BIE-PS1 - Examples

Jan Trdlička

Faculty of Information Technology
Czech Technical University in Prague

trdlicka@fit.cvut.cz

January 3, 2018

Content

Input data from file

Example 1

Create a script that will have the filename as the first argument.
The script prints the number of all the different words of length 8, that have the same letter at the beginning and end of the word, to the standard output.

- Words can contains only uppercase and lowercase of alphabets.
- Line can contain multiple words.
- Treat bad arguments.
- Example of usage

```
$> ./s1.bash words.txt
2
```

1. Check of script arguments.
2. Change the data format: one word per line.
3. Finding words with a given property.

Input data from file

- Solution

```
#! /bin/bash

# Checking argument
if [ $# -ne 1 -o \! -f "$1" -o \! -r "$1" ]
then
  echo "Usage: $0 filename" >&2
  exit 2
fi

# Solution 1
tr -c '[a-zA-Z]' '[\n*]' < "$1" | grep -v '^$' | \
  sort -u | grep -c '^\(.\).....\1$'

# Solution 2
sed 's/[^a-zA-Z]/\n/g' "$1" | grep -v '^$' | \
  sort -u | grep -c '^\(.\).....\1$'
```

Input data from variable

Example 2

The shell variable `DATA` contains directory names separated by a colon. Create a script that prints the three most frequent suffixes of files from these directories (recursively) to standard output.

- The suffix of file is a string after the last dot in the filename (for example, the `abc.xyz.txt` file has the suffix `txt`, the `abcxyz` file has no suffix).
- Example of usage

```
$> export DATA="/tmp/abc:./x y z:/etc"
$> ./s2.bash
conf xml py
```

1. Temporary file management.
2. Find all common filenames.
3. Find all the different suffixes of these files and their frequency.

Input data from variable

- Solution

```
#! /bin/bash

# Prepare temporary file
FILE="/tmp/${USER}_$$.txt"

if [ -e "$FILE" ] ; then
  echo "Temporary file exists!!!" >&2
  exit 1
fi

# Get all filenames
IFS=":"
for D in $DATA
do
  #echo "$D"

  # Solution 1
  find "$D" -type f 2>/dev/null | sed 's:.*/\([^/]*\)$:\1:' >> "$FILE"

  # Solution 2
  #find "$D" -type f -exec basename {} \;  2>/dev/null >> "$FILE"
done

# Get top 3 suffixes
awk -F"." 'NF>1 {print $NF}' "$FILE" | sort | uniq -c | \
  sort -k1,1nr | awk 'NR<4 {printf("%s ", $NF)} END{printf("\n")}'

# Remove temporary file
rm "$FILE"
```

Input data – from process

Example 3

Create a script that will have the user login as your first argument. The script prints the number of threads that the user has run on the standard output.

- Check of script arguments.
- xample of usage

```
$> ./s3.bash root
138
```

1. Check of script arguments.
2. Find the number of threads.

Input data – from process

- Solution

```
#! /bin/bash

# Checking argument
if [ $# != 1 ] ; then
  echo "Usage: $0 username" >&2
  exit 1
fi

# Get number of threads

# Solution 1
ps -LU "$1" | tail -n +2 | wc -l

# Solution 2
#ps -eLo ruser | grep -c "$1"
```

Input data – from process

Example 4

Create a script that will have the user login as your first argument. The script writes to the standard output whether the user account in the system exists.

- Check of script arguments.
- Example of usage

```
$> ./s4.bash osy
Account "osy" exists.
```

```
$> ./s3.bash xyz
Account "xyz" doesn't exist.
```

1. Check of script arguments.
2. Verify account existence.

Input data – from process

- Solution

```
#! /bin/bash

# Checking argument
if [ $# != 1 ] ; then
  echo "Usage: $0 username" >&2
  exit 1
fi

# Solution 1
getent passwd | grep "^$1:" >/dev/null 2>&1

# Solution 2
#id "$1" >/dev/null 2>&1

if [ $? -eq 0 ] ; then
  echo "Account \"$1\" exists."
else
  echo "Account \"$1\" doesn't exist."
fi
```

Input data – from process

Example 5

Create a script that will have the command name as your first argument. The script will execute this command and, when it is finished, prints the 5 names of the system calls, that were most frequently called, to standard output.

- Example of usage

```
$> ./s5.bash date
mmap()
close()
fstat()
open()
read()
```

1. System call tracking command.
2. Obtaining system call names and their frequency.

Input data – from process

- Solution

```
#! /bin/bash

# Solution
strace "$1" 2>&1 1>/dev/null | \
  grep '^[^(]*(' | cut -d'(' -f1 | \
  sort | uniq -c | sort -k1,1nr | \
  head -5 | awk '{printf("%s()\n", $2)}'
```

Processes – signals

Example 6

Create a script that will have the process number PID as the first argument.
The script will send signal `SIGTERM` to all child processes of process that have a PID number.

- The script writes information about the signals it will send.
- Example of usage

```
$> ./s6.bash  11783
Signal SIGTERM will be send the following processes:
51439 sleep
51444 sleep
51449 sleep
```

1. Check of script arguments.
2. Finding child processes.
3. Sending signals.

Input data – from process

- Solution

```
#! /bin/bash

echo "$1" | grep '^[0-9]*$'>/dev/null 2>&1
ES=$?
if [ $# -ne 1 -o $ES -ne 0 ]
then
  echo "Usage: $0 PID" >&2
  exit 1
fi

# Solution
LIST="$(pgrep -lP $1)"
if [ -n "$LIST" ] ; then
  echo "Signal SIGTERM will be send \
  to the following processes:"
  pgrep -lP $1

  pkill -SIGTERM -P $1
fi
```