**Quiz 3**

1. Write a script to delete blank lines in a file

#!/bin/csh -f

#

#-------- delete blank lines from a file

# usage: clnblanklines file

#

set fn="$1"

#

mv -f $1 "$fn".was

cat "$fn".was | awk '$0!~/^$/ {print $0}' > $fn

echo " ---> remove" "$fn".was "?"

rm -i "$fn".was

2. Write a script to delete duplicate lines

#!/bin/ksh

#(needs ksh, not sh, for arithmetic)

#

# clnduplines - clean duplicate lines in a file

# original file is saved in file.was

# (same as cntduplines but file replaced by cleaned file)

#

if [ -z "$1" ]; then

echo

echo "syntax: clnduplines [filename]"

echo

exit

fi

file=$1

echo " --- clnduplines: Searching for duplicate lines in $file ---"

count=0

cat /dev/null > $file.clean

while read line

do

found=`grep "$line" $file.clean`

if [ -n "$found" ]; then

echo ". \c"

(( count = count + 1 )) ## ksh arithmetic

else

echo "$line" >> $file.clean

fi

done < $file

echo ""

echo " $count dup lines found"

#---------- sh version, too slow --------

#echo " ... counting lines ..."

## old=`wc -l $file | awk '{ print $1 }'`

# old=`cat $file | wc -l`

## new=`wc -l $file.clean | awk '{ print $1 }'`

# new=`cat $file.clean | wc -l`

## VERY slow: echo " ---> `expr $old - $new` dups found"

## that's why used ksh's arithmetic: (( ))

echo ""

/bin/mv $file $file.was

/bin/mv $file.clean $file

echo " --- $file is now clean, original in $file.was --- delete it ?"

/bin/rm -i $file.was

3. Write a shell script to delete a directory tree.

#!/bin/sh

#------- delete a dir tree

# usage: deldir dirname

#

set dir = $1

echo " "

if( "$1" == "" ) then

echo -n " ---> deldir dirname ---> Enter a dirname: "

set dir = &<

endif

/bin/rm -f -r "$dir"/\*

sleep 1

rmdir $dir

4. Write a shell script to see if a process is running.

Code snippet.

echo "Checking if process $1 exists..."

[ "$1" = "" ] && return 0

PROCESS\_NUM=$(ps -ef | grep "$1" | grep -v "grep" | wc -l)

if [ $PROCESS\_NUM -ge 1 ];

then

echo “Process is not running”

else

echo “Process is running”

fi

5. Write a shell script to display the following:

Hostname, disk space usage, free & used memory, uptime and logged in users.

#!/bin/bash

# Sample script written for Part 4 of the RHCE series

# This script will return the following set of system information:

# -Hostname information:

echo -e "\e[31;43m\*\*\*\*\* **HOSTNAME INFORMATION** \*\*\*\*\*\e[0m"

hostnamectl

echo ""

# -File system disk space usage:

echo -e "\e[31;43m\*\*\*\*\* **FILE SYSTEM DISK SPACE USAGE** \*\*\*\*\*\e[0m"

df -h

echo ""

# -Free and used memory in the system:

echo -e "\e[31;43m \*\*\*\*\* **FREE AND USED MEMORY** \*\*\*\*\*\e[0m"

free

echo ""

# -System uptime and load:

echo -e "\e[31;43m\*\*\*\*\* **SYSTEM UPTIME AND LOAD** \*\*\*\*\*\e[0m"

uptime

echo ""

# -Logged-in users:

echo -e "\e[31;43m\*\*\*\*\* **CURRENTLY LOGGED-IN USERS** \*\*\*\*\*\e[0m"

who

echo ""

# -Top 5 processes as far as memory usage is concerned

echo -e "\e[31;43m\*\*\*\*\* **TOP 5 MEMORY-CONSUMING PROCESSES** \*\*\*\*\*\e[0m"

ps -eo %mem,%cpu,comm --sort=-%mem | head -n 6

echo ""

echo -e "\e[1;32mDone.\e[0m"

6. Write a shell script to display syntax of a given command

for path in `echo "$PATH" | sed 's/^:/.:/;s/:$/:./;s/:/ /g'`

do

[ -x "$path/gawk" ] && : ${NAWK=$path/gawk}

[ -x "$path/nawk" ] && : ${NAWK=$path/nawk}

done

: ${NAWK=awk}

Usage () {

echo >&2 "$PN - show syntax of a command, $VER (stv '94)

usage: $PN command [command ...]"

exit 1

}

[ $# -lt 1 -o "$1" = "-h" ] && Usage

man "$@" 2>/dev/null |

col -b |

$NAWK '($1 ~ /^[sS][yY][nN][oO][pP][sS][iI][sS]/) {

while ( getline )

{

# assume the next section starts with at least two

# characters in upper case in column 1

if ( $0 ~ /^[A-Z][A-Z][A-Z]\*$/ ) exit 0 # next section

print

}

}

'

7. Write a shell to check if a command is in PATH directory list

#!/bin/sh

# inpath - verify that a specified program is either valid as-is,

# or can be found in the PATH directory list.

in\_path()

{

# given a command and the PATH, try to find the command. Returns

# 0 if found and executable, 1 if not. Note that this temporarily modifies

# the the IFS (input field seperator), but restores it upon completion.

cmd=$1 path=$2 retval=1

oldIFS=$IFS IFS=":"

for directory in $path

do

if [ -x $directory/$cmd ] ; then

retval=0 # if we're here, we found $cmd in $directory

fi

done

IFS=$oldIFS

return $retval

}

checkForCmdInPath()

{

var=$1

# The variable slicing notation in the following conditional

# needs some explanation: ${var#expr} returns everything after

# the match for 'expr' in the variable value (if any), and

# ${var%expr} returns everything that doesn't match (in this

# case just the very first character. You can also do this in

# Bash with ${var:0:1} and you could use cut too: cut -c1

if [ "$var" != "" ] ; then

if [ "${var%${var#?}}" = "/" ] ; then

if [ ! -x $var ] ; then

return 1

fi

elif ! in\_path $var $PATH ; then

return 2

fi

fi

}

if [ $# -ne 1 ] ; then

echo "Usage: $0 command" >&2 ; exit 1

fi

checkForCmdInPath "$1"

case $? in

0 ) echo "$1 found in PATH" ;;

1 ) echo "$1 not found or not executable" ;;

2 ) echo "$1 not found in PATH" ;;

esac

exit 0

8. Write a shell script to transfer a file using ftp.

# Description

# - transfers the file to the given host using FTP (binary

# mode)

# - To automate the transfer, a temporary file "$HOME/.netrc"

# will be created (if not already present)

PN=`basename "$0"` # Program name

VER='1.2'

Netrc=$HOME/.netrc

if [ \( $# -lt 1 \) -o \( ! -r "$1" \) ]

then

echo >&2 "usage: $PN file [hostname]"

exit 1

fi

File="$1"

Host="${2:-`uname -n`}"

Base=`basename "$File"`

# If $HOME/.netrc is readable and contains the target host name,

# try an automatic FTP transfer. Otherwise, create the file

# temporarily, and remove it later on.

[ -r $Netrc ] && grep "$Host" $Netrc > /dev/null || {

User=`id | sed 's/uid=[0-9][0-9]\*(\([^)]\*\)).\*/\1/'`

echo -n "$PN: password ($User@$Host): "

stty -echo

read PW || { stty echo; exit 1; }

stty echo

echo " OK"

if [ -r "$Netrc" ]

then # Create backup of old .netrc

mv "$Netrc" "$Netrc".$$

trap 'mv "$Netrc".$$ "$Netrc"' 0

else

> "$Netrc"

trap 'rm -f "$Netrc"' 0

fi

trap "exit 2" 1 2 3 15

echo "machine $Host login $User password $PW" >> "$Netrc" || exit 1

chmod 600 "$Netrc"

ls -l "$Netrc"

}

ftp $Host <<!

bin

put $File $Base

quit

!

9. Write a shell script to delete files older than a week.

# Description

# Delete files older than a specified number of days.

##########################################################################

PN=`basename "$0"` # Program name

VER='1.2'

Days=7 # remove files older than 7 days

usage () {

echo "$PN - remove temporary files/directories, $VER (stv '92)

usage: $PN [-d days] directory [directory ...]

All files and subdirectories older than the specified number of days

(default $Days) are deleted." >&2

exit 1

}

msg () {

for i

do echo "$PN: $i" >&2

done

}

fatal () { msg "$@"; exit 1; }

set -- `getopt hd: "$@"` || usage

while [ $# -gt 0 ]

do

case "$1" in

-d) case "$2" in

[0-9]\*) Days="$2"; shift;;

\*) fatal "illegal number of days: $2";;

esac;;

--) shift; break;;

-h) usage;;

-\*) usage;;

\*) break;; # First file name

esac

shift

done

[ $# -lt 1 ] && usage

for i

do

if [ ! -d "$i" ]

then msg "no directory: $i"

elif [ ! -r "$i" -o ! -w "$i" ]

then msg "access denied (need read- and write access): $i"

else

# Find files, links, FIFOs or sockets

find "$i" ! -type d -mtime +$Days -print | xargs rm -f > /dev/null 2>&1

# remove all empty subdirectories

[ -d "$i" ] || continue

(cd $i || continue

for d in \*

do

[ -d "$d" ] || continue

find . -depth -type d -mtime +$Days -print |

xargs rmdir > /dev/null 2>&1

done

)

fi

done

exit 0

10. Write a shell script to implement -i version of cp command

# Description

# If "cp" is called with two files as arguments, it will

# overwrite the second file. This script asks for

# confirmation before doing so.

##########################################################################

PN=`basename "$0"` # Program name

VER='1.5'

echon () {

if [ X"$ECHON" = X ]

then

# Determine how to "echo" without newline: "echo -n" or "echo ...\c"

if [ X`echo -n` = X-n ]

then ECHON=echo; NNL="\c"

else ECHON="echo -n"; NNL=""

fi

fi

$ECHON "$\*$NNL"

}

Verbose=false

Force=false

Args=

while [ $# -gt 0 ]

do

case "$1" in

-v) Verbose=true;;

--) shift; break;;

-f) Force=true

Args="${Args:+$Args }$1"

;;

-\*) Args="${Args:+$Args }$1";;

\*) break;; # First file name

esac

shift

done

# Special case: "cp file1 file2": ask for confirmation

if [ $# -eq 2 ] && [ $Force = false ] && [ -f "$2" ] && [ -s "$2" ]

then

while :

do

echon >&2 "$PN: overwrite file <$2> (j/y/n)? "

read OK < /dev/tty || exit 0

case "$OK" in

[jJyY]\*) break;;

[nN]\*) echo >&2 "$PN: nothing copied."; exit 0;;

esac

done

fi

if [ $Verbose = true ]

then

for target # find last argument

do :

done

for file

do

[ x"$file" = x"$target" ] && continue

echo >&2 "$file"

cp $Args "$file" "$target"

done

else

exec cp $Args "$@"

fi

11. Write a shell script to change the group owner of files in a directory to another group.

#!/bin/ksh

#

# Shell script to change group of files that belong to a named group to a new named group.

# Searches down the directory tree from current directory and changes all

# files that belong to the named group only.

# You must have permission to change the group of the files and the

# new group must exist !!!

if [ $# != 2 ]

then echo "Usage : ch-grp oldgroup newgroup"

exit

fi

oldgroup=$1

newgroup=$2

tmpfile=/tmp/change-group$$$$

find ./ -group $oldgroup > $tmpfile

{

while read files

do

oldgrp=`ls -l $files | awk '{ print $4 } '`

chgrp $newgroup $files 2> /dev/null 1> /dev/null

if [ $? -ne 0 ]

then echo "Error changing group of "$files

else newgrp=`ls -l $files | awk '{ print $4 } '`

echo $files" has been changed from " $oldgrp " to " $newgrp

fi

done } < $tmpfile

rm $tmpfile

12. Write a shell script to list files in size order, smallest first.

if [ -n "$\*" ]

then

ls -1 -al $\* | sort -nbk5

else

ls -al | sort -nbk5

fi