Programming in shell 1
Data compression and archiving.
Numeric calculations.

Jan Trdlička

Czech Technical University in Prague, Faculty of Information Technology
Department of Computer Systems

Content

Archiving and compression

- Archive
  - File containing packed files and directories.

- File compression
  - Process of encoding information using fewer bits (or other information-bearing units) than an original representation.
  - Lossless data compression.

- Usage
  - Data transfer.
  - Backup (complete, incremental).

- Backup problems
  - Absolute/relative path.
  - File attributes (owner, modification time, . . .).
  - Hard link.
  - Soft link.
  - Deleted items.

Archiving

- `tar` (Tape ARchive)

    - **C**reate archive(default suffix is .tar)
      ```
      cd directory ; tar -cvf archive.tar *
      ```

      ```
      find . > list.txt ; tar -cvf archive.tar -T list.txt
      ```

    - **T**est archive (list content of archive)
      ```
      tar -tvf archive.tar
      ```

    - **E**xtract archive
      ```
      tar -xvf ../archive.tar
      ```

      ```
      tar -xvpf ../archive.tar          // preserve permissions
      ```

Compression – compress

- `compress`, `uncompress`, `zcat`

  - Data compression algorithm is LZW (Lampel-Ziv-Welch code).

  - Compression (suffix is .Z)

    ```
    compress file
    ```

    ```
    cat file| compress > file.Z
    ```

    ```
    tar cvf - -T list.txt | compress > archive.tar.Z
    ```

    ```
    tar -cvZf archive.tar.Z -T list.txt          # only GNU
    ```

  - Decompression

    ```
    uncompress file.Z
    ```

    ```
    zcat file.tar.Z | tar xvf -
    ```

    ```
    tar -xvf file.tar.Z                           # only GNU
    ```

Compression – gzip

- `gzip`, `gunzip`, `gzcat`
    - Data compression algorithm is LZ77 (Lempel-Ziv code)
    - Compression (default suffix is .gz)
      ```
      gzip file

      cat file | gzip > file.gz

      tar cvf - -T list.txt | gzip > archive.tar.gz

      tar -cvzf archive.tar.Z -T list.txt            # only GNU
      ```
    - Decompression
      ```
      gunzip file.gz

      gzcat file.tar.gz | tar xvf -

      tar -xvf file.tar.Z                            # only GNU
      ```

Compression – bzip2

- bzip2, bunzip2, bzcat
  - Data compression use combination of algorithms BWT (Burrows-Wheelerova transformation), MTF (Move-to-Front) transformation and Huffman code.

  - Compression (default suffix is .bz2)

    ```
    gzip file

    cat file | bzip2 > file.bz2

    tar cvf - -T list.txt | bzip2 > archive.tar.gz

    tar -cvJf archive.tar.Z -T list.txt          # only GNU
    ```

  - Decompression

    ```
    bunzip file.bz2

    bzcat file.tar.bz2 | tar xvf -

    tar -xvf file.tar.Z                          # only GNU
    ```

Archiving and compression – `zip`

- `zip`, `unzip`

    - Use format created by Philem Katzem (program PKZIP).

    - Creation of compress archive (default suffix is .zip)

      ```
      zip archive.zip files
      ```

      ```
      zip archive.zip -@ < list.txt
      ```

    - Listing of content

      ```
      unzip -l archive.zip
      ```

    - Extraction of archive

      ```
      unzip archive.zip [directories/files]
      ```

Archiving and compression – `fastjar`/`jar`

- `fastjar`/`jar` (Java ARchive tool)

  - Use formats ZIP and ZLIB.

  - Originally developed for archiving of JAVA packages.

  - Syntax similar like command `tar`.

  - Creation of compress archive (default suffix is .jar)
    ```
    fastjar -cvf archive.jar directories/files
    ```
    ```
    fastjar cvf elf.jar -@ < list.txt
    ```

  - Test (list) of archive
    ```
    fastjar -tvf archive.jar
    ```

  - Extraction of archive
    ```
    fastjar -xvf archive.jar
    ```

Integer arithmetic – exp

- expr *expression*
  - The utility evaluates the *expression* and writes the result to standard output.
  - Terms of the *expression* must be separated by blanks.
  - In front of shell meta-characters use character $\backslash$.

Operator	Meaning	Example
+	addition	N=$(expr $N1 + 3)
−	subtraction	N=$(expr $N1 − 3)
∗	multiplication	N=$(expr $N1 ∗ 3)
/	integer division	N=$(expr $N1 / 3)
%	remainder of integer division	N=$(expr $N1 % 3)

Integer arithmetic – exp

- Expression is evaluated by priority (like in mathematics)
  - first \(expression \),
  - after operations $*$, $/$, %,
  - at the end operations $+$ and $-$.
- Operations of the same priority are evaluated from left to right.

- Examples

```
~> A=$(expr 5 + 3 \* 2)
~> echo $A
11

~> A=$(expr \( 5 + 3 \) \* 2)
~> echo $A
16
```

Integer arithmetic – `let` or `(( ))`

- `let` *expression* or (((*expression*)))
    - Built-in commands (see `help let`).
    - Operands and operators may not be separated by spaces.
    - Variables are automatically replaced by their values
      (don't use character `$`).

Operator	Meaning	Example
$+$	addition	((N = N1 + 3))
$-$	subtraction	((N = N1 − 3))
$*$	multiplication	((N = N1 ∗ 3))
$/$	integer division	((N = N1 / 3))
%	remainder of integer division	((N = N1 % 3))
#	base	((N = 2#1011))
<<	bit left shifting	((N = 2#1011 << 3))
<<	bit right shifting	((N = 2#1011 >> 3))

Flouting point arithmetic – bc

- bc [-l] [*file*]
  - -l ... load mathematic library.
  - Commands are read from the *file* otherwise from the standard input.
  - Result is printed to the standard output.

Operator	Meaning	Example
+	addition	N=$(echo "$N1 + 3" | bc -l)
−	subtraction	N=$(echo "$N1 − 3" | bc -l)
∗	multiplication	N=$(echo "$N1 ∗ 3" | bc -l)
/	integer division	N=$(echo "$N1 / 3" | bc -l)
%	remainder of division	N=$(echo "$N1 % 3" | bc -l)
^	power	N=$(echo "2^3" | bc -l)
<, <=, >, >=	less than, ...	
==, ! =	equal to, ...	

Flouting point arithmetic – bc

Keyword	Meaning	Example
scale	decimal paces (default 0)	N=$(echo "scale=2 ; 10/3 " | bc -l)
ibase	input base	N=$(echo "ibase=16 ; A + B " | bc -l)
obase	output base	N=$(echo "obase=2 ; 5 + 2 " | bc -l)

Identifiers	Meaning	Example
x	variable (only lower case letters)	N=$(echo "x=2 ; y=5 ; x + y" | bc -l)
x[i]	i-th element of array x	N=$(echo "x[1]=2 ; x[1] + 3" | bc -l)
x(y,z)	function x with parameters y and z	N=$(echo "length(3.1415)" | bc -l)

Flouting point arithmetic – `bc`, `awk`

Function	Meaning	Example
`sqrt(x)`	square root of	N=$(echo "sqrt($A)" | bc -l)
`l(x)`	natural log	N=$(echo "l($A)" | bc -l)
`e(x)`	e^x	N=$(echo "e($A)" | bc -l)
`s(x)`	$sin(x)$	N=$(echo "s($A)" | bc -l)
`c(x)`	$cos(x)$	N=$(echo "c($A)" | bc -l)
`length(x)`	digits of number x	N=$(echo "c($A)" | bc -l)

- `awk`
  - Inside the program of commad `awk` we can make integer/flouting point arithmetic.