PATH HOWTO

Esa Turtiainen etu@dna.fi

v0.4, 15 Novembre 1997

Traduzione a cura di Silvio Donnini e-mail: scaudi at alice dot it Revisione a cura di Giulio Daprelï¿½e-mail: giulio at pluto dot it

Contents

1 Introduzione **2**

2 Copyright **2**

3 Note Generali **3**

4 Init **4**

5 Il Login **4**

6 Le Shell **5**

6.1 Bash 5

6.2 tcsh 6

7 Cambiare l'ID dell'utente **6**

7.1 su 6

7.2 sudo 7

8 Server di rete **7**

8.1 inetd 7

8.2 rsh 7

8.3 rlogin 8

8.4 telnet 8

8.5 ssh 8

9 XFree86 **8**

9.1 XDM 8

9.2 Xterm -ls 9

9.3 Menu e pulsanti del window manager 9

10 Comandi ritardati: cron e at 9

10.1 cron 9

10.2 at 10

11 Alcuni esempi 10

11.1 magicfilter 10

11.2 Stampare da applicazioni X 10

12 Problematiche di sicurezza 11

13 Come individuare i problemi? 11

14 Alcuni strategie per avere lo stesso PATH per tutti gli utenti 12

15 Riconoscimenti 13

1 Introduzione

Questo documento descrive trucchi e problemi comuni per le variabili d'ambiente di Unix / Linux, in particolare per la variabile PATH. PATH è una lista di directory in cui il sistema cerca i comandi. I dettagli riguardano la distribuzione Debian Linux 1.3.

Nota: questo documento è in versione beta. Sono gradite correzioni e commenti.

2 Copyright

3 Note Generali

Tutti i processi Unix contengono un "ambiente". Questo è una lista di variabili che contengono nome e rispettivo valore, entrambi sotto forma di stringhe che possono contenere la maggior parte dei caratteri. Ogni processo Unix ha un processo padre che lo ha creato. Un processo figlio eredita l'ambiente dal processo padre ed esso stesso può fare delle modifiche al proprio ambiente per poi passare a sua volta tale ambiente modificato ai figli.

Una variabile d'ambiente importante è PATH, una lista di directory separate da caratteri due punti (':'). Il sistema cerca in queste directory per trovare i comandi da eseguire. Se si prova ad invocare il comando "foo", il sistema cerca un file eseguibile (con permessi di esecuzione) di nome "foo" in tutte le directory che trova in PATH, nell'ordine in cui vi sono scritte. Se un file del genere è trovato esso viene eseguito.

In questo howto uso il termine "comando" riferendomi ad un programma eseguibile che si intende invocare attraverso questo meccanismo (senza fornire il percorso completo).

In Linux anche le chiamate di sistema di basso livello che servono ad avviare i processi (la famiglia di chiamate exec) cercano gli eseguibili nelle directory indicate nella variabile PATH: si può usare il meccanismo di path ovunque ogni volta che si prova ad eseguire un comando. Se una chiamata di sistema "exec" ha per argomento un nome di file che non contiene il carattere '/' il sistema prova ad usare la variabile d'ambiente PATH. Se nell'ambiente non è presente tale variabile l'eseguibile viene cercato (almeno) nelle directory /bin e /usr/bin.

In sh viene utilizzato il comando "export" per impostare una variabile d'ambiente, in csh il comando equivalente è "setenv". Ad esempio:

sh:

```
PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:.
```

csh:

```
setenv PATH /usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games:.
```

I programmi C possono usare la funzione di libreria setenv() per modificare l'ambiente. In Perl l'ambiente è contenuto nell'array associativo %ENV , PATH può essere impostata semplicemente ad esempio così $ENV{PATH}="/bin".

Il comando "env" è il modo più semplice di visualizzare le variabili d'ambiente correnti. Esso può essere usato anche per modificarle.

Ulteriori informazioni sul meccanismo dell'ambiente sono contenute nelle pagine del manuale "environ", "execl", "setenv", nel file info "env" e nella documentazione delle shell.

All'avvio di Linux, il primo processo normale che viene avviato è init. È un processo particolare, perché non ha un padre, ma tutti gli altri processi derivano da esso. L'ambiente di init è quello che tutti i processi avrebbero se non modificassero il proprio ambiente, ma la maggior parte fa qualche cambiamento.

Init fa partire un gruppo di processi. Il file /etc/inittab dice al sistema quali processi eseguire. Questi processi girano in un ambiente direttamente ereditato da init, tipicamente sono processi come "getty", il programma che scrive "login:" sulla console. Se si fa partire una connessione PPP a questo punto, bisogna ricordare che si sta lavorando nell'ambiente di init. L'inizializzazione del sistema è spesso uno script fatto partire in questa fase. In Debian 1.3 lo script di inizializzazione è /etc/init.d/rc e chiama altri script a sua volta.

Il sistema contiene molti server (demoni) che possono o non possono utilizzare l'ambiente predefinito. La maggior parte dei server sono fatti partire dagli script di inizializzazione e quindi hanno l'ambiente di init.

Nel momento in cui l'utente accede al sistema, l'ambiente è influenzato dalle impostazioni compilate nei programmi, dagli script di inizializzazione del sistema e dagli script di inizializzazione specifici dell'utente. È un meccanismo abbastanza complicato e la situazione corrente non è del tutto soddisfacente. Per l'utente cambia tutto tra il fare il login da console testuale, da XDM o da rete.

4 Init

Init è il processo padre di tutti gli altri processi del sistema. Gli altri processi ereditano l'ambiente del processo init e il loro path è il path iniziale, nei rari casi in cui non viene impostato un altro path. Il path predefinito ("init path") è compilato nel codice sorgente del programma init ed è:

```
/usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin
```

Si noti che l' init path non contiene /usr/local/bin .

Tutti i programmi elencati in /etc/inittab lavrano nell'ambiente di init, in particolare gli scrpit di inizializzazione del sistema in /etc/init.d (Debian 1.3).

Tutto ciò che viene avviato dagli script di inizializzazione del sistema ha per ambiente predefinito quello di init. Ad esempio syslogd, kerneld, pppd (se fatti partire all'avvio), gpm e soprattutto lpd e inetd hanno l'ambiente di init e non lo cambiano.

Un grupo di programmi è fatto partire dagli script di avvio, ma la variabile d'ambiente PATH è impostata esplicitamente nello script di avvio. Ad esempio così fanno atd, sendmail, apache e squid.

Ci sono altri programmi fatti partire da script di avvio ma essi cambiano il path completamente; uno di questi è cron.

5 Il Login

In una console di testo c'è un programma, getty, che aspetta che l'utente faccia il login. Esso scrive "login:" e altri messaggi. Lavora nell'ambiente di init. Quando un utente si autentica, getty invoca il programma "login". Esso imposta l'ambiente utente e invoca la shell.

Il programma login imposta PATH secondo quanto definito in /usr/include/paths.h. Per l'utente root questi percorsi sono diversi da quelli degli altri utenti.

per gli utenti comuni (_PATH_DEFPATH):

```
/usr/local/bin:/usr/bin:/bin:.
```

per root (_PATH_DEFPATH_ROOT):

```
/sbin:/bin:/usr/sbin:/usr/bin
```

Il path di un utente comune non contiene nessuna directory sbin. Tuttavia esso contiene la directory corrente '.', che è considerata pericolosa per l'utente root. Nemmeno la directory /usr/local/bin è disponibile per l'utente root.

Il PATH ottenuto al momento del login viene spesso sostituito da quello definito negli script di inizializzazione delle shell. Comunque è possibile usare altri programmi in /etc/passwd come shell per l'utente. Ad esempio, io ho usato la seguente riga per far partire PPP quando accedo al sistema con un nome utente speciale. In questo caso, pppd ha esattamente il PATH definito al momento del login.

```
etu-ppp:viYabVlxPwzDl:1000:1000:Esa Turtiainen, PPP:/:/usr/sbin/pppd
```

6 Le Shell

Spesso i processi dell'utente sono figli della shell menzionata in /etc/passwd per quel particolare utente. I file di inizializzazione delle shell spesso modificano la variabile PATH.

Al login, il nome della shell è preceduto dal carattere '-', ad esempio bash è chiamata "-bash". Questo per indicare alla shell che è una shell di login (login shell). In tal caso, la shell esegue i file di inizializzazione richiesti dal login. Altrimenti esegue qualche inizializzazione meno impegnativa. In più la shell controlla se essa stessa è interattiva e cioè se i comandi vengono da file o da terminale tty. Anche ciò influisce sull'inizializzazione, cosicché una shell non interattiva e non di login viene inizializzata molto poco (bash non esegue alcun codice di inizializzazione in tale caso).

6.1 Bash

Se avviata come normale login shell, bash legge il file globale /etc/profile, dove può essere specificato l'ambiente di sistema e la variabile PATH. Tuttavia, tale file non viene considerato se il sistema interpreta la shell come non interattiva. Il caso più importante è quello di rsh, in cui un comando è eseguito in una macchina comunicante. Il file /etc/profile non viene letto e PATH è ereditata dal demone rsh.

Bash riceve dalla riga di comando gli argomenti -login e -i che possono essere usati rispettivamente per impostare la shell rispettivamente come login shell o shell interattiva.

L'utente può sostituire i valori impostati in /etc/profile creando un file ~/.bash_profile, ~/.bash_login o ~/.profile. Si noti che solo il primo di questi viene eseguito, deviando dalla logica della convenzionale inizializzazione di csh. ~/.bash_login non viene eseguito in maniera particolare per le login shell e se esiste il file .bash_profile esso non viene eseguito affatto!

Se si usa bash con il nome "sh" invece del nome "bash", il programma emula l'inizializzazione della Bourne shell originale: legge solo i file di configurazione /etc/profile e ~/.profile e solo per le login shell.

6.2 tcsh

Se eseguita come login shell, tcsh esegue i file nel seguente ordine:

- /etc/csh.cshrc
- /etc/csh.login
- ~/.tcshrc
- ~/.cshrc (if .tcshrc is not found)
- ~/.history
- ~/.login
- ~/.cshdirs

Si noti che tcsh può essere compilata per eseguire script di login prima degli script cshrc.

Le shell non interattive eseguono solo gli script *cshrc . Gli script *login possono essere usati per impostare PATH solo una volta al momento del login.

7 Cambiare l'ID dell'utente

7.1 su

Il comando su serve per cambiare l'id dell'utente. Se non è specificato nessun id per argomento, viene usato root.

Normalmente su invoca una subshell come login shell con un user id differente e con argomento '-' (sinonimi più recenti sono -l o –login). Tuttavia esso non fa uso del programma login per fare ciò ed utilizza ancora un altro PATH differente per "simulare" il login. Questo PATH è:

per utenti normali

```
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:.
```

per l'utente root

```
/sbin:/bin:/usr/sbin:/usr/bin:/usr/bin/X11:/usr/local/sbin:/usr/local/bin
```

su apporta molti altri cambiamenti complessi all'ambiente.

7.2 sudo

Esiste un gruppo di comandi che fa uso dei comandi del super utente in maniera più sicura. Questi prevedono un miglior meccanismo di login, restrizioni basate sull'identità dell'utente e l'uso di password individuali. Il più usato è sudo.

```
$ sudo env
```

esegue il comando env come super utente (se ï¿½configurato per permetterlo).

Il comando sudo ha ancora un altro approccio diverso alla gestione di PATH. Esso modifica la strategia di ricerca nei percorsi facendo in modo che la directory corrente sia sempre esaminata per ultima. Comunque non modifica la variabile PATH in sé. "sudo env" e "env" restituiscono lo stesso valore per la variabile PATH. Sudo aggiunge solo un paio di variabili d'ambiente come SUDO_USER.

8 Server di rete

La maggior parte dei server di rete non dovrebbe invocare sottoprocessi di alcun tipo. Per motivi di sicurezza il PATH di tali server dovrebbe essere minimale.

Un'eccezione importante riguarda tutti i servizi che permettono di fare il login del sistema dalla rete. Questa sezione descrive la struttura dell'ambiente in questi casi. Se il comando viene eseguito sulla macchina remota, il suo PATH è diverso a seconda che si stia utilizzando rsh o ssh. Allo stesso modo fare il login con rlogin, Telnet o ssh produce diversi risultati.

8.1 inetd

La maggior parte dei server di rete non ha processi propri che attendono le richieste. Questo compito è delegato ad un super server chiamato inetd. Inetd ascolta tutte le porte specificate e fa partire il server appropriato quando riceve una richiesta. Questo comportamento è definito in /etc/inetd.conf.

inetd è avviato dagli script di inizializzazione del sistema. Esso eredita solo il PATH del processo init, non lo modifica e tutti i server fatti partire da inetd hanno l'init path. Un esempio di tale server è imapd, il server del protocollo IMAP.

Altri esempi di processi inetd sono telnetd, rlogind, talkd, ftp, popd, molti server http e così via.

Spesso l'utilizzo di inetd è complicato ulteriormente da un programma tcpd separato che fa partire il server reale. È un programma che esegue ulteriori controlli di sicurezza prima di far partire l'applicazione vera e propria. Non fa modifiche a PATH (non verificato).

8.2 rsh

Il demone rsh imposta PATH da _PATH_DEFPATH (/usr/include/paths.h), che è lo stesso PATH che usa il programma login per gli utenti comuni. Root avrà lo steso PATH di un utente normale.

In realtà, rshd esegue il comando che gli viene fornito da riga di comando:

```
shell -c command-line
```

e ”shell” qui non è una login shell. È desiderabile che tutte le shell menzionate in /etc/passwd supportino l’opzione -c da riga di comando.

8.3 rlogin

Rlogin invoca login per eseguire la vera procedura di login. Se si esegue il login da rlogin, si ottiene la stessa variabile PATH che con login. La maggior parte degli altri modi per fare il login su una macchina Linux non usa login. Si noti la differenza con rsh.

Per la precisione il comando login usato è

```
login -p -h host-name user-name
```

-p preserva l’ambiente, tranne le variabili HOME, PATH, SHELL, TERM, MAIL e LOGNAME. -h indica il nome dell’host remoto su cui fare il login.

8.4 telnet

Telnet è simile a rlogin. Esso usa il programma login e la riga di comando per invocarlo in maniera analoga.

8.5 ssh

ssh ha una configurazione di PATH tutta sua. Ha un PATH precompilato a cui aggiunge la directory in cui si trova ssh. Ciò spesso significa che /usr/bin è presente due volte nel PATH:

```
/usr/local/bin:/usr/bin:/bin:.:/usr/bin
```

Il path non contiene /usr/X11/bin e la shell invocata dal comando ssh non è una login shell. Quindi

```
ssh remotehost xterm
```

non funziona mai e qualunque cosa sia scritta in /etc/profile e /etc/csh.cshrc non può cambiare questa situazione. Si deve sempre usare il percorso specifico /usr/bin/X11/xterm.

ssh cerca delle variabili d’ambiente nella forma VARIABILE=VALORE nel file /etc/environment. Sfortunatamente ciò causa qualche problema con XFree86.

9 XFree86

9.1 XDM

XDM è il modo più comune di accedere ad un terminale grafico. È simile in aspetto a login, ma il suo funzionamento interno è del tutto differente.

Nella directory /etc/X11/xdm ci sono dei file di configurazione che sono eseguiti in fasi diverse dell'operazione di login. Xstartup (e Xstartup_0 specificamente per lo screen 0) contiene comandi che vanno fatti partire dopo che l'utente ha fatto il login (tali comandi sono fatti girare con i privilegi di root).

Il PATH che viene impostato per gli utenti è in /etc/X11/xdm/xdm-config. Ci sono righe:

```
DisplayManager*userPath: /usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
DisplayManager*systemPath: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/bin/X11
```

che indicheranno un PATH predefinito per utenti normali e per root rispettivamente. È molto importante che /usr/bin/X11 sia disponibile per gli utenti di X. Se un utente di X fa il login su un'altra macchina per far partire un'applicazione X, dovrebbe ottenere /usr/bin/X11 all'interno del suo PATH anche se non sembra venire direttamente dal terminale X.

Dopo aver fatto partire xstartup XDM fa partire /etc/X11/Xsession che è fatto partire con i privilegi dell'utente finale. La configurazione locale è pensata per essere fatta in /etc/environment, che viene letto (incluso) da Xsession, se quest'ultimo è disponibile (Xsession viene eseguito da /bin/sh e quindi /etc/environment deve essere un file per sh). Questa scelta va in conflitto con quella di ssh, che suppone che /etc/environment sia un file che contiene righe formattate come VARIABILE=VALORE.

9.2 Xterm -ls

Il PATH predefinito per tutti i comandi invocati dai menu del window manager X è il PATH ereditato da XDM. Se si vuole usare un altro PATH bisogna specificarlo esplicitamente. Per far partire un emulatore di terminale con un PATH che sia "normale" bisogna usare qualche opzione particolare. In xterm deve essere usata l'opzione -ls (login shell) per ottenere una login shell con PATH specificato come nei file di inizializzazione del login.

9.3 Menu e pulsanti del window manager

Il window manager eredita l'ambiente di XDM. Tutti i programmi che vengono fatti partire dal window manager ereditano l'ambiente del window manager.

L'ambiente della shell dell'utente non influenza i programmi che sono fatti partire da menu e da pulsanti del window manager. Per esempio, se un programma è fatto partire da "xterm -ls", ha l'ambiente predefinito di una shell di login, ma se è fatto partire da menu ha solo l'ambiente del window manager.

10 Comandi ritardati: cron e at

10.1 cron

Cron è un comando che esegue altri comandi periodicamente come specificato in /etc/crontab e altri crontab definiti dall'utente. In Debian 1.3 esiste un meccanismo standard per eseguire i comandi in /etc/cron.daily, /etc/cron.weekly e /etc/cron.monthly.

Cron è fatto partire da script di boot ma sembra cambiare il proprio PATH con un un PATH abbastanza strano:

```
/usr/bin:/binn:/sbin:/bin:/usr/sbin:/usr/bin
```

QUESTO È PROBABILMENTE UN BUG DI CRON. Questo è il PATH di init su cui viene scritto /usr/bin:/bin senza il carattere di terminazione \0! Questo bug non è presente su tutti i sistemi.

In crontab ci possono essere definizioni di PATH. In Debian 1.3 c'è la seguente riga di default all'inizio di /etc/crontab:

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

Proprio per questo, il PATH del programma crond non viene mai usato nei programmi dell'utente. Tutti gli script nelle directory /etc/cron.* prendono questo PATH come predefinito. Questo è il PATH usato anche se un programme è eseguito da un utente senza i privilegi di root.

10.2 at

at è un comando che può essere usato per far girare un programma in un momento specifico.

atd utilizza il PATH di init. Tuttavia, i programmi dell'utente sono sempre fatti girare nell'ambiente dell'utente facendo uso del comando sh. Quindi si applicano le solite regole delle shell. Si veda la sezione su bash.

11 Alcuni esempi

11.1 magicfilter

magicfilter è un tool comune per manipolare file per la stampante. Esso analizza il tipo del file che deve essere stampato e invoca uno script filtro che si occupa della sua formattazione. Questi script sono invocati da lpd, che è fatto partire da /etc/init.d/lpd che a sua volta è fatto partire da init. Quindi il path è quello di init. Che non contiene /usr/bin/X11!

Si potrebbe voler inserire il supporto per la stampa dei file PDF per magicfilter. È possibile fare ciò usando /usr/bin/X11/xpdf. Bisogna ricordarsi di inserire il percorso completo del file perché magicfilter non sarà in grado di trovarlo altrimenti. La maggior parte dei programmi usati da magicfilter non ha bisogno di specificare un percorso completo, perché si trova in /bin o in /usr/bin.

11.2 Stampare da applicazioni X

Si può usare la variabile d'ambiente PRINTER per visualizzare la stampante utilizzazta. Tuttavia, si potrebbe notare che in alcuni casi in applicazioni X tale variabile talvolta è mancante.

Bisogna ricordarsi che se la sessione X è fatta partire da XDM, il window manager non ha avuto modo di valutare gli script di login della shell. Tutte le applicazioni X che sono state fatte partire da xterm hanno la variabile PRINTER. Ma se la stessa applicazione viene fatta partire da menu o da un pulsante del window manager, essa non contiene la variabile PRINTER.

In alcuni casi questo comportamento può essere derivato da un layer ancora inferiore: ad esempio un plugin di Netscape può avere oppure no la definizione della variabile PRINTER.

12 Problematiche di sicurezza

PATH può provocare dei rilevanti problemi di sicurezza. Un modo molto comune per utenti non autorizzati di accedere ad un sistema è quello di sfruttare degli errori nell'impostazione di PATH. È facile perpetrare attacchi di tipo trojan horse se un cracker riesce a fare in modo che root o un altro utente esegua una sua versione dei comandi.

Uno sbaglio comune nel passato (?) era quello di tenere '.' nel PATH di root. Un cracker potrebbe creare un programma "ls" nella sua home directory. Se root esegue:

```
# cd ~cracker
# ls
```

esegue il comando ls del cracker.

Indirettamente ciò si applica a tutti i programmi che vengono eseguiti come root. Se un altro utente può scrivere su un file, quel file non dovrebbe essere mai eseguito da nessun processo demone importante. In alcuni sistemi /usr/local/bin contiene programmi a cui vengono applicati controlli di sicurezza meno severi, viene semplicemente rimosso dal PATH dell'utente root. Tuttavia è risaputo che alcuni demoni eseguono "foo" usando il PATH "/usr/local/bin/:..."; è possibile "imbrogliare" il demone facendogli eseguire "/usr/local/bin/foo" al posto di "/bin/foo". È probabile che chiunque possa scrivere su "/usr/local/bin" possa anche entrare senza autorizzazione nel sistema.

È molto importante considerare in quale ordine le directory sono elencate in PATH. Se /usr/local/bin viene prima di /bin ci troviamo di fronte ad un rischio per la sicurezza del sistema, se viene dopo non è più possibile sostituire il comando /bin/foo con qualche versione localizzata in /usr/local/bin/foo.

Bisogna ricordare che in Linux la valutazione del PATH è eseguita al livello delle chiamate di sistema. Dovunque ci sia un file eseguibile viene fornito un PATH in cui viene ricercato un nome (se esso non contiene il percorso completo) almeno nelle directory /bin e /usr/bin e probabilmente anche in molte altre cartelle.

13 Come individuare i problemi?

Il comando di base per visualizzare l'ambiente è /usr/bin/env.

È possibile usare la directory /proc per cercare il PATH di qualunque programma. Prima bisogna conoscere il numero del processo (si usi ps per quest'ultimo). Per esempio, se il numero del processo di xterm è 1088, si può controllare il suo ambiente con il comando

```
# more /proc/1088/environ
```

Questo non funziona con processi demone come xdm. Per accedere all'ambiente di processi di sistema o altri processi dell'utente sono richiesti i privilegi di root.

Per fare il debug di Netscape, si può creare uno script /tmp/test:

```
$ cat > /tmp/test
#!/bin/sh
/usr/bin/env > /tmp/env
^d
$ chmod +x /tmp/test
```

Poi si può impostare un plugin, ad esempio RealAudio, audio/x-pn-realaudio per fare in modo che chiami il programma "/tmp/test". Quando si cliccherà su dei link RealAudio (ad esempio qualcosa da http://www.realaudio.com/showcase), Netscape chiamerà il programma dummy che scrive l'ambiente in /tmp/env.

14 Alcuni strategie per avere lo stesso PATH per tutti gli utenti

La più importante opzione di configurazione al riguardo si trova nei file di inizializzazione globale delle shell per le login shell: /etc/csh.login per tcsh e /etc/profile per bash.

Eccezioni che non ottengono il PATH corretto da questi file sono i comandi rsh, i comandi ssh, le voci del menu del window manager X che non fanno partire esplicitamente una login shell, i comandi invocati da inittab, i task di cron, i processi che vengono avviati da demoni come lprd, gli script CGI e così via.

Se il PATH è impostato in /etc/csh.cshrc, la variabile PATH viene definita correttamente anche quando rsh o ssh eseguono un comando su una macchina remota con un account che utilizza tcsh/csh. Tuttavia non è possibile fare la stessa cosa se l'account usa bash/sh.

È possibile combinare le impostazioni del PATH in un file, ad esempio nel file /etc/environment-common. Qui possiamo scrivere:

```
${EXPORT}PATH${EQ}/bin:/usr/bin:/sbin:/usr/sbin:/usr/bin/X11:/usr/local/bin:/usr/games:.
```

Questo può essere usato da /etc/csh.login (per tcsh e csh)

```
set EQ=" " set EXPORT="setenv " source /etc/environment-common
```

E da /etc/profile (per bash, non funziona per sh)

```
EQ='=' EXPORT="export " . /etc/environment-common
```

E da /etc/environment (per XDM)

```
EQ="=" EXPORT="export " . /etc/environment-common
```

Questa strategia funziona la maggior parte delle volte ma ssh avrà qualcosa da ridire riguardo alle righe contenute in /etc/environment (e riguardo alle variabili d'ambiente EQ e EXPORT). E comunque i comandi rsh eseguiti con bash non avranno tale PATH.

15 Riconoscimenti

Una ragione per cominciare a scrivere questo documento è stata la forte pressione di Ari Mujunen. Juha Takala mi ha fornito alcuni validi commenti.