dello sviluppo della nozione di aree di autorità possono essere usati in molte parti dell'albero. Tuttavia, il buon senso dice che un committer che non ha mai lavorato precedentemente in un'area dell'albero chieda una revisione del proprio lavoro prima di effettuare il commit, chieda l'approvazione del responsabile appropriato, e/o lavori d'accordo con un mentore. Dato che le regole sulla manutenzione del codice differiscono a seconda dell'area dell'albero, questo è per il bene del committer che lavora in un'area poco familiare tanto quanto per gli altri che lavorano sull'albero.

I committer sono incoraggiati a chiedere la revisione del proprio lavoro come parte del normale processo di sviluppo, indifferentemente dall'area dell'albero in cui stanno lavorando.

2.1. Regolamento dell'attività del `doc/` committer in `src/`

- I doc committer possono effettuare commit riguardanti modifiche alla documentazione sui file src, come pagine man, README, database dei fortune, file dei calendari, e

correzioni sui commenti senza l'approvazione di un src committer, prestando la solita attenzione e cura ai commit.

- I doc committer possono effettuare commit riguardanti piccole modifiche e correzioni ai sorgenti, come correzioni per la compilazione, piccole funzionalità, ecc., con un «Approved by» di un src committer.

- I doc committer possono cercare di ottenere il commit bit sui src acquisendo un mentore, che proporrà il doc committer al core. Una volta approvato, verrà aggiunto al file `access` ed inizierà il normale periodo sotto la guida del mentore, che implica l'aggiunta di «Approved by» per un certo periodo.

- «Approved by» può essere usato solamente se l'approvazione è di un src committer senza mentore - i committer ancora sotto la guida di un mentore possono fornire al più un «Reviewed by» ma non un «Approved by».

3. Operazioni sul CVS

Si assume che tu abbia già familiarità con le operazioni di base di CVS.

I CVS Repository Meisters <cvsadm@FreeBSD.org > sono i «proprietari» del repository CVS e sono responsabili delle sue modifiche dirette allo scopo di ripulire o sistemare dei gravi abusi di CVS da parte di un committer. Nel caso dovessi causare qualche problema al repository, diciamo una errata operazione di `cvs import` o `cvs tag`, invia un messaggio al membro responsabile fra i CVS Repository Meisters <cvsadm@FreeBSD.org >, come stabilito nella tabella qui sotto, (o chiama uno di loro) ed esponi il problema. Per questioni molto importanti che interessano l'intero albero CVS-non solo un'area specifica-puoi contattare i CVS Repository Meisters <cvsadm@FreeBSD.org >. *Non* contattare i CVS Repository Meisters <cvsadm@FreeBSD.org > per copie di repository o altre cose che possono gestire i team più specifici.

Gli unici che hanno il permesso di manipolare direttamente i bit del repository sono i «repomeister». Per questo non ci sono shell di login disponibili sulle macchine del repository, tranne che per i repomeister.

Nota

A seconda dell'area interessata del repository CVS, dovresti mandare la tua richiesta a uno dei seguenti indirizzi email:

- ncvs@ - a proposito di `/home/ncvs`, il repository dei src
- pcvs@ - a proposito di `/home/pcvs`, il repository dei port

- dcvs@ - a proposito di `/home/dcvs`, il repository dei doc
- projcvs@ - a proposito di `/home/projcvs`, il repository dei progetti di terze parti

L'albero CVS è attualmente diviso in quattro repository differenti, ovvero `doc`, `ports`, `projects` e `src`. Questi vengono ricomposti sotto un unico `CVSROOT` quando vengono distribuiti tramite CVSup per la convenienza dei nostri utenti.

Nota

Nota che il modulo `www` che contiene i sorgenti del sito web di FreeBSD è contenuto all'interno del repository `doc`.

I repository CVS sono ospitati sulle macchine repository. Attualmente, ognuno dei repository elencati qui sopra risiede sulla stessa macchina fisica, `ncvs.FreeBSD.org` , ma per permettere la possibilità di averne ognuno su una macchina diversa in futuro, ci sono diversi nomi di host che i committer dovrebbero utilizzare. Inoltre, ogni repository risiede in una directory differente. La seguente tabella racchiude la situazione.

Tabella 1. Repository CVS, Host e Directory di FreeBSD

Repository	Host	Directory
doc	dcvs.FreeBSD.org	/home/dcvs
ports	pcvs.FreeBSD.org	/home/pcvs
projects	projcvs.FreeBSD.org	/home/projcvs
src	ncvs.FreeBSD.org	/home/ncvs

Le operazioni sul CVS sono fatte da remoto impostando la variabile di ambiente `CVSROOT` a `ncvs.FreeBSD.org :/home/ncvs` e la variabile `CVS_RSH` a `ssh`, e quindi effettuando le appropriate operazioni di check-out/check-in. Molti committer definiscono degli alias che si espandono nella corretta invocazione di cvs per il repository appropriato. Per esempio, un utente di tcsh(1) può aggiungere le seguenti righe al suo `.cshrc` per questo scopo:

```
alias dcvs env CVS_RSH=ssh cvs -d user@dcvs.FreeBSD.org:/home/dcvs
alias pcvs env CVS_RSH=ssh cvs -d user@pcvs.FreeBSD.org:/home/pcvs
alias projcvs env CVS_RSH=ssh cvs -d user@projcvs.FreeBSD.org:/home/
projcvs
alias scvs env CVS_RSH=ssh cvs -d user@ncvs.FreeBSD.org:/home/ncvs
```

In questo modo è possibile fare tutte le operazioni di CVS localmente ed usare `Xcvs commit` per effettuare il commit sull'albero CVS ufficiale. Se desideri aggiungere qualcosa di totalmente nuovo (ad esempio dei sorgenti in contrib, ecc.), deve essere usato `cvs import`. Guarda come riferimento la pagina man di cvs(1) per l'utilizzo.

Nota

Per favore *non* usare `cvs checkout` o `update` con la macchina con il repository ufficiale impostata come CVS Root per tenere aggiornato il tuo albero dei sorgenti. CVS da remoto non è ottimizzato per la distribuzione via rete e richiede un grande sovraccarico di lavoro e di amministrazione sul lato server. Utilizza il nostro metodo di distribuzione avanzato `cvsup` per ottenere i bit del repository, ed esegui solamente l'operazione di `commit` sull'host con il repository. Forniamo un'estesa rete di mirror cvsup per questo scopo, così come diamo accesso al `cvsup-master` se hai veramente bisogno di essere aggiornato alle ultime modifiche. Il `cvsup-master` ha la potenza necessaria a gestire questa cosa, il repository principale no. Jun Kuriyama è a capo del `cvsup-master`.

Se devi usare le operazioni `add` e `delete` di CVS come se fosse un'operazione mv(1), allora va effettuata una copia nel repository piuttosto che usare `add` e `delete` di CVS. In una copia nel repository, un CVS Meister copierà il/i file nei loro nuovi nomi e/o locazioni e ti avviserà ad operazione avvenuta. Lo scopo di una copia del repository è di preservare la cronologia dei cambiamenti del file, o i log. Noi del FreeBSD Project diamo molta importanza alla cronologia dei cambiamenti che CVS fornisce al progetto.

Informazioni di riferimento, tutorial, e FAQ su CVS possono essere trovate su: `http://www.cvshome.org/docs/` . Anche le informazioni contenute nei capitoli di Karl Fogel da «Open Source Development with CVS» sono molto utili.

Dag-Erling C. Smørgrav ha fornito inoltre il seguente «mini manuale» su CVS.

1. Effettua il check out di un modulo con il comando `co` o `checkout`.

```
% cvs checkout shazam
```

   Questo estrae una copia del modulo `shazam`. Se non c'è alcun modulo `shazam` nel file dei moduli, cercherà allora una directory di primo livello chiamata `shazam`.

Tabella 2. Opzioni utili con `cvs checkout`

`-P`	Non crea le directory vuote

`-l`	Estrae solo un livello, non le sottodirectory
`-rver`	Estrai la versione, il ramo, o il tag *ver*
`-Ddata`	Estrai i sorgenti com'erano in data *data*

Esempi pratici su FreeBSD:

- Estrai il modulo `miscfs`, che corrisponde a `src/sys/miscfs` :

```
% cvs co miscfs
```

  Ora hai una directory chiamata `miscfs` con le sottodirectory `CVS`, `deadfs`, `devfs`, e così via. Una di queste (`linprocfs`) è vuota.

- Estrai gli stessi file, ma con il percorso completo:

```
% cvs co src/sys/miscfs
```

  Ora hai una directory chiamata `src`, con le sottodirectory `CVS` e `sys`. La directory `src/sys` ha le sottodirectory `CVS` e `miscfs`, ecc.

- Estrai gli stessi file, ma elimina le directory vuote:

```
% cvs co -P miscfs
```

  Ora hai una directory chiamata `miscfs` con le sottodirectory `CVS`, `deadfs`, `devfs`... ma nota che non c'è nessuna sottodirectory `linprocfs`, perché non contiene alcun file.

- Estrai la directory `miscfs`, ma nessuna delle sue sottodirectory:

```
% cvs co -l miscfs
```

  Ora hai una a directory chiamata `miscfs` con solo una sottodirectory chiamata `CVS`.

- Estrai il modulo `miscfs` com'è nel ramo 4.X:

```
% cvs co -rRELENG_4 miscfs
```

  Puoi modificare i sorgenti ed effettuare il commit su questo ramo.

- Estrai il modulo `miscfs` com'era nella 3.4-RELEASE.

```
% cvs co -rRELENG_3_4_0_RELEASE miscfs
```

  Non potrai effettuare il commit delle modifiche, visto che `RELENG_3_4_0_RELEASE` corrisponde ad un preciso istante di tempo, non a un ramo.

- Estrai il modulo `miscfs` com'era il 15 gennaio 2000.

```
% cvs co -D'01/15/2000' miscfs
```

Non potrai effettuare modifiche.

- Estrai il modulo `miscfs` com'era una settimana fa.

```
% cvs co -D'last week' miscfs
```

Non potrai effettuare modifiche.

Tieni presente che cvs salva i metadati in sottodirectory chiamate `CVS`.

Gli argomenti di `-D` e `-r` sono fissi, che vuol dire che cvs se li ricorderà in seguito, ad esempio quando farai un `cvs update`.

2. Controlla lo stato dei file estratti con il comando `status`.

```
% cvs status shazam
```

Questo visualizza lo stato del file `shazam` o di ogni file nella directory `shazam`. Per ogni file, lo stato è uno fra:

Up-to-date	Il file à aggiornato e non è stato modificato.
Needs Patch	Il file non è stato modificato, ma c'è una nuova versione nel repository.
Locally Modified	Il file è aggiornato, ma è stato modificato.
Needs Merge	Il file è stato modificato, e c'è una nuova versione nel repository.
File had conflicts on merge	Ci sono stati conflitti l'ultima volta che il file è stato aggiornato, e non sono ancora stati risolti.

Vedrai anche la versione e la data locale, il numero dell'ultima versione appropriata («ultima appropriata» perché se hai una data, un tag o un ramo fissati, può non essere l'ultima versione), e i tag, le date o le opzioni applicate.

3. Dopo avere estratto qualcosa, puoi aggiornarlo con il comando `update`.

```
% cvs update shazam
```

Questo aggiorna il file `shazam` o il contenuto della directory `shazam` all'ultima versione sul ramo che hai estratto. Se hai estratto un «preciso instante di tempo», non fa nulla a meno che i tag non siano stati spostati nel repository o qualche altra strana cosa sia in corso.

Opzioni utili, in aggiunta a quelle elencate sopra, con `checkout`:

`-d`	Estrae ogni directory aggiuntiva mancante.
`-A`	Scarica l'ultima versione del ramo principale.
`-jver`	Altre magie (guarda sotto).

Se hai estratto un modulo con `-r` o `-D`, l'esecuzione di `cvs update` con un argomento differente di `-r` o `-D` o con `-A` selezionerà un nuovo ramo, una nuova versione o una nuova data. L'opzione `-A` elimina tutti i tag, le date o le versioni fissate mentre `-r` e `-D` ne impostano di nuove.

Teoricamente, specificando `HEAD` come argomento di `-r` avrai lo stesso risultato di `-A`, ma è solo in teoria.

L'opzione `-d` è utile se:

- qualcuno ha aggiunto delle sottodirectory al modulo che hai estratto dopo averlo estratto.
- hai estratto con `-l`, e dopo cambi idea e vuoi estrarre anche le sottodirectory.
- hai cancellato delle sottodirectory e vuoi estrarle nuovamente.

Osserva l'output di `cvs update` con cura. La lettera all'inizio di ogni file indica cosa è stato fatto su di esso:

`U`	Il file è stato aggiornato senza problemi.
`P`	Il file è stato aggiornato senza problemi (vedrai questo solo quando lavorerai su un repository remoto).
`M`	Il file è stato modificato, ed è stato fuso senza conflitti.
`C`	Il file è stato modificato, ed è stato fuso con dei conflitti.

La fusione è ciò che avviene quando estrai una copia di qualche codice sorgente, lo modifichi, quindi qualcun altro effettua il commit di un'altra modifica, e tu esegui `cvs update`. CVS nota che tu hai fatto dei cambiamenti locali, e cerca di fondere le tue modifiche con quelle fatte tra la versione che hai originariamente estratto e quella che stai aggiornando. Se i cambiamenti sono a due parti separate del file, solitamente non ci saranno problemi (sebbene il risultato possa non essere sintatticamente o semanticamente corretto).

CVS stamperà una `M` davanti ad ogni file modificato localmente anche se non c'è una nuova versione nel repository, quindi `cvs update` è adatto per avere un resoconto di quello che hai cambiato in locale.

Se appare una `C`, allora le tue modifiche sono in conflitto con i cambiamenti presenti nel repository (le modifiche sono sulle stesse righe, o righe vicine, o hai cambiato così tanto il file locale che `cvs` non è in grado di applicare le modifiche al repository). Dovrai allora andare a modificare il file a mano e risolvere i conflitti; questi saranno evidenziati da righe di simboli `<`, `=` e `>`. Per ogni conflitto, ci sarà una linea di demarcazione formata da sette `<` e il nome del file, seguita da una porzione di quello che il tuo file locale conteneva, seguita da una riga di separazione con sette `=`, seguita dalla porzione corrispondente presente nella versione del repository, seguita da una riga di separazione con sette `>` e il numero di versione che stai aggiornando.

L'opzione `-j` è un po' voodoo. Aggiorna il file locale alla versione specificata come se avessi usato `-r`, ma non cambia il numero di versione o il ramo registrato del file locale. Non è realmente utile tranne quando usata due volte, nel qual caso fonderà le modifiche tra le due versioni specificate nella copia su cui stai lavorando.

Per esempio, supponiamo che ti abbia effettuato il commit di una modifica a `shazam/shazam.c` in FreeBSD-CURRENT e che più tardi tu voglia effettuare l'MFC. Le modifiche che vuoi fondere sono nella versione 1.15:

- Estrai la versione FreeBSD-STABLE del modulo `shazam`:

```
% cvs co -rRELENG_5 shazam
```

- Applica le modifiche tra la ver 1.14 e la 1.15:

```
% cvs update -j1.14 -j1.15 shazam/shazam.c
```

Quasi certamente avrai un conflitto a causa delle righe `$Id$` (o nel caso di FreeBSD, `$FreeBSD: head/it_IT.ISO8859-15/articles/committers-guide/article.xml 43184 2013-11-13 07:52:45Z hrs $`), quindi dovrai modificare a mano il file per risolvere il conflitto (rimuovi le righe di separazione e la seconda linea `$Id$`, lasciando la linea `$Id$` originale intatta).

4. Guarda le differenze tra la versione locale e quella sul repository con il comando `diff`.

```
% cvs diff shazam
```

mostra ogni modifica che hai fatto al file o al modulo `shazam`.

Tabella 3. Opzioni utili con `cvs diff`

`-u`	Utilizza il formato diff unificato.

`-c`	Utilizza il formato diff contestuale.
`-N`	Visualizza i file mancanti o aggiunti.

Vorrai sempre utilizzare `-u`, visto che le diff unificate sono molto più semplici da leggere rispetto a quasi tutti gli altri formati (in alcune circostanze, le diff contestuali generate con l'opzione `-c` possono essere meglio, ma sono molto più voluminose). Una diff unificata consiste di una serie di parti. Ogni parte inizia con una riga con due caratteri `@` e specifica dove si trovano le differenze nel file e su quante linee si estendono. Questa è seguita da un certo numero di righe; alcune (precedute da uno spazio) fanno parte del contesto; altre (precedute da un `-`) sono quelle eliminate e altre ancora (precedute da un `+`) sono quelle aggiunte.

Puoi anche effettuare una diff con una versione differente rispetto a quella che hai estratto specificando la versione con `-r` o `-D` come per il `checkout` o l'`update`, o anche visualizzare le differenze tra due versioni arbitrarie (indipendentemente da quella che hai localmente) specificando *due* versioni con `-r` o `-D`.

5. Guarda le righe di log con il comando `log`.

```
% cvs log shazam
```

Se `shazam` è un file, questo stamperà un'*intestazione* con le informazioni sul file, come la locazione nel repository dove il file è salvato, a quale versione è l'`HEAD` per questo file, in quali rami si trova il file, e qualsiasi tag valido per questo file. Quindi, per ogni versione del file, viene stampato un messaggio di log. Questo include la data e l'ora del commit, chi ha fatto il commit, quante righe sono state aggiunte e/o tolte, e alla fine il messaggio di log che il committer ha scritto quando ha inviato la modifica.

Se `shazam` è una directory, allora le informazioni di log descritte sopra vengono stampate a turno per ogni file presente nella directory. A meno che tu abbia dato l'opzione `-l` a `log`, vengono stampati anche i log per tutte le sottodirectory di `shazam`, in maniera ricorsiva.

Usa il comando `log` per vedere la storia di uno o più file, come è salvata nel repository CVS. Puoi anche usarlo per vedere il messaggio di log di una versione specifica, se aggiungi `-r`*`ver`* al comando `log`:

```
% cvs log -r1.2 shazam
```

Questo stamperà solamente il messaggio di log per la versione `1.2` del file `shazam` se è un file, oppure i messaggi di log per le versioni 1.2 di ogni file sotto `shazam` se è una directory.

6. Guarda chi ha fatto cosa con il comando `annotate`. Questo comando visualizza ogni riga del file o dei file specificati, insieme all'utente che ha modificato più recentemente quella riga.

```
% cvs annotate shazam
```

7. Aggiungi nuovi file con il comando `add`.

   Crea il file, usa `cvs add` su di esso, quindi `cvs commit`.

   In modo analogo, puoi aggiungere nuove directory creandole e poi utilizzando `cvs add` su di esse. Nota che non c'è bisogno di usare il commit sulle directory.

8. Rimuovi i file obsoleti con il comando `remove`.

   Rimuovi il file, quindi usa `cvs rm` su di esso, ed infine `cvs commit`.

9. Effettua il commit con il comando `commit` o `checkin`.

   Tabella 4. Opzioni utili con `cvs commit`

`-f`	Forza il commit di un file non modificato.
`-m`*`msg`*	Specifica un messaggio di commit sulla riga di comando anziché invocare un editor.

   Usa l'opzione `-f` se ti accorgi che hai lasciato fuori informazioni importanti dal messaggio di commit.

   Buoni messaggi di commit sono importanti. Dicono agli altri perché hai fatto le modifiche che hai fatto, non solo qui ed ora, ma per mesi o anni quando qualcuno si chiederà perché dei pezzi di codice all'apparenza illogici o inefficienti sono entrati nel file sorgente. È inoltre un aiuto inestimabile per decidere su quali modifiche va effettuato l'MFC e su quali no.

   I messaggi di commit devono essere chiari, concisi, e fornire un ragionevole sommario per dare un'indicazione di cosa è stato cambiato e perché.

   I messaggi di commit devono fornire abbastanza informazioni affinché una terza parte possa decidere se la modifica è rilevante per lei e se debba leggere la modifica stessa.

   Evita di effettuare il commit di più modifiche scollegate in una volta sola. Questo rende difficile la fusione, e inoltre rende più complicato determinare quale modifica è colpevole se salta fuori un bug.

   Evita di effettuare il commit di correzioni di stile o di spaziatura insieme a correzioni di funzionalità. Questo rende difficile la fusione, e inoltre rende più complicato capire quali modifiche alle funzionalità sono state fatte. Nel caso di file di documentazione, può rendere il lavoro dei gruppi di traduzione più complicato, visto che diventa difficile per loro determinare esattamente quali modifiche al contenuto vanno tradotte.

Evita di effettuare il commit di cambiamenti a più file con un unico messaggio generico o vago. Invece, effettua il commit di un file alla volta (o di piccoli gruppi di file correlati) con un messaggio di commit appropriato.

Prima di effettuare il commit, devi *sempre*:

- verificare su che ramo stai effettuando il commit, tramite `cvs status`.
- revisionare i tuoi cambiamenti, con `cvs diff`

Inoltre, devi SEMPRE specificare esplicitamente sulla riga di comando su quali file deve essere effettuato il commit, in modo da non toccare incidentalmente altri file non voluti - `cvs commit` senza argomenti effettuerà il commit di ogni modifica nella directory corrente ed ogni sottodirectory.

Suggerimenti e trucchi aggiuntivi:

1. Puoi inserire le opzioni più comunemente usate nel tuo `~/.cvsrc`, come in questo caso:

```
cvs -z3
diff -Nu
update -Pd
checkout -P
```

   Questo esempio dice:

   - usa sempre il livello di compressione 3 quando si parla con un server remoto. Questo è un salvavita quando si lavora su una connessione lenta.
   - usa sempre le opzioni `-N` (visualizza i file aggiunti o rimossi) e `-u` (formato diff unificato) con diff(1).
   - usa sempre le opzioni `-P` (elimina le directory vuote) e `-d` (estrai le nuove directory) quando si effettua l'update.
   - usa sempre l'opzione `-P` (elimina le directory vuote) quando si estrae.

2. Usa lo script `cdiff` di Eivind Eklund per visualizzare le diff unificate. È un wrapper per less(1) che aggiunge i codici colore ANSI per far risaltare le intestazioni delle sezioni, le righe rimosse e quelle aggiunte; il contesto rimane invariato. Inoltre espande i tab correttamente (i tab spesso appaiono errati nelle diff a causa del carattere aggiuntivo all'inizio di ogni riga).

   textproc/cdiff

   Semplicemente usalo al posto di more(1) o less(1):

```
% cvs diff -Nu shazam | cdiff
```

Alternativamente alcuni editor come vim(1) (editors/vim5) hanno il supporto al colore e quando vengono usati con l'evidenziazione della sintassi attiva evidenzieranno molti tipi di file, incluse le diff, le patch, e i log CVS/RCS.

```
% echo "syn on" >> ~/.vimrc
% cvs diff -Nu shazam | vim -
% cvs log shazam | vim -
```

3. CVS è vecchio, arcano, complesso e buggato, e a volte esibisce comportamenti non deterministici che qualcuno sostiene siano la prova che CVS non sia niente di più di una manifestazione Newtoniana di una entità ultradimensionale sensibile. Non è umanamente possibile conoscere ogni dettaglio di CVS, quindi non essere dispiaciuto di chiedere aiuto all'Intelligenza Artificiale (CVS Repository Meisters <cvsadm@FreeBSD.org >).

4. Non lasciare il comando `cvs commit` nella modalità di inserimento del messaggio di commit per troppo tempo (più di 2-3 minuti). Questo blocca la directory in cui stai lavorando ed impedirà ad altri sviluppatori di effettuare commit nella stessa directory. Se devi digitare un messaggio di commit lungo, scrivilo prima di eseguire `cvs commit` e inseriscilo successivamente oppure salvalo in un file prima di effettuare il commit ed usa l'opzione `-F` di CVS per leggere il messaggio di commit da quel file, cioè:

   ```
   % vi logmsg
   % cvs ci -F logmsg shazam
   ```

   Questo è il metodo più veloce per passare un messaggio di commit a CVS ma devi stare attento quando modifichi il file `logmsg` prima del commit, perché CVS non ti darà la possibilità di modificare il messaggio quando effettuerai realmente il commit.

4. Convenzioni e Tradizioni

Come nuovo committer ci sono alcune cose che dovresti fare all'inizio.

- Aggiungi la tua entity di autore in `doc/en_US.ISO8859-1/share/xml/authors.ent` ; questo dovrebbe essere fatto per prima cosa, visto che l'omissione di questo commit farà in modo che i prossimi commit romperanno la compilazione del ramo doc/.

  Questo è un compito relativamente semplice, ma rimane una buona prima prova delle tue abilità con CVS.

- Aggiungi te stesso alla sezione «Developers» della Contributors List e rimuovere te stesso dalla sezione «Additional Contributors».

- Aggiungi una voce per te stesso in `www/en/news/news.xml` . Guarda le altre voci che assomigliano a «A new committer» e segui il formato.

- Dovresti aggiungere la tua chiave PGP o GnuPG in `doc/share/pgpkeys` (e se non ce l'hai, dovresti creartene una). Non dimenticare di effettuare il commit del file `doc/share/pgpkeys/pgpkeys.ent` aggiornato.

  Dag-Erling C. Smørgrav ha scritto uno script di shell per rendere questa operazione molto semplice. Guarda il file README per maggiori informazioni.

  Nota

  È importante avere una chiave PGP/GnuPG aggiornata nel Manuale, visto che potrà essere richiesta per l'identificazione del committer, ad esempio dai FreeBSD Administrators <admins@FreeBSD.org> per il recupero dell'account. Un portachiavi completo degli utenti `FreeBSD.org` è disponibile su http://www.FreeBSD.org/doc/pgpkeyring.txt.

- Alcune persone aggiungono una voce per se stessi in `ports/astro/xearth/files/freebsd.committers.markers` .

- Alcune persone aggiungono una voce per se stessi in `src/usr.bin/calendar/calendars/calendar.freebsd` .

- Presentati agli altri committer, altrimenti nessuno avrà idea di chi tu sia o di cosa ti occupi. Non devi scrivere una biografia completa, basta un paragrafo o due su chi sei e su quello di cui hai intenzione di occuparti come committer di FreeBSD. Invialo alla mailing list degli sviluppatori di FreeBSD e sarai sulla strada giusta!

- Loggati su `hub.FreeBSD.org` e crea un file `/var/forward/utente` (dove *`utente`* è il tuo nome utente) contenente l'indirizzo e-mail dove vuoi che i messaggi indirizzati a *`tuonomeutente`* @FreeBSD.org siano inoltrati. Questo include tutti i messaggi di commit così come ogni altro messaggio inviato alla mailing list dei committer di FreeBSD e alla mailing list degli sviluppatori di FreeBSD. Caselle di posta veramente grandi che hanno preso residenza fissa su `hub` spesso vengono «accidentalmente» troncate senza preavviso, quindi inoltra o leggi i messaggi in modo da non perderli.

  A causa dell'intenso carico per la gestione dello SPAM che arriva ai server di posta centrali che processano le mailing list, i server front-end fanno alcuni controlli e non fanno passare alcuni messaggi in base a questi controlli. Al momento l'unico controllo attivo è la verifica sulla correttezza delle informazioni DNS dell'host che si connette, ma questo potrebbe cambiare. Alcune persone accusano questi controlli di respingere email valide. Se vuoi disabilitare questi controlli per la tua email puoi creare un file chiamato `~/.spam_lover` nella tua directory home su `freefall.FreeBSD.org` .

- Se sei iscritto alla mailing list con i messaggi di commit sul CVS di FreeBSD, probabilmente vorrai disiscriverti per evitare di ricevere copie doppie dei messaggi di commit e della loro evoluzione.

Tutti i nuovi committer hanno un mentore assegnato a loro per i primi mesi. Il tuo mentore è responsabile di insegnarti le regole e le convenzioni del progetto e guidare i tuoi primi passi nella comunità dei committer. È anche personalmente responsabile delle tue azioni durante questo periodo iniziale. Fino a quando il tuo mentore non decide (e lo annuncia con un commit forzato su `access`) che sei diventato pratico e pronto per effettuare commit da solo, non dovresti effettuare commit senza aver prima ottenuto la revisione e l'approvazione del tuo mentore, e dovresti documentare l'approvazione con una riga `Approved by:` nel messaggio di commit.

Tutti i commit `src` dovrebbero andare su FreeBSD-CURRENT prima di essere fusi in FreeBSD-STABLE. Nessuna nuova caratteristica importante o modifica ad alto rischio dovrebbe essere fatta sul ramo FreeBSD-STABLE.

5. Licenza Preferita per i Nuovi File

Attualmente il FreeBSD Project suggerisce di usare il seguente testo come schema di licenza preferito:

```
Copyright © <Year> <Author>. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in ↺
the
   documentation and/or other materials provided with the ↺
distribution.

THIS SOFTWARE IS PROVIDED BY AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR ↺
PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR ↺
CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE ↺
GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, ↺
STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ↺
ANY WAY
```

```
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY ↺
OF
SUCH DAMAGE.
```

Il progetto FreeBSD scoraggia fortemente la cosiddetta clausola pubblicitaria nel nuovo codice. A causa del grande numero di contributi al progetto FreeBSD, osservare questa clausola per molti fornitori commerciali è diventato difficile. Se hai codice nell'albero con la clausola pubblicitaria, pensa a rimuoverla. In pratica, considera di usare la licenza qui sopra per il tuo codice.

Il progetto FreeBSD scoraggia completamente nuove licenze e variazioni sulle licenze standard. Nuove licenze richiedono l'approvazione di <core@FreeBSD.org > per risiedere nel repository principale. Più licenze differenti vengono usate nell'albero, più problemi possono essere causati a chi desidera utilizzare questo codice, tipicamente da conseguenze non previste di una licenza strutturata male.

6. Relazioni tra Sviluppatori

Se stai lavorando direttamente sul tuo codice o su codice che è già stabilito essere di tua responsabilità, allora c'è probabilmente poca necessità di confrontarsi con altri committer prima di effettuare un commit. Se vedi un bug in un'area del sistema che è chiaramente orfana (e ce n'è qualcuna di queste aree, per nostra vergogna), agisci allo stesso modo. Se, tuttavia, stai per modificare qualcosa che è chiaramente mantenuto attivamente da qualcun'altro (ed è solo guardando la mailing list `cvs-committers` che puoi veramente sapere cosa è e cosa non è) allora invia le modifiche a lui, come avresti fatto prima di diventare committer. Per i port, dovresti contattare il `MAINTAINER` specificato nel `Makefile`. Per altre parti del repository, se non sei sicuro di chi possa essere il maintainer attivo, potrebbe essere utile scorrere l'output di `cvs log` per vedere chi ha effettuato delle modifiche in passato. Bill Fenner ha scritto un utile script di shell che può aiutare a determinare chi sia il maintainer attivo. Questo elenca ogni persona che ha effettuato commit su un file specifico con il numero di commit che ha fatto. Può essere trovato su `freefall` in `~fenner/bin/whodid`. Se alle tue richieste non corrisponde una risposta o se il committer in altro modo dimostra uno scarso interesse nell'area oggetto della modifica, vai avanti ed effettua il commit tu stesso.

Se non sei sicuro di un commit per qualunque motivo, fallo revisionare da `-hackers` prima di effettuare il commit. Meglio che sia criticato lì piuttosto che quando è parte del repository CVS. Se ti capita di effettuare un commit che provoca controversie, potresti voler considerare l'annullamento delle modifiche finché il problema sia chiarito. Ricorda - con CVS possiamo sempre tornare indietro.

Non mettere in dubbio le intenzioni di qualcuno che non è d'accordo con te. Se vedono una soluzione differente dalla tua per un problema, o anche un problema diverso, non è perché sono stupidi, perché hanno una dubbia origine, o perché stanno cercando di

distruggere il tuo duro lavoro, la tua immagine personale, o FreeBSD, ma semplicemente perché hanno una visione differente del mondo. La diversità è una buona cosa.

Dissenti onestamente. Argomenta la tua posizione con i suoi meriti, sii onesto sui difetti che può avere, e sii disponibile a guardare le loro soluzioni, o anche le loro visioni del problema, con mente aperta.

Accetta le correzioni. Possiamo tutti sbagliare. Se hai fatto un errore, scusati e vai avanti con la tua vita. Non picchiarti, e sicuramente non picchiare gli altri per il tuo sbaglio. Non sprecare tempo imbarazzandoti o recriminando, risolvi solo il problema e vai avanti.

Chiedi aiuto. Cerca (e dai) revisioni dagli altri. Uno delle cose in cui dovrebbe eccellere il software open source è il numero di occhi che lo scrutano; questo non è vero se nessuno revisionerà il codice.

7. GNATS

Il FreeBSD Project utilizza GNATS per gestire i bug e le richieste di cambiamenti. Assicurati di usare `edit-pr numero-pr` su `freefall` quando effettui il commit di una correzione o di un suggerimento trovato in un PR GNATS per chiuderlo. È inoltre considerato gentile se trovi il tempo di chiudere ogni PR associato al tuo commit, se esistono. Puoi anche usare send-pr(1) tu stesso per proporre qualsiasi cambiamento che pensi debba essere fatto, a seguito di una maggiore revisione da parte di altre persone.

Puoi trovare di più su GNATS su:

- http://www.cs.utah.edu/csinfo/texinfo/gnats/gnats.html
- http://www.FreeBSD.org/support.html
- send-pr(1)

Puoi far girare una copia locale di GNATS, e poi integrare l'albero GNATS di FreeBSD in esso tramite CVSup. In seguito puoi usare i comandi GNATS localmente, o usare altre interfacce, come `tkgnats`. Questo ti permette di interrogare il database dei PR senza bisogno di essere connesso a Internet.

Procedura 1. Utilizzo di un albero GNATS locale

1. Se non stai già scaricando l'albero GNATS, aggiungi questa riga al tuo `supfile`, e riesegui cvsup(1). Nota che siccome GNATS non è sotto il controllo di CVS non ha tag, quindi se lo stai aggiungendo al tuo `supfile` esistente deve apparire prima di ogni voce «tag=» dato che queste rimangono attive una volta impostate.

   ```
   gnats release=current prefix=/usr
   ```

Questo metterà l'albero GNATS di FreeBSD in `/usr/gnats`. Puoi usare un file *refuse* per controllare quali categorie ricevere. Per esempio, per ricevere solo i PR `docs`, metti questa riga in `/usr/local/etc/cvsup/sup/refuse` [1].

```
gnats/[a-ce-z]*
```

Il resto di questi esempi assume che tu abbia scaricato solo la categoria `docs`. Modificali quando è necessario, a seconda delle categorie che tieni in sincronia.

2. Installa il port GNATS da `ports/databases/gnats`. Questo metterà le varie directory GNATS sotto `$PREFIX/share/gnats`.

3. Crea un symlink per le directory GNATS che aggiorni tramite CVSup sotto la versione di GNATS che hai installato.

```
# cd /usr/local/share/gnats/gnats-db
# ln -s /usr/gnats/docs
```

Ripeti tante volte quanto necessario, a seconda di quante categorie GNATS tieni in sincronia.

4. Aggiorna il file `categories` di GNATS con queste categorie. Il file è `$PREFIX/share/gnats/gnats-db/gnats-adm/categories`.

```
# Questa categoria è obbligatoria
pending:Categoria per i PR errati:gnats-admin:
#
# Categorie di FreeBSD
#
docs:Bug di Documentazione:freebsd-doc:
```

5. Esegui `$PREFIX/libexec/gnats/gen-index` per ricreare l'indice GNATS. L'output deve essere reindirizzato su `$PREFIX/share/gnats/gnats-db/gnats-adm/index`. Puoi fare questo periodicamente da cron(8), o eseguire cvsup(1) da uno script di shell che fa anche questo.

```
# /usr/local/libexec/gnats/gen-index \
        > /usr/local/share/gnats/gnats-db/gnats-adm/index
```

6. Verifica la configurazione interrogando il database dei PR. Questo comando visualizza i PR `docs` aperti.

```
# query-pr -c docs -s open
```

Anche altre interfacce, come quella fornita dal port databases/tkgnats, dovrebbero funzionare correttamente.

[1] Il percorso preciso dipende dall'impostazione `*default base` nel tuo `supfile`.

7. Prendi un PR e chiudilo.

Nota

Questa procedura funziona solo per permetterti di visualizzare ed interrogare i PR localmente. Per modificarli o chiuderli dovrai ancora loggarti su `freefall` e farlo da lì.

8. Chi è Chi

Oltre ai meister del repository, ci sono altri membri e team del FreeBSD Project che probabilmente arriverai a conoscere nel tuo ruolo di committer. Brevemente, e senza pretesa di elencarli tutti, questi sono:

John Baldwin
: John è il manager dell'SMPng Project, e ha autorità sulla progettazione architetturale e sull'implementazione del passaggio a un sistema di threading e locking del kernel a grana fine. È anche l'autore dell'SMPng Architecture Document. Se stai lavorando sullo stesso sistema, coordinati con John.

Jake Burkholder, Thomas Möstl
: Jake e Thomas sono i maintainer del port sull'architettura SPARC64®.

Documentation Engineering Team <doceng@FreeBSD.org >
: doceng è il gruppo responsabile dell'infrastruttura per la realizzazione della documentazione, approva i nuovi committer della documentazione, e assicura che il sito web di FreeBSD e la documentazione sul sito FTP siano aggiornati rispetto all'albero CVS. Non è un organo di risoluzione dei conflitti. La maggior parte delle discussioni relative alla documentazione prendono posto sulla mailing list sul progetto di documentazione di FreeBSD. Maggiori dettagli riguardanti il team doceng possono essere trovati nel suo statuto. I committer interessati a contribuire alla documentazione dovrebbero familiarizzare con il Documentation Project Primer.

Ruslan Ermilov
: Ruslan è Mister mdoc(7). Se stai scrivendo una pagina man e hai bisogno di qualche suggerimento sulla struttura, o sul linguaggio di markup, chiedi a Ruslan.

Bruce Evans
: Bruce è lo Style Police-Meister. Quando fai un commit che poteva essere fatto meglio, Bruce sarà lì a dirtelo. Ringrazia che qualcuno lo sia. Bruce conosce anche molto bene gli standard applicabili a FreeBSD.

Andrew Gallatin, Matthew Jacob, Doug Rabson, David O'Brien
: Questi sono gli sviluppatori e i supervisori primari della piattaforma DEC Alpha AXP.

David Greenman
: David è il supervisore del sistema VM. Se hai in mente una modifica al sistema VM, coordinala con David.

Doug Rabson, Marcel Moolenaar, Peter Wemm, Paul Saab
: Questi sono i principali sviluppatori e supervisori della piattaforma Intel IA-64, ufficialmente conosciuta come l'Itanium® Processor Family (IPF).

Murray Stokely, Steve Price, Robert Watson, John Baldwin, Scott Long, Ken Smith, Hiroki Sato
: Questi sono i membri del Release Engineering Team <re@FreeBSD.org >. Questo team è responsabile di decidere i tempi delle release e controllare il processo di release. Durante i periodi di congelamento del codice, gli ingegneri di release hanno l'autorità finale su tutte le modifiche al sistema per quel ramo di cui si sta preparando la release. Se c'è qualcosa che vuoi sia fuso da FreeBSD-CURRENT a FreeBSD-STABLE (qualsiasi valore queste possano avere in un dato momento), queste sono le persone con cui devi parlare.

  Hiroki è anche l'autore della documentazione di release (`src/release/doc/*`). Se effettui il commit di una modifica che pensi sia degna di menzione nelle note di release, assicurati che lo sappia. Meglio ancora, inviagli una patch con il tuo commento.

Benno Rice
: Benno è il maintainer ufficiale del port per PowerPC®.

Brian Somers
: Maintainer ufficiale di `/usr/sbin/ppp` .

Jacques Vidrine
: Jacques è il FreeBSD Security Officer e supervisiona il Security Officer Team <security-officer@FreeBSD.org >.

Garrett Wollman
: Se hai bisogno di consigli sulle oscure parti interne delle reti o non sei sicuro di qualche eventuale modifica al sottosistema di rete che hai in mente, Garrett è qualcuno con cui parlare. Garret è inoltre molto esperto sui vari standard applicabili a FreeBSD.

mailing list dei committer di FreeBSD
: cvs-committers è l'entità che CVS usa per inviarti tutti i messaggi di commit. Non devi *mai* inviare email direttamente a questa lista. Puoi solamente rispondere a questa lista quando i messaggi sono brevi e direttamente correlati a un commit.

mailing list degli sviluppatori di FreeBSD
: Tutti i committer sono iscritti a -developers. Questa lista è stata creata per essere un forum sulle questioni della «comunità» dei committer. Esempi sono le votazioni per

il Core, annunci, ecc. Questa lista *non* è intesa come posto per la revisione del codice o come rimpiazzo della mailing list sull'architettura e la progettazione di FreeBSD o della mailing list di controllo del codice sorgente di FreeBSD. Infatti usarla in questo modo urta il FreeBSD Project dato che dà l'impressione di una lista privata dove vengono prese le decisioni generali che influenzano tutta la comunità che usa FreeBSD senza essere rese «pubbliche». Ultimo, ma non per importanza *mai e poi mai invia un messaggio alla mailing list degli sviluppatori di FreeBSD mettendo in CC:/BCC: un'altra lista FreeBSD*. Mai e poi mai invia un messaggio su un'altra mailing list mettendo in CC:/BCC: la mailing list degli sviluppatori di FreeBSD. Fare questo può diminuire enormemente i benefici di questa lista. Inoltre, non pubblicare o inoltrare mai email inviate alla mailing list degli sviluppatori di FreeBSD. L'atto di inviare un messaggio alla mailing list degli sviluppatori di FreeBSD anziché a una lista pubblica significa che le informazioni contenute non sono ad uso pubblico.

9. Guida Rapida a SSH

1. Se stai usando FreeBSD 4.0 o successivo, OpenSSH è incluso nel sistema base. Se stai usando una release precedente, aggiorna ed installa uno dei port di SSH. In generale, probabilmente vorrai prendere OpenSSH dal port security/openssh. Potresti anche voler estrarre l'ssh1 originale dal port security/ssh, ma sii certo di porre la dovuta attenzione alla sua licenza. Nota che questi port non possono essere installati contemporaneamente.

2. Se non vuoi digitare la tua password ogni volta che usi ssh(1), e usi chiavi RSA o DSA per autenticarti, ssh-agent(1) è lì per la tua comodità. Se vuoi usare ssh-agent(1), assicurati di eseguirlo prima di utilizzare altre applicazioni. Gli utenti X, per esempio, solitamente fanno questo dal loro file `.xsession` o `.xinitrc`. Guarda ssh-agent(1) per i dettagli.

3. Genera un paio di chiavi con ssh-keygen(1). Le chiavi finiranno nella tua directory `$HOME/.ssh/` .

4. Invia la tua chiave pubblica (`$HOME/.ssh/id_dsa.pub` o `$HOME/.ssh/id_rsa.pub`) alla persona che ti sta configurando come committer in modo che possa inserirla nel file `tualogin` su `freefall`.

Ora dovresti essere in grado di usare ssh-add(1) per autenticarti una volta a sessione. Ti verrà richiesta la pass phrase della tua chiave privata, e quindi verrà salvata nel tuo agente di autenticazione (ssh-agent(1)). Se non vuoi più avere la tua chiave salvata nell'agente, l'esecuzione di `ssh-add -d` la rimuoverà.

Verifica facendo qualcosa come `ssh freefall.FreeBSD.org ls /usr` .

Per maggiori informazioni, guarda security/openssh, ssh(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), e scp(1).

10. Il Lungo Elenco di Regole dei Committer di FreeBSD

Traduzione in corso

11. Supporto per Diverse Architetture

Traduzione in corso

12. FAQ Specifiche sui Port

Traduzione in corso

13. Benefici del Lavoro

Sfortunatamente, non ci sono molti benefici derivanti dall'essere un committer. Il riconoscimento di essere un progettista di software competente è probabilmente l'unica cosa che sarà di tuo vantaggio a lungo termine. Ciononostante, ci sono comunque alcuni benefici:

Accesso diretto al `cvsup-master`
: Come committer, puoi chiedere a Jun Kuriyama accesso diretto a `cvsup-master.FreeBSD.org` , fornendo l'output della tua chiave pubblica tramite `cvpasswd yourusername@FreeBSD.org freefall.FreeBSD.org` . Nota: devi specificare `freefall.FreeBSD.org` sulla riga di comando si `cvpasswd` anche se il server attuale è `cvsup-master`. L'accesso al `cvsup-master` non dovrebbe essere abusato visto che è una macchina carica di lavoro.

Un abbonamento gratuito al set da 4 CD o DVD
: FreeBSD Mall, Inc. offre un abbonamento gratuito al set da 4 CD o DVD a tutti i committer di FreeBSD. Le informazioni su come ottenere il prodotto gratuitamente vengono spedite a <`developers@FreeBSD.org` > dopo ogni release.

14. Domande Generali

Traduzione in corso