Shell variables

Tutor: Lưu Thanh Trà

Email: lttra@hoasen.edu.vn

Outline

- Shell variable
- Embedding documentation
- Surrounding text
- Exporting variables
- Default values
- Arrays

Shell variables

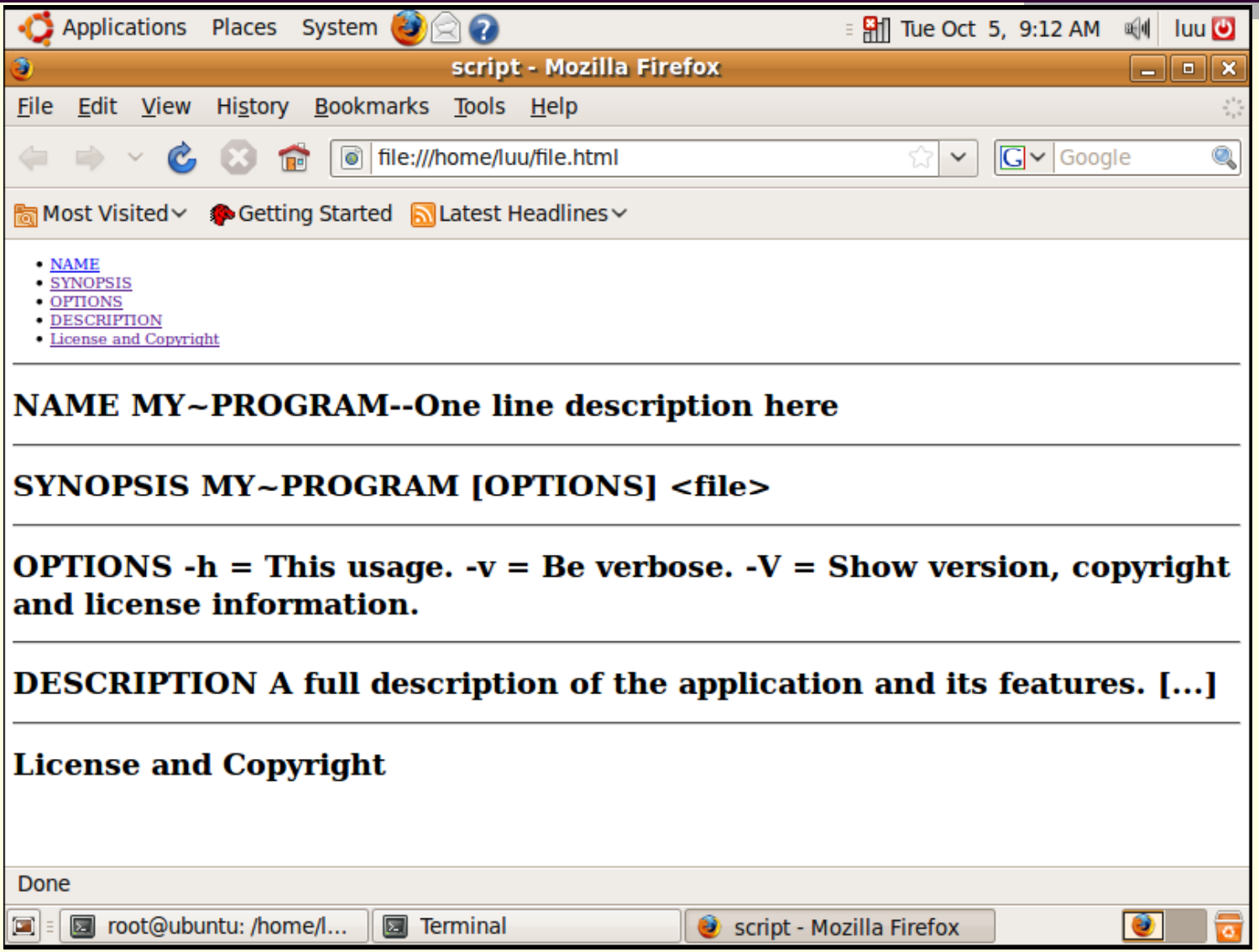
- Example
  - MYVAR=something
  - echo $MYVAR
- Variable name:
  - Do not use $ to name variables
  - Do not use space (“ “)

Embedding documentation in scripts

- #!/usr/bin/env bash
- # cookbook filename: embedded_documentation
- echo 'Shell script code goes here'
- # Use a : NOOP and here document to embed documentation,
- : <<'END_OF_DOCS'
- =head1 NAME
- MY~PROGRAM--One line description here
- =head1 SYNOPSIS
- MY~PROGRAM [OPTIONS] <file>
- =head1 OPTIONS
- -h = This usage.
- -v = Be verbose.
- -V = Show version, copyright and license information.
- =head1 DESCRIPTION

- =head1 LICENSE AND COPYRIGHT
- =cut
- END_OF_DOCS

pod2html, pod2man



Surrounding text

- Use ${ }
- Example
  - for FN in 1 2 3 4 5
  - do
  - somescript /tmp/rep$FNport.txt
  - done
- => somescript /tmp/rep**${FN}**bay.txt

Export variables

- Make a variable is available in other shells
- Example:
  - export FNAME
  - export SIZE
  - export MAX
  - ...
  - MAX=2048
  - SIZE=64
  - FNAME=/tmp/scratch
  - and at other times you'll see:
  - export FNAME=/tmp/scratch
  - export SIZE=64
  - export MAX=2048
  - ...
  - FNAME=/tmp/scratch2

- set: list the shell variables, functions
- env: list only the exported variables
- Example:
  - set | grep MYVAR

Paramters in a shell

- Example
  - $ cat tricky.sh
  - echo $1 $10 ${10}
  - $ ./tricky.sh I II III IV V VI VII VIII IX X XI
  - I I0 X
  - $

Parameters with blanks

- Unix accepts the file name with blanks, however, we have to use quote
  - $ cat quoted.sh
  - # note the quotes
  - ls -l "${1}"
  - $
  - $ ./quoted.sh "Oh the Waste"
  - -rw-r--r-- 1 smith users 28470 2007-01-11 19:22 Oh the Waste

$* and $@

- Example:
  - ls
  - vocals.mp3
  - cool music.mp3
  - tophit.mp3
  - for FN in $*
  - for FN in “$*”
  - for FN in $@
  - for FN in “$@”

Counting arguments with $#

- #!/usr/bin/env bash
- # cookbook filename: check_arg_count
- # Check for the correct # of arguments:
- # Use this syntax or use: if [$# -lt 3]
- if (($# < 3))
- then
- printf "%b" "Error. Not enough arguments.\n" >&2
- printf "%b" "usage: myscript file1 op file2\n" >&2
- exit 1
- elif (($# > 3))
- then
- printf "%b" "Error. Too many arguments.\n" >&2
- printf "%b" "usage: myscript file1 op file2\n" >&2
- exit 2
- else
- printf "%b" "Argument count correct. Proceeding...\n"
- fi

Arguments

```
#!/usr/bin/env bash
# parse the optional argument
VERBOSE=0;
if [[ $1 = -v ]]
then
VERBOSE=1;
shift;
fi
# the real work is here
for FN in "$@"
do
if (( VERBOSE == 0 ))
then
echo changing $FN
fi
chmod 0750 "$FN"
done
```

Default value with ${:-}

- FILEDIR=${1:-"/tmp"}
- Set and Unset
  - $ echo ${HOME:=/tmp}
  - /home/uid002
  - $ unset HOME # generally not wise to do
  - $ echo ${HOME:=/tmp}
  - /tmp
  - $ echo $HOME

Null default value with ${=}

- $ echo ${HOME=/tmp} # no substitution needed
- /home/uid002
- $ HOME=""
- $ echo ${HOME=/tmp} # will NOT substitute
- $ unset HOME
- $ HOME=${HOME:?”Error”}
- $ echo ${HOME=/tmp} # will substitute
- /tmp
- $ echo $HOME
- /tmp

Array

- names=(Jennifer Tonya Anna Sadie)
  - => array with **4** elements
- names=("John Smith" "Jane Doe")
  - => array with **2** elements
- colors[0] = red
  colors[3] = green
  colors[4] = blue
  - => echo ${colors[0]}
- filearray=(`cat filename | tr '\n' ' '`)
  - =>echo ${filearray[0]}