Shells, Part II: Aliases, Variables, Customization*

Boris Veytsman

July 12, 2001

*This document contains lecture notes for informal Unix seminar for ITT AES employees (Reston, VA). No information in this document is either endorsed by or attributable to ITT. This document contains no ITT Privileged/Proprietary Information.

Aliases

The mark of a good party is that you wake up the next morning wanting to change your name and start a new life in different city.
Vance Bourjaily, "Esquire"

A way to quickly create new commands:

```
alias dir 'ls -ls' # C shell
alias dir='ls -ls' # bash & ksh
```

Good practice: Always use quotes

```
boris@reston-0491:~$ alias d='ls -ls'
boris@reston-0491:~$ d
total 108
   4 -rw-r--r--    1 boris    users         457 Jul 10 12:24 7_shellsii.aux
  16 -rw-r--r--    1 boris    users       15970 Jul 10 12:24 7_shellsii.log
   4 -rw-r--r--    1 boris    users          40 Jul 10 12:24 7_shellsii.out
  68 -rw-r--r--    1 boris    users       62863 Jul 10 12:24 7_shellsii.pdf
   4 -rw-r--r--    1 boris    users        1813 Jul 10 12:26 7_shellsii.tex
   4 -rw-r--r--    1 boris    users        1398 Jul  2 12:06 7_shellsii.tex~
   4 drwxr-xr-x    2 boris    users        4096 Jul  2 12:05 RCS
   4 drwxr-xr-x    2 boris    users        4096 Jul 10 12:26 auto
boris@reston-0491:~$ alias
alias d='ls -ls'
```

You can even redefine existing commands:

```
boris@reston-0491:~$ alias rm='rm -i'
boris@reston-0491:~$ rm a
rm: remove 'a'? n
```

Some people think *this* alias is bad!

If you want to use the original command:

- Put \ in front:

```
boris@reston-0491:~$ \rm a
boris@reston-0491:~$
```

- Put the command in quotes:

```
boris@reston-0491:~$ 'rm' a
boris@reston-0491:~$
```

List of all aliases: *alias* or *alias -p*. Remove an alias: *unalias*.

```
boris@reston-0491:~$ alias -p
alias d='ls -ls'
alias rm='rm -i'
boris@reston-0491:~$ unalias rm
boris@reston-0491:~$ alias -p
alias d='ls -ls'
```

Variables

As Will Rogers would have said, "There is no such things as a free variable."

Two kinds of variables:

1. Shell variables (convention: lowercase)

2. Environment variables (convention: uppercase)

/bin/sh:

```
boris@reston-0491:~$ var=5
boris@reston-0491:~$ echo $var
5
boris@reston-0491:~$ export VAR=6
boris@reston-0491:~$ echo $VAR
6
```

/bin/csh:

```
reston-0491:~> set var=5
reston-0491:~> echo $var
5
reston-0491:~> setenv VAR 6
reston-0491:~> echo $VAR
6
```

Deleting variables:

/bin/sh

```
boris@reston-0491:~$ unset var
boris@reston-0491:~$ unset VAR
```

/bin/csh

```
reston-0491:~> unset var      # Shell variable
reston-0491:~> unsetenv VAR   # Environment variable
```

Parents, Children

Children are natural mimics who act like their parents despite every effort to teach them good manners.

The main purpose of a shell—start another process.

Parent: the process that started another process

Child: the process started by another process

```
boris@reston-0491:~$ pstree
init-+-atd
     |-cron
     |-emacs
     |-6*[getty]
     |-gpm
     |-inetd---nmbd
     ...
     |-rxvt---bash---acroread
     |-rxvt---bash---pstree
     |-syslogd
     |-xclock
     |-xdm-+-XF86_SVGA
     |     `-xdm---sh-+-fvwm1
     |                |-ssh-agent
     |                `-xscreensaver
     |-xfs
     `-xterm---ssh
```

You can invoke a subshell by brackets:

```
boris@reston-0491:~$ (cd /; ls);
bin        dev   dosi  dosn    initrd      proc  usr
boot       dosc  dosk  etc     lib         root  var
boot.sav   dose  dosl  floppy  lost+found  sbin  vmlinuz
cdrom      dosh  dosm  home    mnt         tmp   vmlinuz.old
```

Everything inside brackets is done in a *subshell*

I Rule of Inheritance

Blessed are the young, for they shall inherit the national debt. *Herbert Hoover*

Internal commands: *cd*, *alias* etc.—integrated into shell

External commands: *ls*, *rm*, all your scripts: shell stops, gives them control and then returns. This is called *subprocess* or *child*

Parent shell waits for the child to end and then resumes.

All environment changes made by external commands are *lost* after they are done.

Parents do not inherit from children

```
boris@reston-0491:~$ cat setx.sh
#!/bin/sh
x=5
echo $x

boris@reston-0491:~$ unset x
boris@reston-0491:~$ ./setx.sh
5
boris@reston-0491:~$ echo $x
```

The way to use scripts to change environment:

```
boris@reston-0491:~$ unset x
boris@reston-0491:~$ . setx.sh
5
boris@reston-0491:~$ echo $x
5
```

or

```
boris@reston-0491:~$ unset x
boris@reston-0491:~$ source setx.sh
5
boris@reston-0491:~$ echo $x
5
```

II Rule of Inheritance

We have not inherited the earth from our parents, we've borrowed it from our children.

The big difference between shell variables and environment variables:

Children inherit from parents only environment variables

A rule for */bin/bash*: if you do not your variables to be taxed away—export them!

```
boris@reston-0491:~$ cat display.sh
#!/bin/sh
echo x=$x
echo X=$X

boris@reston-0491:~$ x=5
boris@reston-0491:~$ export X=5
boris@reston-0491:~$ echo x=$x X=$X
x=5 X=5
boris@reston-0491:~$ ./display.sh
x=
X=5
```

Exemption: subshell.

- Subshell is invoked by brackets
- I rule of inheritance works for subshell, but II rule does not

```
boris@reston-0491:~$ unset x
boris@reston-0491:~$ (x=5; echo subshell x=$x); echo outer x=$x
subshell x=5
outer x=

boris@reston-0491:~$ unset x
boris@reston-0491:~$ x=5; (echo subshell x=$x); echo outer x=$x
subshell x=5
outer x=5
boris@reston-0491:~$
```

Setting and Querying Environment

If a child looks like his father, that's heredity.
If he looks like a neighbor, that's environment.

Why do we need to set up environment variables:

1. System-wide customization: paths to common utilities, common umasks etc

2. User's customization: create the environment *you* prefer.

Setting environment: *export* (*/bin/sh*) and *setenv* (/progr/bin/csh). Printing environment: *printenv*

```
boris@reston-0491:~$ printenv
PWD=/home/boris
WINDOWID=121634818
HOSTNAME=reston-0491
HOSTDISPLAY=reston-0491:0.0
PS1=\u@\h:\w\$
USER=boris
MACHTYPE=i386-pc-linux-gnu
LANG=ru_RU.KOI8-R
COLORTERM=rxvt
DISPLAY=:0.0
LOGNAME=boris
SHLVL=2
SHELL=/bin/bash
HOSTTYPE=i386
OSTYPE=linux-gnu
HOME=/home/boris
TERM=rxvt
PATH=/usr/local/bin:/usr/bin:/bin:/usr/bin/X11
_=/usr/bin/printenv
```

```
kenny ~[1]>printenv
HOME=/home/bveytsma
PATH=/:/usr:/usr/bin:/usr/local/bin:/usr2/apps/SUNWspro/\
   bin:/usr/ccs/bin:/usr/sbin:/usr/openwin/bin:/usr/openwin/\
   local/bin:/usr/lib/lp/postscript:/usr/ucb:/nstb/alsim/bin:\
   /vni/naste:/vni/analysis:/etc:/usr2/apps/cmvc/bin:.
LOGNAME=bveytsma
HZ=100
TERM=rxvt
TZ=US/Eastern
SHELL=/bin/csh
MAIL=/var/mail/bveytsma
DISPLAY=reston-0491.reston.aes.itt.com:0.0
USER=boris
PWD=/home/bveytsma
ARCH=solaris
MANPATH=/usr/man:/usr/openwin/man:/usr2/apps/SUNWspro/man
OW_WINDOW_MANAGER=/usr/openwin/local/bin/olwm
VNI_DIR=/usr2/vni
WAVE_DIR=/usr2/vni/wave
OPENWINHOME=/usr/openwin
```

```
LD_LIBRARY_PATH=/usr/openwin/lib
LICENSE_DIR=/usr2/vni/license
LM_LICENSE_FILE=/usr2/vni/license/license.dat
WAVE_CODEBOOK=/usr2/vni/wave/demo/codebook
WAVE_DATA=/usr2/vni/wave/data
WAVE_DEMO=/usr2/vni/wave/demo
WAVE_GALL2=/usr2/vni/wave/demo/gallery2
WAVE_GALL3=/usr2/vni/wave/demo/gallery3
WAVE_GALL=/usr2/vni/wave/demo/gallery3
WAVE_LIB=/usr2/vni/wave/lib
WAVE_HELPDIR=/usr2/vni/wave/help
WAVE_HELP_PATH=/usr2/vni/wave/help
WAVE_PATH=/usr2/vni/wave/lib:/usr2/vni/wave/lib/std:\
    /usr2/vni/wave/lib/std/motif:/usr2/vni/wave/lib/vdatools:\
    /usr2/vni/wave/lib/user:/usr2/vni/wave/lib/std/guitools:\
    /usr2/vni/wave/lib/user/guitools
XFILESEARCHPATH=/usr/lib/X11/%L/%T/%N%C:/usr/lib/X11/%l/%T/%N%C:\
    /usr/lib/X11/%T/%N%C:/usr/lib/X11/%L/%T/%N:\
    /usr/lib/X11/%l/%T/%N:\/usr/lib/X11/%T/%N:\
    /usr2/vni/wave/xres/vdatools/%N:\
    /usr2/vni/wave/xres/american/vdatools/%N
HHHOME=/usr2/vni/hyperhelp
```

```
HHPATH=/usr2/vni/hyperhelp/bin/bin_solaris
HOHPATH=/usr2/vni/hyperhelp
WAVE_HLPFILE=/usr2/vni/hyperhelp/wave.hlp
XLIBI18N_PATH=/usr2/vni/hyperhelp/lib
IMSLERRPATH=/usr2/vni/math-1_0/bin/
IMSLSERRPATH=/usr2/vni/stat-1_0/bin/
WAVEMAPLE_DIR=/usr2/vni/maple
WAVE_VERSION=PV-WAVE v6.05  Build #728  (Mon Dec 16 10:17:40 MST 1996)
VNI_PHONE=303-530-5200
WAVE_PRODUCT=advantage
NASTE_DIR=/usr2/vni/naste-7.0
EDITOR=/usr/ucb/vi
MAKEFILE=-s
MANSECTS=\1:1m:1c:1f:1s:1b:2:\3:3c:3i:3n:3m:3k:3g:\
    3e:3x11:3xt:3w:3b:9:4:5:7:8
SHLVL=1
HOST=kenny
HOSTTYPE=sun4
```

Shells and Initialization Files

How can you govern a nation which has 246 kinds of cheese? *Charles de Gaulle*

Login shell: The first shells you have when you login

Interactive Shell: The shell you use on a terminal (input and output are terminal)

Sourcing of init files by */bin/bash*

Login shell: `/etc/profile` and the first of `~/.bash_profile`, `~/.bash_login`, `~/.profile`

Interactive non-login shell: `~/.bashrc`

Sourcing of init files by */bin/csh*

Login shell: `/etc/csh.cshrc`, then `~/.cshrc` (*tcsh* uses `~/.tcshrc` if present), then `~/.login`

Non-login shells: `/etc/csh.cshrc` and `~/.cshrc`

Note the difference in treating *rc* files!

A problem with windowing system: often they do not start login shells! Solutions:

1. Use *xterm -ls* instead of *xterm*
2. Put your initialization in *rc*-file
3. Put your initalization into `~/.xsession` file