**Lesson 13 – More Bash programming**

**Setup:** Please install unzip and vim in Cygwin. (Run setup-x86\_64.exe; when you get to the package installation screen, enter unzip in the search box, select all the packages listed; do the same for vim; then install.

unzipcommand – extracts files from a zip archive.

* A zip archive is a set of files stored in a single file.
* The files are *compressed* to reduce the archive's file size.

|  |  |
| --- | --- |
| -l | List the names of the files in the zip archive. |
| -p | Extract the contents of all the files to stdout. |
| -c | Same as –p but includes file names in the output stream. |
| -a | Convert text files. (E.g., changes line endings \r\n or \n to appropriate values.) |

filecommand – determines the type of data in a file.

* Unix does not mandate or require file extensions, such as .exe, .txt, or .docx
* file's default output is a string of information, which varies widely based on the file. For example:
  + XML 1.0 document, ASCII text, with very long lines, with CRLF line terminators
  + Microsoft Word 2007+

|  |  |
| --- | --- |
| -i | Output mime type strings (both mine-type and type of encoding), e.g., application/xml; charset=us-ascii |
| --mime-type | Output only the mime type, e.g., application/xml |
| --mime-encoding | Output only the type of encoding, e.g., charset=us-ascii, or charset=binary |
| -F *string* | Use the specified string separator as the separator between the filename and the file result returned. Defaults to ‘:’. |

iconv -f encoding -t encoding command – convert the encoding of a file. –f is "from", -t is "to"

|  |  |
| --- | --- |
| -l | List the encodings that are supported |
| -f | The current encoding of the file |
| -t | The target encoding. |

**Problem:** Search every file in a file hierarchy and find every email address stored in the files.

**A solution step by step:**

Step 1: Get a list of all the file names from a file hierarch you want to search; use the find command.

#!/bin/bash

IFS=$'\n' # The separator between file names will be \n

for n in $(find . –type f); do

printf "$n\n"

done

exit 0 # normal exit with no errors

Step 2: Determine the type of data in a file, which will determine how to search it; use the file command.

IFS=$'\n' # The separator between file names will be \n

for n in $(find . -type f); do

printf "$n\n"

t=$(file "$n" | sed -r "s/.+: //")

printf "$t\n"

done

exit 0 # normal exit with no errors

Step 3: Create appropriate commands to search specific types of files; use bash if statements.

#!/bin/bash

IFS=$'\n' # The separator between file names is \n

shopt -s nocasematch # make pattern matching ignore case

emailPattern="\w+(\.\w\*)\*@\w+(\.\w+)\*"

for n in $(find . -type f); do

# printf "$n\n"

t=$(file "$n" | sed -r "s/.+: //")

# printf "$t\n"

if [[ $t =~ "Microsoft" ]]; then

printf "From: $n\n"

unzip -p "$n" | grep -Eo $emailPattern

elif [[ $t =~ "ASCII" ]]; then

printf "From: $n\n"

grep -Eo $emailPattern "$n"

else

printf "Skipping file $n which has type $t\n"

fi

done

exit 0 # normal exit with no errors

Step 4: Save all of the email addresses that were found to a file for later processing

#!/bin/bash

IFS=$'\n' # The separator between file names is \n

shopt -s nocasematch # make pattern matching ignore case

emailPattern="\w+(\.\w\*)\*@\w+(\.\w+)\*"

> all\_emails.txt # Create an empty file to store the results

for n in $(find . -type f); do

# printf "$n\n"

t=$(file "$n" | sed -r "s/.+: //")

# printf "$t\n"

if [[ $t =~ "Microsoft" ]]; then

printf "From: $n\n"

unzip -p "$n" | grep -Eo $emailPattern >> all\_emails.txt

elif [[ $t =~ "ASCII" ]]; then

printf "From: $n\n"

grep -Eo $emailPattern "$n" >> all\_emails.txt

else

printf "Skipping file $n which has type $t\n"

fi

done

exit 0 # normal exit with no errors

Step 5: Optional: Sort the email addresses, combine duplicates and save to a file.

cat all\_emails.txt | sort | uniq –c >> final\_email\_list.txt