The Environment

environment practice for CSC420 Operating Systems Spring 2022 Lyon College

README

- This file accompanies lectures on the shell and `bash(1)`. To gain practice, you should type along in your own Org-mode file. You have to have Emacs and my `.emacs` file installed on your PC or the Pi you're working with.
- To make this easier, use the auto expansion (`<s`). This will only work if you have my `.emacs` file (from GDrive) installed.

- Add the following two lines at the top of your file, and activate each line with `C-c C-c` (this is confirmed in the echo area as `Local setup has been refreshed`)):

```
#+PROPERTY: header-args:bash :results output
```

- Remember that `C-M-\` inside a code block indents syntactically (on Windows, this may only work if you have a marked region - set the mark with `C-SPC`).
- This section is based on chapter 11 of Shotts, The Linux Command Line (2e), NoStarch Press (2019).

What is it?

- The environment is the information retained by the shell about our shell session
- Programs use the data stored in the environment for configuration - e.g. when installing files, setting permissions
- Knowing the environment helps us customize our shell experience (including shell scripting)

- Types of data stored in the environment:

DATA	DESCRIPTION	EXAMPLE
Shell variables	Placed by bash	`$PWD`
Environment variables	Placed by OS	`$HOME`
Aliases	User-defined	`alias ll='ls -lh'`
Shell functions	User-defined	`hello.sh`

```
!# /usr/bin/bash
echo Hello world
```

```
Hello world
```

Examining the environment

- [ ]

  You can use the builtin bash programs `set`, or the program `printenv` to view the environment.

  Pipe the output of `printenv` into `cat` to view it.

```
printenv | cat
```

```
SHELL=/bin/bash
XDG_CONFIG_DIRS=/etc/xdg
XDG_MENU_PREFIX=lxde-pi-
COGL_DRIVER=gles2
LANGUAGE=en_US.UTF-8
_LXSESSION_PID=1156
SSH_AUTH_SOCK=/tmp/ssh-SrWIgorWXSAz/agent.1156
XDG_CONFIG_HOME=/home/pi/.config
DESKTOP_SESSION=LXDE-pi
SSH_AGENT_PID=1206
NO_AT_BRIDGE=1
XDG_SEAT=seat0
PWD=/home/pi/GitHub/admin/spring22/os420
LOGNAME=pi
QT_QPA_PLATFORMTHEME=qt5ct
XDG_SESSION_TYPE=tty
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
_=/usr/bin/printenv
XAUTHORITY=/home/pi/.Xauthority
DESKTOP_STARTUP_ID=lxpanel-1253-raspberrypi-/usr/bin/emacs-0_TIME6217404
WINDOWPATH=1
MOTD_SHOWN=pam
HOME=/home/pi
LANG=en_US.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=4
XDG_CURRENT_DESKTOP=LXDE
V3D_IGNORE_SCANOUT_USAGES=1
XDG_SESSION_CLASS=user
TERM=dumb
USER=pi
DISPLAY=:0
SHLVL=1
XDG_VTNR=1
XDG_SESSION_ID=1
XDG_RUNTIME_DIR=/run/user/1000
CLUTTER_DRIVER=gles2
LC_ALL=en_US.UTF-8
XDG_DATA_DIRS=/usr/share/fkms:/usr/local/share:/usr/share/raspi-ui-overrides:/usr/
HUSHLOGIN=FALSE
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games
SAL_USE_VCLPLUGIN=gtk3
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
MAIL=/var/mail/pi
TEXTDOMAIN=Linux-PAM
```

- [ ]

  How many different environment variables are there? (Make sure you removed potential duplicates.)

```
printenv | uniq | wc -l
printenv | wc -l
```

44

44

- [ ]

You can also print individual variables with `printenv`, e.g. `$USER`.

```
printenv USER
```

```
pi
```

- [ ]

The bash command `set` does the same thing, or does it? Write a pipe that tees the output of `cat` to a file `set.txt` to view later, and counts its lines.

```
set | cat | tee set.txt | wc -l
```

```
68
```

- [ ]

You already know another way of printing variable values - with `echo`. Print the value of `HOME` using this command.

```
echo $HOME
```

```
/home/pi
```

- [ ]

Neither `set` nor `printenv` display aliases. To see them, open an Emacs shell with `M-x shell` and enter `alias` without arguments at the prompt. This is what I see on my Pi:

```
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias ls='ls --color=auto'
```

- [ ]

Go back to the shell buffer and create an alias `dh` for the human readable file system disk space usage information. Check the man page for `df` if necessary.

```
alias dh='df --human-readable'
```

- `[ ]`

  Now check the alias listing again. This is what I see on my Pi:

```
    pi@raspberrypi:~/GitHub/os420$ alias
    alias dh='df --human-readable'
pp    alias egrep='egrep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias grep='grep --color=auto'
    alias ls='ls --color=auto'
```

Interesting variables

Your specific environment may differ from the entries of this list, but you're likely to see these variables.

Check their values out with `printenv` or `echo` (see below). Don't worry if some values are missing - they vary with the distribution.

VARIABLE	CONTENT
DISPLAY	Graphical display name (`:0`)
EDITOR	Program used for text editing
SHELL	Name of your shell program
HOME	Pathname of your home directory
LANG	Character set of your language
OLDPWD	Previous working directory
PAGER	Program for paging output (`less`)
PS1	Shell prompt string 1
PWD	Current working directory
TERM	Name of terminal type
TZ	Your time zone (UTC)
USER	Your user name

```
echo "DISPLAY": $DISPLAY
echo "EDITOR": $EDITOR
echo "SHELL": $SHELL
echo "HOME": $HOME
echo "LANG": $LANG
echo "OLDPWD:" $OLDPWD
echo "PAGER:" $PAGER
echo "PATH:" $PATH
echo "PS1:" $PS1
echo "PWD:" $PWD
echo "TERM:" $TERM
```

```
echo "TZ:" $TZ
echo "USER:" $USER
```

```
DISPLAY: :0
EDITOR:
SHELL: /bin/bash
HOME: /home/pi
LANG: en_US.UTF-8
OLDPWD:
PAGER:
PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/games:/us
PS1:
PWD: /home/pi/GitHub/admin/spring22/os420
TERM: dumb
TZ:
USER: pi
```

- [ ]

  Inside Org-mode, the value of `TERM` is most likely `dumb`. Compare this with its value 1) on the Emacs `*shell*`, and 2) in the regular terminal.

  This is what I see on my Pi in the terminal:

  ```
  xterm-256color
  ```

- [ ]

  The shell prompt string `PS1` also won't be displayed inside Org-mode. You can display it on the Emacs shell (or in the terminal):

  ```
  ${debian_chroot:+($debian_chroot)}\u@\h:\w\$
  ```

Starting the environment

- When you log on, `bash` starts and reads its startup files
- The startup files are configuration scripts that defined the environment for all users
- Next, `bash` reads startup files in your `HOME` directory to define your personal user environment
- The exact sequence depends on the type of shell session (login sessions when you're prompted, or non-login session, e.g. when you open a terminal in the GUI).

- Here is a list of some important startup files that you can find on your system. In Emacs, you can just go to the file directly.

FILE	CONTENTS
/etc/profile	Global script for all users
~/.bash.profile	User's personal startup file
~/.bash_login	If ~/.bash_profile not found

FILE	CONTENTS
~/.profile	If the previous two are not found
/etc/bash.bashrc	Global GUI config file
~/.bashrc	Personal GUI config file

- `[ ]`

  How many configuration files do you have in your home directory? Use `grep` with the `-l` option and `wc` in a pipe to get the answer.

```
cd /home/pi
ls ./ |grep -l .* | wc -l
```

```
16
```

- In addition to reading the startup files listed, non-login shells inherit the environment from their parent process (login shell)
- `[ ]` Take a look at your `.bashrc` file in the `HOME` directory. Can you identify any of the settings?

What's in a startup file

- A typical `.bash_profile` looks like this:

```
#.bash_profile

# Get the aliases and functions
if[-f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
export PATH
```

- Lines beginning with `#` are comments and are not read
- The if statement is a compound command, translated:

  > "If the file ~/.bashrc exists, then read the ~/.bashrc file."

- At the end, the `PATH` variable is extended by a directory so that personal files in that directory can be found. What type of files are likely to be stored in `$HOME/bin`?
- `[ ]`

  Try this parameter expansion yourself - not: there must not be any empty spaces in the parameter definition!

  1. Define `foo` to be the string `"This is some "`
  2. Display `foo`
  3. Expand `foo` by the string `"text."`

4. Display the expanded `foo`.

```
foo="This is some "
echo $foo
foo=$foo"text."
echo $foo
```

```
This is some
This is some text.
```

- Lastly, the `export PATH` command tells the shell to make the contents of `PATH` available to all child processes of this shell.
- Child processes of a parent process are all processes spawned in it. You can see them with the command `ps -a`

```
ps -a
```

```
 PID TTY          TIME CMD
 926 tty1     00:00:00 bash
1127 tty1     00:00:00 startx
1149 tty1     00:00:00 xinit
1150 tty1     00:02:21 Xorg
1156 tty1     00:00:01 lxsession
1248 tty1     00:02:06 mutter
1251 tty1     00:00:00 lxpolkit
1253 tty1     00:00:52 lxpanel
1254 tty1     00:00:02 pcmanfm
1264 tty1     00:00:00 applet.py
1594 tty1     00:03:36 emacs
```

Author: Marcus Birkenkrahe
Created: 2022-03-29 Tue 08:20
Validate