Unit 8
More on Shell Variables

Unit Objectives

After completing this unit, you should be able to:

- Use variable replacements
- Evaluate variable substrings
- Evaluate variable lengths
- Understand further **`typeset`** options
- Use compound variables
- Use indirect variables
- Use tilde expansions

Variable Replacements

Value of variables can be replaced with alternate values

${variable:-WORD}	value is **WORD** if **variable** is unset (use as a temporary value)
${variable:=WORD}	value is **WORD** if **variable** is unset and assigns word to **variable** if it is unset (use as a permanent value)
${variable:+WORD}	value is null if **variable** is unset, else value is **WORD** (use as alternate value)
${variable:?WORD}	if **variable** is unset, **WORD** is displayed on standard error and the shell script or function terminates with a non-zero exit code (exit 1)

Variable Replacement Examples

Some simple examples...

- Print date and time using command substitution, or use what was set earlier (do not allow null date):

```
print ${date:-$(date)}
```

- To assign the value of TERM_DEF to TERM if it is unset or null:

```
TERM_DEF=ibm3162
...
print "TERM set as ${TERM:=$TERM_DEF}"
```

- Using the alternate value "1" if variable has a value:

```
var_flag=${var:+1}
```

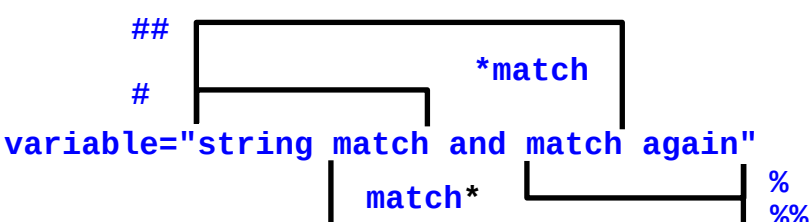
- To exit the script if var1 is unset or null

```
${flag:?"flag is unset"}
```

Shell Substrings

In the shell the **`${ }`** syntax also works with patterns:

`${variable#pattern}`	removes smallest matching left pattern from variable
`${variable##pattern}`	removes the largest matching left pattern
`${variable%pattern}`	removes the smallest right matching pattern
`${variable%%pattern}`	removes the largest matching right pattern



Shell Substring Examples

A bit of chopping...

```
$ variable="Now is the time"
$ print ${variable#N*i}          shortest left
s the time
$ print ${variable##N*i}         longest left
me
$ print ${variable%time}         shortest right
Now is the
$ print ${variable%%t*e}         longest right
Now is
$ _
```

Here's a function to strip out the file name from its path and print it...

```
function base
{
      print  ${1##*/}            # match what?
}
```

Shell Substring Quiz

Now it's your turn...

1. How can I strip the ".c" extension from a C program file name held in variable "name", and print it?

2. Write a function "path" to print the pathname part of a file name.
   -- /usr/local/bin/program

Variable Lengths

A special variant of the **${}** syntax can be used to find the length of a variable:

- To find the number of characters in a variable...

```
${#variable}
```

- The number of positional parameters is...

```
${#*}        or      ${#@}
```

- For the number of elements set in an array (not the highest element subscript)...

```
${#array[*]}      or      ${#array[@]}
```

typeset Options Review

typeset command is used to:

- Set attributes for variables or functions
- Create local variables in functions

typeset ±LN variable=value...

where **L** is...

i	**integer**, **N** is a fixed base
r	to set **readonly**
x	to **export** the variable

typeset ±fL function...

where **L** is...

x	to **export** the function
u	for an **autoload** function
t	to set **xtrace** in the function

- To set attributes, display names and values

+ To unset attributes or display just names

Further typeset Options

Options below allow variables to be formatted upon expansion by the Korn shell:

```
typeset ±LN variable=value...
```

where **L** is...

u	convert ***value*** to uppercase when expanded
l	convert ***value*** to lowercase
L	left-justify, pad with trailing blanks to width **N** – if value is too big, truncate from the right
R	right-justify, adding leading blanks to width **N**– if wider than **N**, truncate from the left
LZ	left-justify to width **N** and strip leading zeros
RZ	right-justify to width **N**, adding lead zeros if the first character is a digit

*The bash shell does not support these options

typeset Examples

Here are the different types in action...

```
$ typeset -u var=upper
$ print $var
UPPER
$ typeset -l var=LOWER                  # lower case "ell"
$ print $var
lower
$ typeset -L6 text=SIDE
$ print "${text}="
SIDE  =
$ typeset -R6 text
$ print "=$text"
=  SIDE
$ typeset -LZ4 num=000.1234567
$ print ${num}
.123
$ typeset -RZ5 num=123
$ print $num
00123
```

Compound Variables in ksh93

ksh93 has an additional feature called *compound variables*, for example:

```
$ time="10:47:24 EST"
$ time.hour=10; time.minute=47
$ time.seconds=24; time.zone=EST
$ print $time
10:47:24 EST
$ print ${time.hour}
10
```

- or -

```
$ time=(hour=10 minute=47 seconds=24 zone=EST)
$ print $time
(hour=10 minute=47 seconds=24 zone=EST)
$ print ${time.zone}
EST
```

Variable Pattern Substitution in bash and ksh93

The bash and ksh93 shells allow for ***on the fly*** variable pattern substitution.

Syntax:

`${variable/pattern/newpattern}`

If **`variable`** contains **`pattern`** the ***first*** match the of **`pattern`** is replaced with **`newpattern`**

`${variable//pattern/newpattern}`

Same as above syntax, except ***every*** match of **`pattern`** is replaced

Also:

`${variable:offset:length}`

Show the substring beginning at **`offset`** for **`length`** number of characters

Tilde Expansions

Following **alias** expansion the Korn shell checks for a leading unquoted **~** character to see if it is:

~	tilde by itself is replaced by **$HOME**
~user_name	is expanded into the **$HOME** value for the ***user_name*** given
~other_text	will be left alone

Examples...

```
cd ~              =     cd $HOME
lastdir=~-        =     lastdir=$OLDPWD
johns=~john       =     johns=/home/john
```

Checkpoint

1. What happens when the variable **TMOUT** is set and you enter the following? **TMOUT=${TMOUT:-60}**

2. What would your prompt say if you were in your bin directory and you entered this: **PS1='${PWD#$HOME/} $'**.

3. How could you find out the number of characters in the variable **HOME**?

Unit Summary

- Variable replacements
  - For unassigned/null strings
- Variable substrings
  - Simple pattern matches
- Variable lengths
  - The # operator
- Further typeset options
  - Justification and padding
- Tilde expansions
  - Shortcuts
- Compound variables
  - ksh93
- Indirect variables
  - ksh93