PATH HOWTO

Table of Contents

Table of Contents

PATH HOWTO

Esa Turtiainen `etu@dna.fi`

v0.4, 15 November 1997

1.Introduction

This document describes common tricks and problems with Unix / Linux environment variables, especially with PATH variable. PATH is a list of directories where commands are looked for. The details apply for Debian Linux 1.3 distribution.

Note! This document is in beta release status. Please send comments and corrections.

2.Copyright

3.General

All the Unix processes contain an "environment". This is a list of variables that contain name and value, both just strings that can contain most characters. All Unix processes have a parent process – the process that created this process as child. Child processes inherit environment from parent process. They can make some modifications to the environment before passing it in turn to their child processes.

One important environment variable is PATH, a list of directories separated by colons (':'). These directories are searched through to find commands. If you try to invoke command 'foo', all the directories in PATH (in that order) are searched for an executable file 'foo' (one with x–bit on). If a file is found, it is executed.

In this howto, I use term 'command' to refer executable program that is meant to be called with short names, using the path mechanism.

In Linux, even the low level operating system calls to start processes (the exec family of calls) searches through directories in the PATH variable: you can use the path mechanism anywhere where you try to execute a command. If exec operating system call gets a file name that does not contain '/', it evaluates the

PATH environment variable. Even if there is no variable PATH in the environment, at least directories /bin and /usr/bin are looked for suitable commands.

In sh you use export command to set environment, in csh you use setenv command. For example:

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−programs can use setenv() library call to change environment. Perl has environment in an associative array %ENV, you can set PATH as $ENV{PATH}="/bin".

env command is the basic way of asking the current environment variables. It can be used to modify it as well.

More information of the basic environment mechanism can be found from manual pages 'environ', 'execl', 'setenv', info file 'env' and documentation of shells.

When Linux boots up, the first normal process that starts is the init process. It is a special process because it does not have parent. However, it is the ancestor of all the other processes. Init environment will remain as environment of all the processes if they do not touch it explicitly. Most processes do touch.

Init starts a group of processes. File /etc/inittab tells what processes the system starts. These processes work in the environment that is directly inherited from init − typically they are processes like 'getty', the program that writes 'login:' to console. If you start PPP connections here, you must remember that you are working in the init environment. The system initialization is often a script that is started here. In Debian 1.3 initialization script /etc/init.d/rc and it calls other initialization scripts in turn.

The system contains many running servers (daemons) that may or may not use the default environment. Most servers are started from the initialization scripts and thus they have the init environment.

When user logs in to the system, the environment is affected by the settings that are compiled into the programs, system wide initialization scripts and user initialization scripts. This is pretty complicated and the current situation is not completely satisfactory. It is totally different if user logs in from text console, XDM or from network.

4.Init

Init is a parent process for all the other processes of the system. Other processes inherit environment of the init process and the path is the init path in the rare case that no other path is set.

The 'init path' is fixed in the source of the init program and it is:

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

Note that init path does not contain /usr/local/bin.

All the programs that are started from /etc/inittab work in init environment, especially system initialization scripts in /etc/init.d (Debian 1.3).

Everything that is started from system initialization scripts has init environment as default environment. For example, syslogd, kerneld, pppd (when started from startup), gpm and most importantly lpd and inetd have init environment and they do not change it.

A group of programs are started from startup scripts but the PATH environment variable is explicitly set in the startup script. Examples are: atd, sendmail, apache and squid.

There are other programs that are started from boot scripts but they change the path completely. One such example is cron.

5.Login

In text console there is a getty program waiting for user login. It writes 'login:' and other messages. It is working in init environment. When getty gets user to log in to the system, it invokes the 'login' program. This program sets the user environment and invokes the shell.

Login program sets path as defined in /usr/include/paths.h. This 'login path' is different for root users and other users.

for common users (_PATH_DEFPATH):

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

for root (_PATH_DEFPATH_ROOT):

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

Common user's path does not contain any sbin directories. However, it contains the current directory, '.', which is considered dangerous for the root user. Not even /usr/local/bin is available for the root user.

Login path is often overwritten by shell initialization. However, it is possible to use other programs in /etc/passwd as user shells. For example, I have used the following line to start PPP when I log in using special user name. In this case, the pppd has exactly login path.

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

6.Shells

Often user processes are children processes of the shell mentioned in /etc/passwd for this user. Initialization files of shells often modify path.

In login, the name of the shell is preceded with '−', for example bash is called as '−bash'. This signals to the shell that it is a 'login' shell. In this case, the shell executes the 'login' initialization files. Otherwise some lighter initialization is performed. Additionally, the shell checks if it is interactive − are the commands coming from file or interactive tty. This modifies the shell initialization so that a non−interactive non−login shell is initialized very lightly − bash do not execute any initialization file in this case!

6.1 bash

As a normal login shell, bash 'sources' system−wide file /etc/profile, where the system environment and path can be set for bash users. However, it is not run when the system interprets the shell as non−interactive. The most important case is in rsh, where remote command is executed in the neighboring machine. The /etc/profile is not run and the path is inherited from rsh daemon.

bash receives command line arguments −login and −i that can be used to set the shell as a login shell or interactive shell respectively.

The user can overwrite values set in /etc/profile by creating a file ~/.bash_profile, ~/.bash_login or ~/.profile. Note that just the first one of these is executed thus differing of the logic of csh initialization. ~/.bash_login is not executed specially for login shells and if .bash_profile exists, it is not executed at all!

If bash is used with name sh instead of the name bash, it emulates original Bourne shell initialization: it sources just files /etc/profile and ~/.profile and just for login shells.

6.2 tcsh

As a login shell tcsh executes the following files in this order:

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

tcsh can be compiled to execute login scripts before cshrc scripts. Beware!

Non−interactive shells execute just the *cshrc scripts. *login scripts can be used to set the path just once in the login.

7.Changing user ID

7.1 su

Command su sets a new user id to use. If no user id is given, root is used.

Normally su invokes a subshell with a different user id. With argument '−' (more recent synonyms −l or −−login) su invokes shell like login shell. However, it does not use login program to do this but uses a yet another built−in path for login 'simulation' (term used in the source code). It is:

for normal users

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

for root user

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

su makes many quite subtle environment changes as well.

7.2 sudo

There is a group of commands that make use of super user commands safer. They allow better logging, user−based restrictions and usage of individual passwords. Most widely used is sudo.

```
$ sudo env
```

executes command env as super user (if it is configured to allow it).

sudo command has again a different approach to path handling. It modifies the search path so that the current directory is always the last one. However, it does not modify PATH environment variable. 'sudo env' and 'env' give the same value for PATH variable. Sudo adds just couple of environment variables like SUDO_USER.

8.Network servers

Most network servers should not invoke subprocesses of any kind. For security reasons, their path should be minimal.

An important exception is all the services that allow logging in to the system from network. This section describes what is the environment in these cases. If the command is executed in the remote machine with rsh it gets different path than if it is executed with ssh. Similarly, logging in with rlogin, Telnet or ssh is different.

8.1 inetd

Most network servers do not have process of their own waiting for requests all the time. This work is delegated to an Internet super server called inetd. Inetd listens for all the defined network ports and starts the appropriate server when there is an incoming request. This behaviour is defined in /etc/inetd.conf.

inetd is started from system startup scripts. It inherits just path of init process. It does not modify it and all the servers started from inetd has init path. An example of such a server is imapd, the server of IMAP post office protocol.

Other examples of inetd processes are telnetd, rlogind, talkd, ftp, popd, many http servers and so on.

Often usage of inetd is still complicated by using a separate tcpd program to start the real server. It is a program that makes additional security checks before starting the real application. It does not affect the path (not verified).

8.2 rsh

rsh daemon sets the path from _PATH_DEFPATH (/usr/include/paths.h) that is the same path that login program uses for normal users. Root will get the same path than the normal user.

Actually, rshd executes the command it gets with the command line:

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

and shell is not a login shell. It is desirable that all the shells mentioned in /etc/passwd support −c option to give on the command line.

8.3 rlogin

Rlogin is invokes login to make the real login procedure. If you login with rlogin, you get the same path than in login. Most other ways to log in to a Linux computer do not use login. Note the difference with rsh.

The login command actually used is

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

−p preserves the environment except the variables HOME, PATH, SHELL, TERM, MAIL and LOGNAME. −h tells the remote host name for logging.

8.4 telnet

Telnet is similar than rlogin. It uses the login program and the command line to invoke it in a similar way.

8.5 ssh

ssh has a path setting of it's own. It has a fixed path where it adds the directory where ssh is. Often this means that /usr/bin is in the path twice:

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

The path does not contain /usr/X11/bin and shell invoked by ssh command is not a login shell. Thus

```
ssh remotehost xterm
```

never works and anything in /etc/profile or /etc/csh.cshrc can change this. You must always use explicit path /usr/bin/X11/xterm.

ssh searches environment variables of form VAR=VALUE from file /etc/environment. Unfortunately this causes some problems with XFree86.

9.XFree86

9.1 XDM

XDM is the most common way to log in to a graphical terminal. It a bit looks like login but it is internally totally different.

In directory /etc/X11/xdm there are configuration files that are executed on different login phases. Xstartup (and Xstartup_0 specially for screen 0) contains commands to be run after the user has logged in (commands are run as user root).

The path that is set for users is in /etc/X11/xdm/xdm−config. There are lines:

```
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/
```

That will be a default path for normal and root users respectively. It is very important that /usr/bin/X11 is available for X users. If X user logs in to another machine to start and X client application, he should get /usr/bin/X11 to his path even he don't seem to come directly from X terminal.

After running Xstartup the XDM runs /etc/X11/Xsession that is run as the final user. Local configuration is meant to be done in /etc/environment that is sourced (included) from Xsession if available (Xsession is run with /bin/sh and thus /etc/environment must be a sh file). This clashes with ssh that supposes that /etc/environment is a file that contains just lines of form VAR=VALUE.

9.2 xterm −ls

By default the path for all the commands invoked from X window manager menus is the path inherited from XDM. To use something different it must be set explicitly. To start a terminal emulator with a path that is "normal" some special option must be used. In xterm the option −ls (login shell) must be used to get a login shell with path specified in shell login initialization files.

9.3 Window manager menus and buttons

Window manager inherits environment of XDM. All the programs started by the window manager inherit the environment of the window manager.

User shell environment does not affect the programs that are started from window manager buttons and menus. For example, if program is started from 'xterm −ls', it has the default environment of login shell but if it is started from menu, it has just environment of the window manager.

10.Delayed commands cron and at

10.1 cron

Cron is a command that executes commands periodically as specified in /etc/crontab and user−defined crontabs. In Debian 1.3 there is a standard mechanism to execute commands in /etc/cron.daily, /etc/cron.weekly and /etc/cron.monthly.

Cron is started from boot scripts but it seems to change it's PATH to a pretty strange one:

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

THIS IS LIKELY A BUG IN CRON. This is the init path where there is /usr/bin:/bin written over the beginning without terminating 0! This bug does not exist in all the systems.

In crontab there can be PATH definition. In Debian 1.3 there is the following default line in the beginning of /etc/crontab:

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

Because of this, the PATH of crond program is never used in user programs. All the scripts in /etc/cron.* directories get this path by default. This path is used even if a program is executed as non−root.

10.2 at

at is a command that can be used to run a one−time program at specific time.

atd is run using init path. However, the user programs are always run in the user environment using sh command. Therefore the usual shell overwrites apply. Look the chapter on bash.

11.Some examples

11.1 magicfilter

magicfilter is a common tool to manipulate files for printer. It analyzes the type of the file to be printed and invokes a filter script to make appropriate pretty−printing. These scripts are invoked from lpd that is started from /etc/init.d/lpd that is started from init. Thus, the path is that of init. That does not contain /usr/bin/X11!

You might want to insert printing of PDF files to magicfilter. It is possible to do this by using /usr/bin/X11/xpdf. Now you must remember to insert full directory path to the file name because magicfilter would not find it otherwise. Most programs used in magicfilter do not need full path, because they are on /bin or /usr/bin.

11.2 Printing from X applications

You may use PRINTER environment variable to show what is the printer that you are using. However, you may notice that in some cases in X applications it is sometimes lost.

You must remember that if the X session is started from XDM, the window manager has never evaluated your shell login scripts. All the X applications that you have started from xterm have your PRINTER variable. However, if the same application is started from menu or window manager button, it does not contain your PRINTER variable.

In some cases this can be inherited to an even lower layer: for example a Netscape helper application can have or have not your PRINTER definition.

12.Security concerns

The path is sometimes a big security problem. It is a very common way to hack into a system using some mistakes in path settings. It is easy to make Trojan horse attacks if hacker gets root or other users to execute his versions of commands.

A common mistake in the past (?) was to keep '.' in the root's path. Malicious hacker makes program 'ls' in his home directory. If root makes

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

he executes ls command of hacker's.

Indirectly, this same applies to all the programs that are executed as root. Any of the important daemon processes should never execute anything that some other user can write into. In some systems, /usr/local/bin is allowed to contain programs with less strict security screening – it is just removed from the path of the root user. However, if it is known that some daemon executes 'foo' using path '/usr/local/bin/:...', it may be possible to cheat daemon to execute '/usr/local/bin/foo' instead of '/bin/foo'. Likely anybody who can write to '/usr/local/bin' is able to break into the system.

It is very important to consider in what order the directories are in the path. If /usr/local/bin is before /bin, it is a security risk – if it is after, it is not possible to overwrite command /bin/foo with some localized modification in /usr/local/bin/foo.

In Linux it should be remembered that the path evaluation is done in the operating system call level. Everywhere where an executable file path is given you can give a short name that is searched at least from

/bin and /usr/bin – likely from many other places as well.

13.How to debug problems?

The basic command to read environment is /usr/bin/env.

It is possible to use /proc directory to find out path of any program. First you must know the process number – use ps command to get that. For example, if xterm is process number 1088, you can find it's environment with command

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

This does not work with daemon processes like xdm. To access environment of system processes or other user processes, root access is required.

To debug Netscape, you can create a script /tmp/test:

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

Then set some helper application, for example RealAudio, audio/x−pn−realaudio to call program "/tmp/test". When you try to browse some RealAudio link (something from http://www.realaudio.com/showcase), Netscape calls the dummy program that stores environment to /tmp/env.

14.Some strategies to get the same path for all the users

The most important settings is possible to set in the global shell initialization files for login shells: /etc/csh.login for tcsh and /etc/profile for bash.

Exceptions that do not get the right path from these files are rsh commands, ssh commands, menu items from X window manager that do not explicitly start login shell, commands invoked from inittab, cron jobs, daemons jobs like magic filters started from lprd, WWW CGI scripts, and so on.

If the path is set in /etc/csh.cshrc, the path is right even when rsh or ssh execute command in remote machine with account using tcsh/csh. However, it is not possible to set path if account uses bash/sh.

It is possible to combine path setting to one file, for example to a file /etc/environment–common. There we write:

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

This can be used from /etc/csh.login (for tcsh and csh)

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

And from /etc/profile (for bash, doesn't work for ordinary sh)

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

And from /etc/environment (for XDM)

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

This strategy works mostly but ssh will complain of the lines in /etc/environment (and defined environment variables EQ and EXPORT). And still, rsh commands executed with bash won't get this path.

15.Acknowledgements

One reason to start writing this document was the big frustration of Ari Mujunen. Juha Takala gave some valuable comments.
