4. Finding Files.

The output is as following:

Alias dfr='find . -maxdepth 1 -mtime -1 -o -ctime -1'

'Alias dfr =' is call the command after the equal sign 'Alias dfr'. 'find .' means find every file that match the following criteria. '-maxdepth 1' is the search to the deeps of subfolders 1 level lower. '-mtime' is the last modification time. The '-1' here is at most n days. '-o' is the same as '-or' It means the file has to satisfy the condition Infront of it or the condition behind it. '-ctime' is the time since the last status change. The '-1' here is at most n days before.

5. Directory Checksum.

```
#!/bin/sh

#
# ds - Compute checksum of all files under specified directory.
#
# 22Jan11   Added -noleaf to find command
# 29Dec10   Modified to unset CDPATH before cd
# 13Jan10   Modified to deal with spaces in directory names
# 05Oct09   Modified to deal with spaces in filenames and specification of
#                a directory on the command line
# 23Nov08   Written by Everett Lipman
#
```

These lines start with # are just commons. It is telling us what ds script do and when it is been changed.

SUMCOMMAND=md5sum

We are assigning 'md5sum', which is a command that calculate md5 value, to 'SUMCOMMAND'

PROGNAME=`basename $0`

We set the 'ds' to be the 'PROGNAME'. In other words, we set the name of the program to be 'ds'.

USAGE="$PROGNAME [directory]"

We set 'USAGE' to be the name of the program and the '[directory]' is just the string as it is.

NARGSa=0

Crate variable 'NARGSa' and assign value 0 to this variable.

NARGSb=1

Crate variable 'NARGSb' and assign value 1 to this variable.

error_exit()

This crate a function called 'error exit' this function is used to display errors.

{

This tells shell this is the start of the function

echo ""

Prints out an empty line.

echo $1

Print out a string where ‘$1’ would be replaced by the content or value of ‘1’

echo ""

Prints out an empty line.

echo "$PROGNAME exiting."

Print out string ‘PROGNAME exiting’ where ‘$PROGNAME’ would be the name of the program.

echo ""

Prints out an empty line.

exit 1

}

This tells shell this is the end of the function.

errorcheck()

This crate a function called ‘errorcheck’ this function is used to check wither there is errors and would print out something if there are errors.

{

This tells shell this is the start of the function

if [$? -ne 0]

this is an if function if the condition in the [] met shell will run of things after ‘then’. ‘$?’ means to carry the value ‘1’ that had been exit in the ‘error_exit’ function to here. ‘-ne’ means not equal. ‘0’ is just the value ‘0’. Here it is saying if the value ‘error_exit’ return is not 0 then it will run the thing after ‘then’.

then

This is just a logistical word in shell that work together with ‘if’.

error_exit "Error $1"

If the condition after ‘if’ is true then this call the function error_exit above and print out “Error $1” where ‘$1’ is the content or value of ‘1’

fi

This work together with the ‘if’ statement. If fi statement let shell to make judgement and run the right script accordingly.

}

This tells shell this is the end of the function.

if [$# -ne $NARGSa -a $# -ne $NARGSb]

this is another ‘if’ statement. ‘$#’ means the number of parameters which the script has been called. ‘-ne’ is not equal to. ‘$NARGSa’ is the value which is ‘0’ we signed to ‘NARGSa’ at the very beginning. ‘-a’ is and. This means both the condition before ‘-a’ and after ‘-a’ has to be true in order to make the if condition true. ‘$#’ means the number of parameters which the script has been called. ‘-ne’ is not equal to. ‘$NARGSb’ is the value which is ‘1’ we signed to ‘NARGSa’ at the very beginning.

This line is saying if the number of parameters when run inning this script is not 0

then

This is just a logistical word in shell that work together with ‘if’.

error_exit "usage: $USAGE"

if the condition above had been met then it will call the function ‘error_exit’ which we defined before. ‘$USAGE’ is calling the value we assigned to ‘USAGE’ before.
The “usage: $USAGE” is what this if statement is going to print out. It would print out “usage: and the value we assigned to ‘$USAGE’ which is the ‘name of this program (cd)+[directory]’”.

fi

This work together with the ‘if’ statement. If fi statement let shell to make judgement and run the right script accordingly.

echo

print out an empty line.

if [$# -eq 1]

This is another if function. ‘$#’ means the number of parameters which the script has been called. ‘-eq’ means equal. ‘1’ is the value it has to equal to. This line is saying if the script has been called one time then it run script after ‘then’.

then

This is just a logistical word in shell that work together with ‘if’.

unset CDPATH

unset the variable ‘CDPATH’ which means it clears the ‘CDPATH’ directory.

-

cd "$1" > /dev/null

Redirect ‘$1’ to directory ‘/dev/null’.

errorcheck "changing to specified directory."

call function ‘errorcheck’ with parameter ‘changing to specified directory’ just like how we called function before.

fi

This work together with the ‘if’ statement. If fi statement let shell to make judgement and run the right script accordingly.

find . -noleaf -type f | sort | while read i   # deal with spaces in filenames

find file in current directory from ‘-noleaf’. ‘-type f’ means regular file type. ‘sort’ means sort the file in alphabetical order. ‘while read I’ is a while loop that read every file that satisfy the condition before. This line is saying find all file with regular file type in current directory and read the content of all files.

do

cat "$i"

‘cat’ means touch content of the file. This line means touch the content of every file.

done | $SUMCOMMAND

end the script and loop and give the content we just get above to ‘SUMCOMMAND’ which we defined at the very beginning.

echo
print out an empty line.