**Quiz 4**

1. Write a shell script to find the number of lines in a list of files using sed.

#!/bin/bash

# Counting the number of lines in a list of files

# for loop over arguments

if [ $# -lt 1 ]

then

echo "Usage: $0 file ..."

exit 1

fi

echo "$0 counts the lines of code"

l=0

n=0

s=0

for f in $\*

do

l=`wc -l $f | sed 's/^\([0-9]\*\).\*$/\1/'`

echo "$f: $l"

n=$[ $n + 1 ]

s=$[ $s + $l ]

done

echo "$n files in total, with $s lines in total"

2. Write a shell script to substitute one pattern for another in a text file.

#!/bin/bash

# subst.sh: a script that substitutes one pattern for

#+ another in a file,

#+ i.e., "sh subst.sh Smith Jones letter.txt".

# Jones replaces Smith.

ARGS=3 # Script requires 3 arguments.

E\_BADARGS=85 # Wrong number of arguments passed to script.

if [ $# -ne "$ARGS" ]

then

echo "Usage: `basename $0` old-pattern new-pattern filename"

exit $E\_BADARGS

fi

old\_pattern=$1

new\_pattern=$2

if [ -f "$3" ]

then

file\_name=$3

else

echo "File \"$3\" does not exist."

exit $E\_BADARGS

fi

# -----------------------------------------------

# Here is where the heavy work gets done.

sed -e "s/$old\_pattern/$new\_pattern/g" $file\_name

# -----------------------------------------------

# 's' is, of course, the substitute command in sed,

#+ and /pattern/ invokes address matching.

# The 'g,' or global flag causes substitution for EVERY

#+ occurence of $old\_pattern on each line, not just the first.

# Read the 'sed' docs for an in-depth explanation.

exit $? # Redirect the output of this script to write to a file.

3. Write a shell script to print complete pathname associated with pid. User has to pass the PID from command line.

#!/bin/bash

# pid-identifier.sh:

# Gives complete path name to process associated with pid.

ARGNO=1 # Number of arguments the script expects.

E\_WRONGARGS=65

E\_BADPID=66

E\_NOSUCHPROCESS=67

E\_NOPERMISSION=68

PROCFILE=exe

if [ $# -ne $ARGNO ]

then

echo "Usage: `basename $0` PID-number" >&2 # Error message >stderr.

exit $E\_WRONGARGS

fi

pidno=$( ps ax | grep $1 | awk '{ print $1 }' | grep $1 )

# Checks for pid in "ps" listing, field #1.

# Then makes sure it is the actual process, not the process invoked by this script.

# The last "grep $1" filters out this possibility.

if [ -z "$pidno" ] # If, after all the filtering, the result is a zero-length string,

then #+ no running process corresponds to the pid given.

echo "No such process running."

exit $E\_NOSUCHPROCESS

fi

# To simplify the entire process, use "pidof".

if [ ! -r "/proc/$1/$PROCFILE" ] # Check for read permission.

then

echo "Process $1 running, but..."

echo "Can't get read permission on /proc/$1/$PROCFILE."

exit $E\_NOPERMISSION # Ordinary user can't access some files in /proc.

fi

# The last two tests may be replaced by:

# if ! kill -0 $1 > /dev/null 2>&1 # '0' is not a signal, but

# this will test whether it is possible

# to send a signal to the process.

# then echo "PID doesn't exist or you're not its owner" >&2

# exit $E\_BADPID

# fi

exe\_file=$( ls -l /proc/$1 | grep "exe" | awk '{ print $11 }' )

# Or exe\_file=$( ls -l /proc/$1/exe | awk '{print $11}' )

#

# /proc/pid-number/exe is a symbolic link

#+ to the complete path name of the invoking process.

if [ -e "$exe\_file" ] # If /proc/pid-number/exe exists,

then #+ then the corresponding process exists.

echo "Process #$1 invoked by $exe\_file."

else

echo "No such process running."

fi

exit 0

4. Write a shell script to print all users on system using awk.

#!/bin/bash

# userlist.sh

PASSWORD\_FILE=/etc/passwd

n=1 # User number

for name in $(awk 'BEGIN{FS=":"}{print $1}' < "$PASSWORD\_FILE" )

# Field separator = : ^^^^^^

# Print first field ^^^^^^^^

# Get input from password file /etc/passwd ^^^^^^^^^^^^^^^^^

do

echo "USER #$n = $name"

let "n += 1"

done

5. Write a shell script to list the frequency of words used in a file.

#!/bin/bash

# wf.sh: Crude word frequency analysis on a text file.

# This is a more efficient version of the "wf2.sh" script.

# Check for input file on command-line.

ARGS=1

E\_BADARGS=85

E\_NOFILE=86

if [ $# -ne "$ARGS" ] # Correct number of arguments passed to script?

then

echo "Usage: `basename $0` filename"

exit $E\_BADARGS

fi

if [ ! -f "$1" ] # Check if file exists.

then

echo "File \"$1\" does not exist."

exit $E\_NOFILE

fi

sed -e 's/\.//g' -e 's/\,//g' -e 's/ /\

/g' "$1" | tr 'A-Z' 'a-z' | sort | uniq -c | sort -nr

# =========================

# Frequency of occurrence

# Filter out periods and commas, and

#+ change space between words to linefeed,

#+ then shift characters to lowercase, and

#+ finally prefix occurrence count and sort numerically.

exit 0

6. Write a script to take backup of files changed in last 24 hours and archive them.

#!/bin/bash

# Backs up all files in current directory modified within last 24 hours

BACKUPFILE=backup-$(date +%m-%d-%Y)

archive=${1:-$BACKUPFILE}

# If no backup-archive filename specified on command-line,

#+ it will default to "backup-MM-DD-YYYY.tar.gz."

tar cvf - `find . -mtime -1 -type f -print` > $archive.tar

gzip $archive.tar

echo "Directory $PWD backed up in archive file \"$archive.tar.gz\"."

7. Write a shell script to determine if a particular service is active or not. For eg: if SSH service is active it should display yes and vice versa

*#!/bin/bash*

ports="22 80 25"

service="SSH WEB MAIL"

c=1

echo "Running services status:"

/bin/netstat -tulpn | grep -vE '^Active|Proto' | while read LINE

do

*# get active port name and use : as delimiter*

t=$(echo $LINE | awk '{ print $4}' | cut -d: -f2)

[ "$t" == "" ] && t=-1 || :

*# get service name from $services and : as delimiter*

sname=$(echo $service | cut -d' ' -f$c)

sstatus="$sname: No"

*# now compare port*

for i in $ports

do

if [ $i -eq $t ]; then

sstatus="$sname: Ok"

fi

done

*# display service status as OK or NO*

echo "$sstatus"

*#next service please*

c=$( expr $c + 1 )

*# break afer 3 services*

[ $c -ge 4 ] && break || :

done

8. Write a shell sccript to remove spaces from filenames and replace it with underscore

#!/bin/bash

find . -type f -name "\* \*" | while read file

do

mv "$file" "${file// /\_}"

done

9. Write a shell script which prints the df output in more formatted way as below

Filesystem Size Used Avail Capacity Mounted

/dev/sda1 446.71G 18.11G 405.88G 5% /

udev 10M 0 10M 0% /dev

tmpfs 1.14G 9.16M 1.13G 1% /run

#!/bin/sh

# newdf - a friendlier version of df

sedscript="/tmp/newdf.$$"

trap "rm -f $sedscript" EXIT

cat << 'EOF' > $sedscript

function showunit(size)

{ mb = size / 1024; prettymb=(int(mb \* 100)) / 100;

gb = mb / 1024; prettygb=(int(gb \* 100)) / 100;

if ( substr(size,1,1) !~ "[0-9]" ||

substr(size,2,1) !~ "[0-9]" ) { return size }

else if ( mb < 1) { return size "K" }

else if ( gb < 1) { return prettymb "M" }

else { return prettygb "G" }

}

BEGIN {

printf "%-27s %7s %7s %7s %8s %-s\n",

"Filesystem", "Size", "Used", "Avail", "Capacity", "Mounted"

}

!/Filesystem/ {

size=showunit($2);

used=showunit($3);

avail=showunit($4);

printf "%-27s %7s %7s %7s %8s %-s\n",

$1, size, used, avail, $5, $6

}

EOF

df -k | awk -f $sedscript

exit 0

10. Write a shell script to summarize available disk space and present in a logical aand readable fashion.

#!/bin/sh

# diskspace - summarize available disk space and present in a logical

# and readable fashion

tempfile="/tmp/available.$$"

trap "rm -f $tempfile" EXIT

cat << 'EOF' > $tempfile

{ sum += $4 }

END { mb = sum / 1024

gb = mb / 1024

printf "%.0f MB (%.2fGB) of available disk space\n", mb, gb

}

EOF

df -k | awk -f $tempfile

exit 0

11. Write a shell function to rename .txt files to .text

txt2text (){

for i in `ls -1`

do

mv $i `echo $i | sed 's/.\*\.txt$/.text/'`

done

}