

PATH HOWTO

Esa Turtainen

etu@dna.fi

Перевод: [Александр Ермолаев, SWSoft Pte Ltd.](#)

Версия 0.4, 15 Ноября 1997

1. Введение

Этот документ описывает общие проблемы с переменными окружения Unix / Linux, особенно с переменной PATH. Переменная PATH содержит список каталогов, в которых производится поиск команд. Применимо для дистрибутива Debian Linux 1.3.

Примечание! Этот документ находится в стадии разработки. Пожалуйста, посыпайте комментарии и исправления.

2. Copyright

This documentation is free documentation; 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 documentation 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 documentation; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

3. Авторские права

Авторские права на русский перевод этого текста принадлежат © 2000 SWSoft Pte Ltd. Все права зарезервированы.

Этот документ является частью проекта Linux HOWTO.

Авторские права на документы Linux HOWTO принадлежат их авторам, если явно не указано иное. Документы Linux HOWTO, а также их переводы, могут быть воспроизведены и распространены полностью или частично на любом носителе, физическом или электронном, при условии сохранения этой заметки об авторских правах на всех копиях. Коммерческое распространение разрешается и поощряется; но, так или иначе, автор текста и автор перевода желали бы знать о таких дистрибутивах.

Все переводы и производные работы, выполненные по документам Linux HOWTO, должны сопровождаться этой заметкой об авторских правах. Это делается в целях предотвращения случаев наложения дополнительных ограничений на распространение документов HOWTO. Исключения могут составить случаи получения специального разрешения у координатора Linux HOWTO, с которым можно связаться по адресу приведенному ниже.

Мы бы хотели распространить эту информацию по всем возможным каналам. Но при этом сохранить авторские права и быть уведомленными о всех планах распространения HOWTO. Если у вас возникли вопросы, пожалуйста, обратитесь к координатору проекта Linux HOWTO по электронной почте: <linux-howto@metalab.unc.edu> или к координатору русского перевода Linux HOWTO компании SWSoft Pte Ltd. по адресу <linux-howto@asplinux.ru>

4. Общие принципы

Все процессы в Unix содержат "окружение" (environment). Это список переменных, которые содержат имена и значения, они являются строками и могут содержать большинство символов. Все процессы в Unix имеют родительский процесс - процесс созданный этим процессом называется дочерним. Дочерние процессы наследуют окружение от родительского процесса. Они могут делать некоторые изменения в окружении перед принятием окружения уже их дочерними процессами.

Одна важная переменная окружения - PATH (ПУТЬ), список каталогов, разделенных двоеточием (':'). Эти каталоги просматриваются, чтобы найти команды. Если вы пробуете вызвать команду 'foo', все каталоги из переменной PATH (в указанном порядке) будут просмотрены для выполнения файла 'foo' (с установленными правами на выполнение). Если файл найден, он исполняется.

В этом документе я использую термин 'команда', к которому отношу программы, которые, как предполагается, имеют короткое имя, используя механизм путей.

В Linux для запуска процесса операционная система просматривает каталоги, записанные в переменной PATH: вы можете использовать механизм путей там, где пробуете выполнить команду. Если операционная система получает имя файла, который не содержит '/' то просматриваются каталоги из переменной окружения PATH. Даже если в среде не имеется никакой переменной PATH, по крайней мере, каталоги /bin и /usr/bin будут просматриваться.

В sh вы используете команду export, чтобы установить окружение, в csh используйте команду setenv. Например:

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:.
```

C-программы могут использовать библиотечную функцию `setenv()` для изменения окружения. Perl содержит окружение в ассоциативном массиве `%ENV`, вы можете установить PATH так `$ENV{PATH}="/bin"`.

Команда `env` - основной путь выяснения текущих переменных окружения. Она может использоваться также, чтобы изменить их.

Более исчерпывающую информацию по основам механизма окружения можно взять из страниц руководства (man) '`environ`', '`execl`', '`setenv`', или `info 'env'` и из документации по вашей оболочке.

Когда Linux загружается, первый процесс, который запускается является `init`. Это - специальный процесс, потому что не имеет родителя. Однако он первичный для всех процессов. Окружение `init` останется окружением всех процессов, если он не касается окружения явно. Большинство процессов изменяют окружение.

`Init` запускает группу процессов. Файл `/etc/inittab` сообщает то, какие процессы система запускает. Эти процессы работают в окружении, прямо унаследованом от `init` - например программа '`getty`' пишет '`login:`' на консоль. Если вы запускаете соединение PPP, вы должны помнить, что вы работаете в окружении `init`. Инициализация системы - это часто сценарий, который запускается отсюда. В Debian 1.3 скрипт инициализации `/etc/init.d/rc` вызывает, в свою очередь, другие сценарии инициализации.

Система содержит много запускаемых серверов (демоны), которые могут использовать или не использовать окружение, установленное по умолчанию. Большинство серверов запускается из сценария инициализации, и, таким образом, имеют окружение от `init`.

Когда пользователь заходит в систему, на среду действуют назначения, которые записаны в программу при компиляции, в системный скрипт инициализации и в пользовательский скрипт инициализации. Это довольно усложнено, и текущая ситуация не полностью удовлетворительна. Процесс различается, если пользователь заходит из консоли, XDM или из сети.

5. Процесс Init

`Init` - родительский процесс для всех остальных процессов системы. Другие процессы наследуют окружение процесса `init` и пути, прописанные в `init`. Только в редких случаях другие пути не установлены.

'Пути `init`' зафиксированы в исходниках программы и они выглядят так:

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

Заметьте, что пути `init` не содержат `/usr/local/bin`.

Все программы, которые запускаются из `/etc/inittab` работают в окружении `init`, особенно системные скрипты инициализации из `/etc/init.d` (Debian 1.3).

Все, что запускается из системных сценариев инициализации, имеет окружение init, как окружение по умолчанию. Например, syslogd, kerneld, pppd (когда стартует при запуске), gpm, lpd и inetd имеют окружение init, и не изменяют его.

Группа программ стартует из загрузочного скрипта, и PATH явно установлена в этом скрипте. Например: atd, sendmail, apache и squid.

Имеются другие программы, которые стартуют из сценариев начальной загрузки, но они заменяют путь полностью. Один такой пример - cron.

6. Процесс Login

В текстовой консоли имеется программа getty, ожидающая вход в систему пользователя. Она пишет 'login:' и другие сообщения. Работает в окружении init. Когда пользователь входит в систему, getty вызывает программу 'login'. Эта программа устанавливает пользовательское окружение и вызывает оболочку.

Программа Login устанавливает пути, определенные в /usr/include/paths.h. 'Пути login' различны для root и других пользователей.

для обычных пользователей (_PATH_DEFPATH):

```
/usr/local/bin:/usr/bin:/bin:  
для root (_PATH_DEFPATH_ROOT):  
/sbin:/bin:/usr/sbin:/usr/bin
```

Пути обычных пользователей не содержат пути, содержащие любые sbin каталоги. Однако, они содержат текущий каталог, '.', который считается опасным для пользователя root. Даже /usr/local/bin не указан для root.

Пути Login часто переписываются скриптом инициализации оболочки. Однако, возможно использовать другие программы, записанные в /etc/passwd как оболочки пользователя. Например, я использовал следующую строку, чтобы запускался PPP, когда я вхожу в систему, используя специальное имя пользователя. В этом случае, pppd имеет точный путь входа в систему.

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

7. Оболочки

Часто пользовательские процессы - дочерние процессы оболочки записаны в /etc/passwd для этого пользователя. Файлы инициализации оболочек часто изменяют пути.

В login, названию оболочки предшествует '-', для bash например, написано '-bash'. Это сигнал системе, что оболочка запускается при входе в систему. В этом случае, оболочка выполняет инициализационные файлы при входе в оболочку. Иначе происходит более легкая инициализация. Дополнительно оболочка проверяет - являются ли команды исходящими из файла или набираемыми на терминале. Это модифицирует инициализацию оболочки так, что неинтерактивная оболочка инициализируется совсем слегка, bash, в этом случае, не выполняет никакой инициализации.

7.1. bash

Как нормальная оболочка, bash просматривает общесистемный файл `/etc/profile`, где описано системное окружение и пути, которые могут быть установлены для пользователей bash. Однако, это не выполняется, когда система интерпретирует оболочку как не-интерактивную. Наиболее важный случай находится в `rsh`, когда удаленная команда выполняется на соседней машине. `/etc/profile` не запускается, и пути наследуются от `rsh` демона.

bash получает аргументы командной строки `-login` и `-i`, которые могут быть использованы, чтобы установить оболочку, как оболочку для входа или как интерактивную.

Пользователь может переписать значения, установленные в `/etc/profile` путем создания файлов `~/.bash_profile`, `~/.bash_login` или `~/.profile`. Обратите внимание, что только самый первый из них выполняется - отличается от логики инициализации `csh`. `~/.bash_login` не выполняется специально для оболочки входа в систему и если `.bash_profile` существует, он не выполняется вообще.

Если bash используется с именем `sh` вместо имени `bash`, он эмулирует инициализацию `bash`: ищет файлы `/etc/profile` и `~/.profile` только для входных оболочек.

7.2. tcsh

При входе оболочки `tcsh` исполняет следующие файлы в данной последовательности:

- `/etc/csh.cshrc`
- `/etc/csh.login`
- `~/.tcshrc`
- `~/.cshrc` (если `.tcshrc` не найден)
- `~/.history`
- `~/.login`
- `~/.cshdirs`

`tcsh` может быть скомпилирован так, чтобы выполнять `login` скрипт до `cshrc` скрипта. Остерегайтесь!

Не-интерактивные оболочки выполняют только *cshrc скрипты. *login скрипты могут использоваться, чтобы установить путь только однажды во входе в систему.

8. Изменение идентификатора пользователя

8.1. su

Команда su делает переключение на нового пользователя. Если никакое имя пользователя не указано, то используется пользователь root.

Обычно su вызывает подоболочку другого пользователя. С аргументом '-' (более новые синонимы -l или --login) su вызывает оболочку, подобную входной. Однако она не использует программу login, чтобы сделать это, но использует встроенную функцию для 'симуляции' (simulation - термин используемый в исходном тексте) программы login. Итак:

для нормальных пользователей

```
/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:..  
для пользователя root  
/sbin:/bin:/usr/sbin:/usr/bin:/usr/bin/X11:/usr/local/sbin:/usr/local/bin
```

su также делает много других изменений в среде окружения.

8.2. sudo

Имеется группа команд, которые делают использование команд супер-пользователя более безопасным. Они позволяют лучше регистрироваться в системе, устанавливать ограничения пользователям и использовать индивидуальные пароли. Наиболее широко используется команда sudo.

```
$ sudo env
```

выполняет команду env как супер-пользователь (если конфигурация позволяет это сделать).

Команда sudo имеет различный подход к обработке путей. Она изменяет пути поиска файлов так, чтобы текущая директория всегда находилась в конце списка. Однако, она не изменяет переменную окружения PATH. Для 'sudo env' и 'env' переменная PATH имеет одинаковые значения. Sudo добавляет только пару переменных среды, подобные SUDO_USER.

9. Сетевые серверы

Большинство сетевых серверов не должно вызвать подпроцессы любого вида. Из соображений безопасности их путь должен быть минимален.

Важное исключение - все сервисы, которые позволяют подключаться к системе через сеть. Этот раздел описывает окружение в этих случаях. Если команда выполняется на удаленной машине с помощью rsh, она получает другой path, чем тот, если команда выполнена с помощью ssh. Подобно, вход в систему с помощью rlogin, Telnet или ssh различны.

9.1. inetd

Большинство сетевых сервисов не имеют собственного процесса для постоянного ожидания запросов. Эта работа поручается супер-серверу интернет, называемому inetd. Inetd слушает все определенные сетевые порты и запускает соответствующий сервер, когда имеется входящий запрос. Этот режим определен в /etc/inetd.conf.

inetd стартует из скрипта при запуске системы. Он наследует пути процесса init. Он не изменяет их, и все серверы стартующие от inetd имеют пути процесса init. Пример такого сервера imapd, сервер почтового протокола IMAP.

Другие примеры процессы inetd - telnetd, rlogind, talkd, ftp, popd, многие http серверы и т.д.

Часто использование inetd еще усложняется, при использовании отдельной программы tcpd для запуска конкретного сервера. Это программа делает дополнительную проверку безопасности до запуска конкретного приложения. Она не изменяет пути (не проверено).

9.2. rsh

Демон rsh устанавливает пути из определения _PATH_DEFPATH (/usr/include/paths.h), этот же путь использует программа login для нормальных пользователей. Root получит тот же путь, что и нормальный пользователь.

Фактически, rshd выполняет команду, полученную из командной строки:

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

и shell, это не login-shell. Желательно, чтобы все оболочки, упомянутые в /etc/passwd, поддерживали опцию -c.

9.3. rlogin

Rlogin вызывается при входе в систему, чтобы запустить реальную процедуру входа в систему. Если вы входите с помощью rlogin, то получаете те же самые пути, что и при обычном входе в систему. Большинство других способов войти на Linux компьютер не использует login. Обратите внимание на разность с rsh.

Команда login фактически использует

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

-р сохраняет среду окружения, кроме переменных HOME, PATH, SHELL, TERM, MAIL и LOGNAME. -h сообщает удаленному хосту имя для регистрации.

9.4. telnet

Telnet является че- то подобным rlogin. Использует программу login и командную строку.

9.5. ssh

ssh имеет собственную установку путей. Он добавляет каталог, где находится ssh. Часто это означает, что /usr/bin находится в пути дважды:

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

Путь не содержит /usr/X11/bin, и оболочка, вызванная командой ssh - не оболочка входа в систему. Таким образом,

```
ssh remotehost xterm
```

не работает, и что-либо в /etc/profile или в /etc/csh.cshrc может заменить это. Вы должны всегда использовать явный путь /usr/bin/X11/xterm.

ssh ищет переменные окружения в форме VAR=VALUE в файле /etc/environment. К сожалению, это вызывает некоторые проблемы с XFree86.

10. XFree86

10.1. XDM

XDM является наиболее общим способом войти в систему через графический терминал. Это немного напоминает вход в систему, но внутренне полностью отличается.

В каталоге /etc/X11/xdm имеются конфигурационные файлы, которые выполняются на различных фазах входа в систему. Xstartup (и Xstartup_0 специально для screen 0) содержат команды, запускаемые после входа пользователем в систему (команды выполняются под root'ом).

Путь, который установлен для пользователей, находится в /etc/X11/xdm/xdm-config. Имеются строки:

```
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
```

Это будет путь по умолчанию для нормальных пользователей и для root соответственно. Это очень важно, что /usr/bin/X11 является доступным для пользователей X. Если пользователь X входит на другую машину, чтобы запустить X приложения, он должен добраться до /usr/bin/X11.

После запуска Xstartup XDM запускает /etc/X11/Xsession, которая работает с конечным пользователем. Локальная конфигурация, как предполагается, будет сделана в /etc/environment, это исходит от Xsession, если доступно (Xsession выполнен с /bin/sh, и, таким образом /etc/environment должен быть sh файлом). Это конфликтует с ssh, предполагающим, что /etc/environment - файл, содержащий строки вида VAR=VALUE.

10.2. xterm -ls

По умолчанию путь для всех команд, вызываемых из диспетчера X window, наследуется от XDM. Использование чего-то отличного от этого должно быть установлено явно. Чтобы запустить эмуляцию терминала с "нормальными" путями, нужно использовать специальную опцию. В xterm опция -ls (login shell) должна использоваться, чтобы получить пути определенные в скриптах инициализации оболочки.

10.3. Window manager

Window manager наследует окружение XDM. Все программы запущенные window manager'ом наследуют окружение window manager'a.

Окружение оболочки пользователя не затрагивает программы, запускаемые из window manager'a. Например, если программа запускается от 'xterm -ls', она имеет заданную по умолчанию среду при входе в систему, но если она запускается из меню, то имеет только окружение window manager'a.

11. Команды периодического выполнения cron и at

11.1. cron

Cron является командой, которая периодически выполняет команды, как определено в /etc/crontab, и определяемых пользователем crontab'ах. В Debian 1.3 имеется стандартный механизм, чтобы выполнить команды в /etc/cron.daily, /etc/cron.weekly и /etc/cron.monthly.

Cron стартует с начала загрузки, но это, кажется, изменяет PATH к довольно странному значению:

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

ЭТО ВЕРОЯТНО БАГ В CRON. Это путь, где имеется /usr/bin:/bin, написанный без завершающего нуля! Этот баг есть не во всех системах.

В crontab может находиться определение PATH. В Debian 1.3 имеется следующая, заданная по умолчанию, строка в начале /etc/crontab:

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

Из-за этого PATH программы crond никогда не используется в пользовательских программах. Все скрипты каталогов в /etc/cron.* получают этот путь по умолчанию. Этот путь используется, даже если программы выполняются не под root'ом.

11.2. at

at - команда, которая может использоваться для запуска программы в определенное время.

atd запускается, используя путь по умолчанию. Однако, пользовательские программы всегда работают в операционной среде, используя команду sh. Поэтому обычно оболочка изменяет этот путь. Смотрите главу по bash.

12. Некоторые примеры

12.1. magicfilter

magicfilter является общим инструментом, чтобы управлять файлами для принтера. Он анализирует тип файла, который будет напечатан, и вызывает скрипт для печати соответствующих файлов. Эти скрипты вызываются от демона lpd, который стартует из /etc/init.d/lpd, запускающегося от init. Таким образом путь такой, как и у init. Он не содержит /usr/bin/X11!

Вы можете захотеть поставить на печать PDF файл. Это возможно сделать, используя /usr/bin/X11/xpdf. Теперь вы не должны забыть вставить полный путь к каталогу файла, потому что magicfilter не найдет его. Большинство программ, используемых в magicfilter, не нуждаются в полном пути, т.к. находятся в /bin или /usr/bin.

12.2. Печать из X приложений

Вы можете использовать переменную окружения PRINTER, показывающую принтер, который вы используете. Однако, вы можете обратить внимание, что, в некоторых случаях, в X приложениях эта переменная теряется.

Вы должны помнить, что если X сессия запускается из XDM, window manager никогда не обрабатывает ваши сценарии для входа. Все X приложения, которые вы запускаете из xterm, имеют переменную PRINTER. Однако, если то же самое приложение запускается из меню window manager'a, оно не будет содержать вашу переменную PRINTER.

В некоторых случаях это может быть наследовано даже в нижних уровнях: например, программа помощи Netscape может иметь или не иметь определение переменной PRINTER.

13. По поводу безопасности

Пути представляют иногда большую проблему для безопасности системы. Очень просто взломать систему, используя некоторые ошибки в установках путей. Довольно просто запустить трояна, если хакер получает права root или других пользователей, чтобы исполнять свои программы.

Обычная ошибка раньше состояла в том, что '.' содержалась в путях root'a. Злобный хакер делал программу 'ls' и держал ее в своем домашнем каталоге. Если root делал

```
# cd ~hacker  
# ls
```

он исполнял хакерскую программу ls.

Косвенно, это может применяться ко всем программам, запущенным от root. Любой важный демон никогда не должен запускать программы, записанные другим пользователем. В некоторых системах, /usr/local/bin разрешает содержать программы менее строгие в отношении безопасности - это только что удалено из переменной пути пользователя root. Однако, если известно, что некоторый демон выполняет 'foo', используя пути '/usr/local/bin/:...', то возможно обмануть демон, чтобы он выполнил '/usr/local/bin/foo', вместо '/bin/foo'. Вероятно любой, кто может записывать в '/usr/local/bin', способен взломать систему.

Очень важно рассмотреть, в каком порядке каталоги прописаны в путях. Если /usr/local/bin записан перед /bin - вы рискуете защитой.

В Linux нужно помнить, что определение путей сделано на уровне вызовов операционной системы. Вы можете использовать короткое имя файла, который будет искааться по крайней мере в /bin и /usr/bin - вероятно и в других местах.

14. Как выявлять проблемы?

Основная команда для чтения переменных окружения - /usr/bin/env.

Возможно использовать каталог /proc, чтобы выяснить окружение любой программы. Первое, вы должны знать номер процесса - используйте команду ps, чтобы получить его. Например, если номер процесса xterm - 1088, можно просмотреть его окружение с помощью команды

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

Это не работает с демонами, типа xdm. Чтобы обращаться к среде окружения системных процессов или к другим пользовательским процессам, требуются права root.

Для отладки Netscape, вы можете создать скрипт /tmp/test:

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

Затем установите вспомогательное приложение, например RealAudio, audio/x-pn-realaudio, чтобы вызвать программу "/tmp/test". Теперь попробуйте в вашем браузере пойти по ссылке с RealAudio контентом (например, <http://www.realaudio.com/showcase>), Netscape вызовет вашу программу-куклу, которая сохранит окружение в /tmp/env.

15. Некоторые стратегии, позволяющие получить одинаковые пути для всех пользователей

Наиболее важные установки находятся в глобальных файлах инициализации: /etc/csh.login для tcsh и /etc/profile для bash.

Исключения - программы, которые не могут получить правильные пути от этих файлов - это команды ssh, rsh, X window manager (явно не запускает login shell), ,команды вызываемые inittab, задачи запускаемые cron, демоны, подобные magic filters запускаемые lprd, WWW CGI скрипты и т.д.

Если пути установлены в /etc/csh.cshrc, то они правильные, даже когда rsh или ssh выполняют команду на удаленной машине с аккаунтом, использующим tcsh/csh. Однако, не возможно установить пути, если аккаунт использует bash/sh.

Возможно комбинировать установку путей в один файл, например, в файл /etc/environment-common. Теперь пишем:

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

Это можно использовать из /etc/csh.login (для tcsh и csh)

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

И из /etc/profile (для bash, не работает для обычного sh)

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

И из /etc/environment (для XDM)

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

Такая стратегия, главным образом, работает, но ssh будет выдавать сообщения об ошибках в /etc/environment (на определения EQ и EXPORT). И еще, rsh-команды, выполняемые в bash, не будут получать этот путь.

16. Благодарности

Большое расстройство Ari Miijunen было одной из причин, побудивших меня написать этот документ. Juha Takala дал некоторые ценные комментарии.