UNIX Shell Scripts
(Part 3)

Operating Systems 2019
Assist. Prof. Milos Jovanovik, PhD

Exam Tasks

Tasks from previous exams, as examples.

Task 1

- Write a shell script which will write out the total time a given user (first command line argument) has been logged in, in minutes.
- The script should write the output into the out.txt file.
- If the script has been invoked without any arguments, the script should print out a usage manual.
- If the output file already exists, it should be overwritten.
- In the end, the script needs to show the content of the output file, out.txt.

Task 1 – Analysis

```
121170    pts/25        92.53.4.153       Fri May  1 18:09 - 18:35  (00:26)
131513    pts/0         89.205.57.206     Fri May  1 18:08 - 20:29  (02:21)
131003    pts/0         79.141.126.75     Fri May  1 18:05 - 18:07  (00:01)
121079    pts/5         31.11.103.171     Fri May  1 18:02 - 21:01  (02:58)
125002    pts/24        78.157.14.93      Fri May  1 18:00 - 21:22  (03:21)
121021    pts/23        92.53.48.149      Fri May  1 17:58 - 18:29  (00:30)
141544    pts/22        85.30.78.174      Fri May  1 17:58 - 18:28  (00:30)
125018    pts/9         77.28.6.87        Fri May  1 17:55 - 21:20  (03:24)
131513    pts/20        89.205.57.206     Fri May  1 17:47 - 20:01  (02:14)
131004    pts/19        79.125.179.42     Fri May  1 17:47 - 18:37  (00:50)
141544    pts/17        85.30.78.174      Fri May  1 17:45 - 19:57  (02:12)
133011    pts/16        31.11.115.225     Fri May  1 17:39 - 18:34  (00:54)
131003    pts/5         79.141.126.75     Fri May  1 17:36 - 18:01  (00:24)
125015    pts/0         46.217.136.22     Fri May  1 17:33 - 18:03  (00:29)
133011    pts/16        31.11.115.225     Fri May  1 17:29 - 17:35  (00:05)
```

Task 1 – Solution

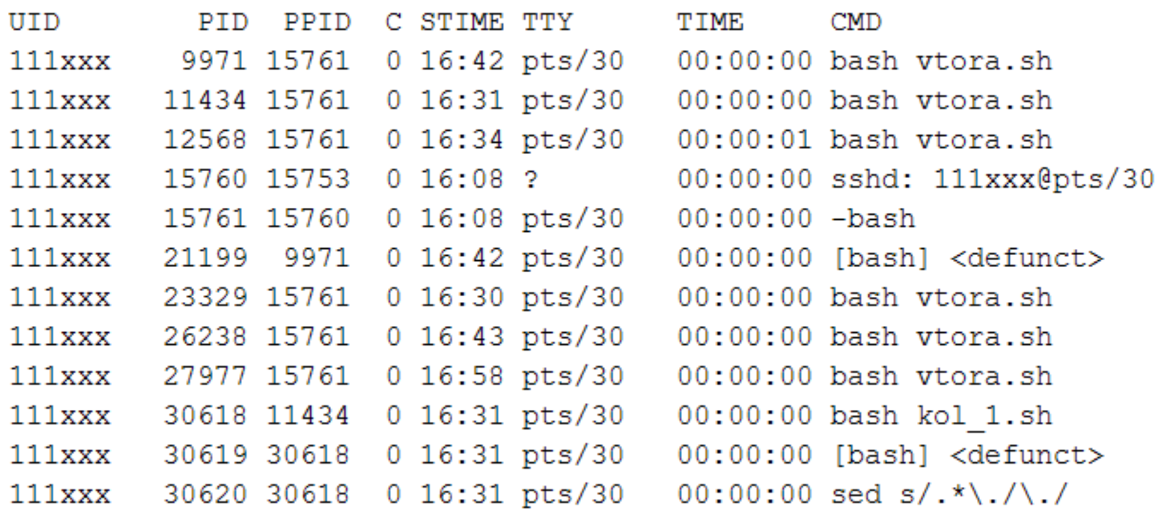
```
#!/bin/bash
if [ $# -lt 1 ]
then
   echo "USAGE: `basename $0` username"
   exit 1
fi

logins=`last | grep ^$1`
times=`echo "$logins" | awk '{print $10}'`
timesCleared=`echo "$times" | sed -e 's/(//' -e 's/)//'`
minutes=`echo "$timesCleared" | awk -F: '{ print $1*60+$2}'`

total=0
for m in $minutes
do
   total=$(( $total + $m ))
done

echo $total > out.txt
cat out.txt
```

Task 2

- Write a shell script which will write out the number of child processes each process of a given user has.
- The output should be written into out.txt.
- The username of the user is provided as the first command line argument.
- If there are no arguments on the command line, the script should print out a usage manual.
- If the output file already exists, it should be overwritten.
- In the end, the script needs to show the content of the output file, out.txt.

Task 2 – Analysis

- If the output of the 'ps' variant you should use is:



```
UID          PID  PPID  C STIME TTY          TIME     CMD
111xxx      9971 15761  0 16:42 pts/30   00:00:00 bash vtora.sh
111xxx     11434 15761  0 16:31 pts/30   00:00:00 bash vtora.sh
111xxx     12568 15761  0 16:34 pts/30   00:00:01 bash vtora.sh
111xxx     15760 15753  0 16:08 ?        00:00:00 sshd: 111xxx@pts/30
111xxx     15761 15760  0 16:08 pts/30   00:00:00 -bash
111xxx     21199  9971  0 16:42 pts/30   00:00:00 [bash] <defunct>
111xxx     23329 15761  0 16:30 pts/30   00:00:00 bash vtora.sh
111xxx     26238 15761  0 16:43 pts/30   00:00:00 bash vtora.sh
111xxx     27977 15761  0 16:58 pts/30   00:00:00 bash vtora.sh
111xxx     30618 11434  0 16:31 pts/30   00:00:00 bash kol_1.sh
111xxx     30619 30618  0 16:31 pts/30   00:00:00 [bash] <defunct>
111xxx     30620 30618  0 16:31 pts/30   00:00:00 sed s/.*\./\./
```

- The out.txt file should contain:

```
9971        1
11434       1
12568       0
15760       1
15761       6
21199       0
...
```

Task 2 – Solution

```
#!/bin/bash
if [ $# != 1 ]
then
   echo "USAGE: `basename $0` username"
   exit 1
fi

if [ -f out.txt ]
then
   rm out.txt
fi

for proc in `ps -ef | grep ^$1 | awk '{ print $2; }'`
do
   count=0
   for pproc in `ps -ef | grep ^$1 | awk '{ print $3; }'`
   do
        if [ $proc -eq $pproc ]
        then
                  count=$(( $count + 1 ))
        fi
   done
   echo "$proc $count" >> out.txt
done
cat out.txt
```

Task 3

- Write a shell script which will copy all files from a directory defined by the first command line argument which start with a number, followed by lower-case letters and which have the ‘.out’ extension, into a directory defined by the second command line argument.
- Then, calculate and print the total size of the copied files for which the user has execute permissions.
- If the arguments are missing, write a usage manual.
- If the directory denoted by the second command line argument does not exist, create it.

Task 3 – Solution ...

```
#!/bin/bash
if [ $# -lt 2 ]
then
  echo "USAGE: `basename $0` sourcefolder/
                            destinationfolder/"
  exit 1
fi

from=$1
to=$2

if [ ! -d $to ]
then
  mkdir $to
fi
```

Task 3 – ... Solution

```
files=`ls -l $from | grep '^-' | awk '{ print $9; }' |
  grep '^[0-9][a-z]*\.out$'`
for file in $files
do
  cp ${from}${file} ${to}${file}
done

filesX=`ls –l $to | grep '^-..x' | awk '{ print $5; }'`
total=0
for i in $filesX
do
  total=`expr $total + $i`
done

echo $total
```