**Bash Shell Scripts Assignment**

There are several very nice shell scripting resources on the web that contain many example scripts. Here are a couple of them:

[https://www.shellscript.sh/](http://www.freeos.com/guides/lsst/)

<http://www.freeos.com/guides/lsst/>

<http://www.tldp.org/LDP/abs/html/>

In the third resource, there are several exercises (in section O), and we have selected a few of them for you to try, so they are copied here, leaving in the links back into the tutorial. We have added a few notes/extensions.

During Lab today, work with a partner and try to do as many of these tasks as you can. Do them in any order that you want.

**Analyzing Scripts**

1. Examine the following script. Run it, and then explain what it does. Annotate the script and rewrite it in a more compact and elegant manner.

|  |
| --- |
| #!/bin/bash  MAX=10000  for((nr=1; nr<$MAX; nr++))  do  let "t1 = nr % 5"  if [ "$t1" -ne 3 ]  then  continue  fi  let "t2 = nr % 7"  if [ "$t2" -ne 4 ]  then  continue  fi  let "t3 = nr % 9"  if [ "$t3" -ne 5 ]  then  continue  fi  break # What happens when you comment out this line? Why?  done  echo "Number = $nr"  exit 0 |

This script finds the first number which is when dividing by 5, 7, 9 gives the corresponding reminders.

#!/bin/bash

MAX=10000

for((nr=1; nr<$MAX; nr++))

do

let "t1 = nr % 5"

let "t2 = nr % 7"

let "t3 = nr % 9"

if [ "$t1" -ne 3 ] || [ "$t2" -ne 4 ] || [ "$t3" -ne 5 ];

then

continue

fi

break # What happens when you comment out this line? Why?

done

echo "Number = $nr"

exit 0

1. Explain what the following script does. It is really just a parameterized command-line pipe. Rewrite the script to accept command line arguments for its 3 variables.

|  |
| --- |
| #!/bin/bash  DIRNAME=/usr/bin  FILETYPE="shell script"  LOGFILE=logfile  file "$DIRNAME"/\* | fgrep "$FILETYPE" | tee $LOGFILE | wc -l  exit 0 |

DIRNAME=$1

FILETYPE=$2

LOGFILE=$3

./pipe.sh /usr/bin "shell script" logfile

1. Analyze the following "one-liner" (here split into two lines for clarity) contributed by Rory Winston:

|  |
| --- |
| export SUM=0; for f in $(find src -name "\*.java");  do export SUM=$(($SUM + $(wc -l $f | awk '{ print $1 }'))); done; echo $SUM |

Hint: First, break the script up into bite-sized sections. Then, carefully examine its use of [double-parentheses](http://www.tldp.org/LDP/abs/html/dblparens.html) arithmetic, the [export](http://www.tldp.org/LDP/abs/html/internal.html#EXPORTREF) command, the [find](http://www.tldp.org/LDP/abs/html/moreadv.html#FINDREF) command, the [wc](http://www.tldp.org/LDP/abs/html/textproc.html#WCREF) command, and [awk](http://www.tldp.org/LDP/abs/html/awk.html#AWKREF).

Double parentheses are used to perform arithmetic operations. Export is to make the variable SUM available for other subprocesses. Find command finds the files with java extension. Wc counts the number of lines in each file. Awk puts the output of wc command into variable $1.

**Writing Scripts**

**Self-reproducing Script**

Write a script that backs itself up, that is, copies itself to a file named backup.sh.

cat talk.sh > backup.sh

Hint: Use the [cat](http://www.tldp.org/LDP/abs/html/basic.html#CATREF) command and the appropriate [positional parameter](http://www.tldp.org/LDP/abs/html/othertypesv.html#SCRNAMEPARAM) (command-line argument).

Next, create a more general version of this: there may be any number of filename arguments. Back up each file specified (including the script itself) into a new file named by appending “.backup” to the original filename.

**Converting**[**for**](http://www.tldp.org/LDP/abs/html/loops1.html#FORLOOPREF1)**loops to**[**while**](http://www.tldp.org/LDP/abs/html/loops1.html#WHILELOOPREF)**and**[**until**](http://www.tldp.org/LDP/abs/html/loops1.html#UNTILLOOPREF)**loops**

Convert the  *for loops* in [Example 11-1](http://www.tldp.org/LDP/abs/html/loops1.html#EX22) to *while loops*. Hint: store the data in an [array](http://www.tldp.org/LDP/abs/html/arrays.html#ARRAYREF) and step through the array elements.

#!/bin/bash

array=( Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune Pluto )

index=0

while [ "$index" -lt "${#array[\*]}" ]

do

echo ${array[index]}

let "index = $index + 1"

done

Having already done the "heavy lifting," now convert the loops in the example to *until loops*.

#!/bin/bash

array=( Mercury Venus Earth Mars Jupiter Saturn Uranus Neptune Pluto )

index=1

until [ "$index" = "${#array[\*]}" ]

do

echo ${array[index]}

let "index = $index + 1"

done

**Changing the line spacing of a text file**

Write a script that reads each line of a target file, then writes the line back to stdout, but with an extra blank line following. This has the effect of double-spacing the file.

Include all necessary code to check whether the script gets the necessary command-line argument (a filename), and whether the specified file exists.

When the script runs correctly, modify it to triple-space the target file.

Finally, write a script to remove all blank lines from the target file, single-spacing it.

**Backwards Listing**

Write a script that echoes itself to stdout, but backwards.

**Checking whether a process is still running**

Given a [process ID](http://www.tldp.org/LDP/abs/html/special-chars.html#PROCESSIDREF) (*PID*) as an argument, this script will check, at user-specified intervals, whether the given process is still running. You may use the [ps](http://www.tldp.org/LDP/abs/html/system.html#PPSSREF) and [sleep](http://www.tldp.org/LDP/abs/html/timedate.html#SLEEPREF) commands.

**Lottery Numbers**

One type of lottery involves picking five different numbers, in the range of 1 - 50. Write a script that generates five pseudorandom numbers in this range, with no duplicates. The script will give the option of echoing the numbers to stdout or saving them to a file, along with the date and time the particular number set was generated. (If your script consistently generates winning lottery numbers, then you can retire on the proceeds and leave shell scripting to those of us who have to work for a living.)

**Primes**

Print (to stdout) all prime numbers between 60000 and 63000. The output should be nicely formatted in columns (hint: use [printf](http://www.tldp.org/LDP/abs/html/internal.html#PRINTFREF)).