



Write a simple bash script to find a regex value for a credit card in a folder



Here's a simple Bash script that searches for credit card numbers (based on common patterns for Visa, MasterCard, American Express, and Discover) within all the files of a given folder. The script uses `grep` with a regular expression pattern to match the credit card numbers.



```
bash Copy code

#!/bin/bash

# Define the folder where you want to search for credit card numbers
SEARCH_FOLDER="/path/to/your/folder"

# Define the regular expression pattern for credit card numbers
CREDIT_CARD_PATTERN='^(4[0-9]{12}(?:[0-9]{3})?[5[1-5][0-9]{14}|6(?:011|5[0-9]{2}|4[0-9]{3})|3[47][0-9]{14})$'

# Search for files containing credit card numbers based on the regex pattern
find "$SEARCH_FOLDER" -type f -exec grep -lE "${CREDIT_CARD_PATTERN}" {} +
```

To use this script:

1. Replace `/path/to/your/folder` with the path to the folder you want to search.
2. Save the script as a file, e.g., `find_credit_cards.sh`.
3. Make the script executable by running `chmod +x find_credit_cards.sh`.
4. Run the script with `./find_credit_cards.sh`.

Keep in mind that this script may produce false positives or negatives, as the regular expression pattern is based on common credit card formats and does not account for all possible variations. Moreover, this script does not perform any validation on the credit card numbers it finds.



Can you write a simple, easy to understand, bash script to look for pdf files in a raw hard disk image. Just using the file extension is fine



Here's a simple Bash script to search for PDF files in a raw hard disk image by looking for filenames with the .pdf extension. The script uses `binwalk` to extract the file system and then `find` to search for PDF files within the extracted file system.



```
bash Copy code

#!/bin/bash

# Replace with the path to your raw hard disk image
RAW_IMAGE_PATH="/path/to/your/raw_image.img"

# Set the output folder where the extracted file system will be stored
EXTRACTION_OUTPUT_FOLDER="./extracted_file_system"

# Extract the file system using binwalk
binwalk -e -C "$EXTRACTION_OUTPUT_FOLDER" "$RAW_IMAGE_PATH"

# Find and list PDF files in the extracted file system
find "$EXTRACTION_OUTPUT_FOLDER" -iname "*.pdf"
```

To use this script:

1. Make sure you have `binwalk` installed. If not, you can install it using:
  - Debian-based systems: `sudo apt-get install binwalk`
  - Red Hat-based systems: `sudo yum install binwalk`
  - Arch-based systems: `sudo pacman -S binwalk`
2. Replace `/path/to/your/raw_image.img` with the path to the raw hard disk image you want to search.
3. Save the script as a file, e.g., `find_pdf_files.sh`.
4. Make the script executable by running `chmod +x find_pdf_files.sh`.
5. Run the script with `./find_pdf_files.sh`.

This script will list the paths of any found PDF files within the extracted file system. Note that this script assumes the file system within the raw hard disk image can be handled by `binwalk`.



Can you write a simple, easy to understand, script to look through a folder of text files with various file types extensions for a regex for a 16 digit number



Here's a simple Bash script that searches for 16-digit numbers in all text files within a specified folder, regardless of their file extensions. The script uses `grep` with a regular expression pattern to match the 16-digit numbers.



```
bash Copy code

#!/bin/bash

# Define the folder where you want to search for 16-digit numbers
SEARCH_FOLDER="/path/to/your/folder"

# Define the regular expression pattern for 16-digit numbers
NUMBER_PATTERN='[0-9]{16}'

# Search for files containing 16-digit numbers based on the regex pattern
find "$SEARCH_FOLDER" -type f -exec grep -lE "${NUMBER_PATTERN}" {} +
```

To use this script:

1. Replace `/path/to/your/folder` with the path to the folder you want to search.
2. Save the script as a file, e.g., `find_16_digit_numbers.sh`.
3. Make the script executable by running `chmod +x find_16_digit_numbers.sh`.
4. Run the script with `./find_16_digit_numbers.sh`.

The script will list the file paths of any files containing 16-digit numbers. Note that the script does not specifically filter for text files, but it assumes that the files in the folder are plain text, as binary files may produce false positives.



Can you update the script to validate the credit card numbers found to adhere to credit card number validation for Mastercard, Visa and American Express?



Here's an updated Bash script that searches for credit card numbers in all text files within a specified folder, and then validates them for Visa, MasterCard, and American Express using the Luhn algorithm. The script combines `grep` and `awk` to match credit card numbers and validate them.



```
bash Copy code

#!/bin/bash

# Define the folder where you want to search for credit card numbers
SEARCH_FOLDER="/path/to/your/folder"

# Define the regular expression pattern for Visa, MasterCard, and American Express
CREDIT_CARD_PATTERN='^(4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|3[47][0-9]{14})$'

# Search for files containing credit card numbers and validate them using the Luhn algorithm
find "$SEARCH_FOLDER" -type f -exec grep -oE "${CREDIT_CARD_PATTERN}" {} + | \
function luhn_check(number) {
    sum = 0
    alt = 1
    for (i = length(number); i > 0; i--) {
        digit = substr(number, i, 1)
        if (alt) {
            sum += digit
        } else {
            doubled = digit * 2
            sum += (doubled % 10) + int(doubled / 10)
        }
        alt = !alt
    }
    return sum % 10 == 0
}

{
    if (luhn_check($0)) {
        print $0
    }
}'
```

To use this script:

1. Replace `/path/to/your/folder` with the path to the folder you want to search.
2. Save the script as a file, e.g., `validate_credit_cards.sh`.
3. Make the script executable by running `chmod +x validate_credit_cards.sh`.
4. Run the script with `./validate_credit_cards.sh`.

The script will print valid Visa, MasterCard, and American Express credit card numbers found in the text files. Note that the script does not specifically filter for text files, but it assumes that the files in the folder are plain text, as binary files may produce false positives.