Unix Shell Programming

Chapter 5: Reading, Formatting and Printing

Roshan Chitrakar, PhD
roshanchi@gmail.com

read

- syntax: ***read variables***
- reads a line from standard input and assigns the first word to the first variable, the second word to the second variable and so on.
- the excess words get assigned to the last variable.

  read x y

  read text

- read always returns an exit status of zero unless an end of file (or Ctrl + d) condition is detected on the input.

Special ***echo*** Escape Characters

- ***echo*** command always automatically displays a terminating newline character
- This can be suppressed if the last two characters given to echo are the special escape characters **\c** e.g.

  echo "$to already exists; overwrite (yes/no)? \c"

- Some other escape characters: -
- \b Backspace
- \c The line without a terminating newline
- \f Formfeed
- \n Newline
- \r Carriage return
- \t Tab character
- \\ Backslash character
- \0nnn The character whose ASCII value is nnn, where nnn is a one- to three-digit octal number

printf

- prints formatted output
- syntax : ***printf "format" arg1 arg2 …***

  $ ***printf "This is a number: %d\n" 10***

  This is a number: 10
- printf doesn't add a newline character to its output like echo; however, printf understands the same escape characters that echo does

- **Some formats: -**
- d Integers
- u Unsigned integers
- o Octal integers
- x Hexadecimal integers, using a-f
- X Hexadecimal integers, using A-F
- c Single characters
- s Literal strings
- b Strings containing backslash escape characters
- % Percent signs

printf examples

- $ ***printf "The octal value for %d is %o\n" 20 20***
  - The octal value for 20 is 24
- $ ***printf "The hexadecimal value for %d is %x\n" 30 30***
  - The hexadecimal value for 30 is 1e
- $ ***printf "The unsigned value for %d is %u\n" –1000 –1000***
  - The unsigned value for -1000 is 4294966296
- $ ***printf "This string contains a backslash escape: %s\n" "test\nstring"***
  - This string contains a backslash escape: test\nstring
- $ ***printf "This string contains an interpreted escape: %b\n" "test\nstring"***
  - This string contains an interpreted escape: test string
- $ ***printf "A string: %s and a character: %c\n" hello A***
  - A string: hello and a character: A
- $ ***printf "Just the first character: %c\n" abc***
  - a

printf examples (contd.)

- The general format of a conversion specification is

  %[flags][width][.precision]type

- $ printf "%+d\n%+d\n%+d\n" 10 –10 20
- +10
- –10
- +20
- $ printf "% d\n% d\n% d\n" 10 –10 20
- 10
- -10
- 20
- $ printf "%#o %#x\n" 100 200
- 0144 0xc8

- - Left justify value.
- + Precede integer with + or -.
- (space) Precede positive integer with space character.
- # Precede octal integer with 0, hexadecimal integer with 0x or 0X.
- width Minimum width of field; * means use next argument as width.
- precision Minimum number of digits to display for integers; maximum number of characters to display
- for strings; * means use next argument as precision.

printf examples (contd.)

- $ ***printf "%20s%20s\n" string1 string2***
- string1 string2
- ***$ printf "%-20s%-20s\n" string1 string2***
- string1 string2
- $ ***printf "%5d%5d%5d\n" 1 10 100***
- 1 10 100
- $ ***printf "%5d%5d%5d\n" -1 -10 -100***
- -1 -10 -100
- $ ***printf "%-5d%-5d%-5d\n" 1 10 100***
- 1 10 100

more printf examples

- The .precision modifier is a positive number that specifies a minimum number of digits to be displayed for %d, %u, %o, %x, and %X. This results in zero padding on the left of the value:
- ***$ printf "%.5d %.4X\n" 10 27***
  - 00010 001B
- ***$ printf "%.5s\n" abcdefg***
  - abcde
- A width can be combined with .precision to specify both a field width and zero padding (for numbers) or truncation (for strings):
- ***$ printf ":%#10.5x:%5.4x:%5.4d\n" 1 10 100***
- : 0x00001: 000a: 0100
- ***$ printf ":%9.5s:\n" abcdefg***
- : abcde:
- ***$ printf ":%-9.5s:\n" abcdefg***
- :abcde :

more printf examples (contd.)

- if an * is used in place of a number for width or precision, the argument preceding the value to be printed must be a number and will be used as the width or precision, respectively.
- If an * is used in place of both, two integer arguments must precede the value being printed and are used for the width and precision:
- ***$ printf "%*s%*.*s\n" 12 "test one" 10 2 "test two"***
- test one te
- ***$ printf "%12s%10.2s\n" "test one" "test two"***
- test one te