Assignment 2

1. “cat x y > y” concatenates the contents of file “x” into file “y”, overwriting “y.” If it didn’t give an error message, it would also try to concatenate file “y” into itself. With a very large input file, this process could run indefinitely until the disk is full. Similarly, “cat x >> x” would append the contents of “x” onto the end of itself until the disk is full.
2. The “cx” script runs “chmod +x” on every supplied filename argument, making each of these files executable by the user. If there are no filenames given, the script prints an error message.

#!/bin/bash

# cx: executes the command "chmod +x" on every supplied file

# check if arguments are supplied

if [[ $# -ne 0 ]] ; then

# run chmod if arguments are given

exec /usr/bin/chmod +x "$@"

else

# print an error if there are no files given

echo 'cx: missing filename arguments'

fi

1. The “nf” script prints just the number of files in the current directory. It pipes the output of the “ls” command into the command “wc -l” (count lines) to count the number of files.

#!/bin/bash

# nf: prints the number of files in the current directory

# pipe output of ls into wc -l in order to print the number of files

exec /usr/bin/ls | /usr/bin/wc -l

1. The “lss” script lists all the files in the current directory in decreasing order of file size. It simply prints the output of running the command “ls -lS”.

#!/bin/bash

# lss: lists all the files in the current directory in order of decreasing size

# use ls with options l and S to display files ordered by size

exec /usr/bin/ls -lS

1. The “whoson” script prints a sorted list of undergrad students logged in on the current machine. It loops over the output space-delimited string from the “users” command, using the command “groups $user” piped into “grep -o ‘ugrad’” to check if the user is in the “ugrad” group.

#!/bin/bash

# whoson: display a sorted list of undergrad students logged in on current machine

# keep a string of all ugrad users

ugrads=""

# loop over all logged on users, adding user to ugrads if the user is in the group

for user in $(exec /usr/bin/users)

do

# pipe output of "groups $user" into "grep -o 'ugrad'" to get a string

# containing "ugrad" only if the user is in the ugrad group

if [[ $(exec groups $user 2>/dev/null | grep -o 'ugrad' ) = "ugrad" ]] ; then

ugrads="$ugrads $user"

fi

done

# output the list of ugrad users

echo $ugrads

1. The “howmany” script prints the number of undergrad students logged in on the current machine. It pipes the output of the “whoson” script into the command “wc -w” (count words) in order to get the number of undergrads.

#!/bin/bash

# howmany: display the number of undergrad students logged in on the machine

# set up the PATH variable so this script can reasonably find the whoson script

# if the whoson script isn't in one of the appended dirs then this script can't be run

oldPath=$PATH

PATH=$PATH:/bin:/usr/bin:$HOME/bin:.

# pipe the output of the "whoson" script into "wc -w" to count number of words (users)

exec whoson | wc -w

# restore old path

PATH=$oldPath

1. The “valid” script determines if the argument is a valid shell variable name (begins with an underscore or letter and contains only alphanumeric characters). If there is more than one input argument, then the script ignores the extra arguments and prints a message to stderr, but still runs on the first argument. It uses the “grep” command with extended regex expressions to identify if the argument is valid.

#!/bin/bash

# valid: determines if the argument is a valid shell variable name

# ensure at least one argument is supplied

if [[ $# -eq 0 ]] ; then

echo 'valid: supply one argument'

else

# indicate in stderr that only the first argument is checked

if [[ $# -gt 1 ]] ; then

>&2 echo 'valid: ignoring extra arguments'

fi

# pipe the first arg name into grep using echo, and check it for validity using grep

# and a regex expression

if [[ $(echo $1 | grep -Eo "[\_a-zA-Z][\_a-zA-Z0-9]\*") = $1 ]] ; then

echo 'yes'

else

echo 'no'

fi

fi

1. The “prargs” script prints out a numbered list of its arguments, which includes the name of the script. If executed with no command line arguments, it will print only the name of the script. Arguments within quotations are kept together in the output.

#!/bin/bash

# prargs: prints a numbered list of the arguments passed to it

# print argument 0 ( the name of the script )

echo "0: \"$0\""

# keep count of args

i=1

# loop over all arguments, printing its position in the arg list

# and the name of the arg

for arg in "$@"

do

echo "$i: \"$arg\""

i=$((i+1))

done