* Assignment 1 *

* Introduction:

Brief history of shell scripting.

1) Early Days (1960 - 1970)
Developed in 1960s at AT&T's Bell Labs by Ken Thompson. Dennis Ritchie & others. The term 'shell' refers to the user interface for accessing an operating system's servies.

2) Thomson shell (sh) by Ken Thompson.
It was simple & allowed users to execute commands, run programs, and perform basic input / output redirection.

3) Bourne shell (sh): by Stephen Bourne in 1979
Become standard shell in Unix systems as it introduced control flow constructs, command substitution & variables.

4) Korn shell (ksh): by David Korn
Is considered superset of bourne shell and combined features of C shell & Bourne shell.

5) C shell (csh): by Bill Joy at U.C Berkeley.
It has syntax resembling the C programming language & introduced features like

aliases, history substitution & job control

6) Bourne Again Shell (Bash): by Brian Fox
Bash is an open source, backward compatible replacement for Bourne Shell one of the mostly widely used shells today

7) Z Shell (Zsh): by Paul Falstad
It is known for its advanced features, customization options & users friendely enhancements.

8) Fish Shell (fish):
Friendly interactive shell designed for user friendliness, autosuggestions, syntax highlighting, and simplified scripting.

* Basic shell commands
1) ls : lists files & directories in the current directory
2) cd : changes current directory to specified directory.
3) pwd: prints current working directory's absolute path.
4) cp : copies files or directories from one location to another.
5) mv : moves or renames files or directories
6) rm : removes (deletes) files or directories
7) touch: creates an empty file or updates

8) mkdir
9) cat
10) echo

* Task

1) Conve

```
#!/b
if [
echo
exit
fi
ass
outp
{
whil
echo
echo
echo
echo
ech
ech
don
}
```

timestamp of an existing file.

8) mkdir : create a new directory.
9) cat : concatenation & displaying the content of files.
10) echo : outputs specified text or variables to the terminals or file.

* Tasks

1) convert a csv file to vcf format.

```
#!/bin/bash
if [$# -ne 2 ]; then
echo "Usage: $0 ass.csv output.vcf"
exit 1
fi
ass_csv ="$1"
output_vcf ="$2"
{
while IFS=, read -r name phone email; do
echo "BEGIN:VCARD"
echo "VERSION: 3.0"
echo "FN: $ name"
echo "TEL; TYPE = CELL : $phone"
echo "EMAIL; TYPE = INTERNET : $email"
echo "END : VCARD"
done < "$ass_csv"
} > "$output_vcf"
```

COEP TECHNOLOGICAL UNIVERSITY, PU

Wellesly Road, Shivajinagar, Pune - 411 005

2) Convert a youtube transcript to SRT format

```
#!/bin/bash
if [ $# -ne 2 ] ; then
echo "Usage: $0 ass2.txt output.srt"
exit 1
fi
ass2_txt = "$1"
output_srt = "$2"
{
counter = 1
while IFS = read -r line ; do
echo "$counter"
echo "$line"
echo " "
counter = $((counter +1))
done < "$ass2_txt"
} > "$output_srt"
```

3) Find top 10 size files created in last 20 days

```
#!/bin/bash
if [ $# -ne 1]; then
echo "Usage: $0 directory"
exit 1
fi
~~fi~~ directory = "$1"
find "$directory" -type f -ctime -20 - exec
ls -lh {} + | sort -k 5 -hr | head -n 10
```

CO

4) Move a
folder

```
#!/bin
if [ $
echo "
exit 1
fi
source
target
if [ !
mkdir
fi
find
{} +
hash
do
mv
done
```

* Con
→ Su
→ Con
~~num~~
ass
→ Ide
→ Re

4. move all duplicates files (except one) from a folder to a target location

```
#!/bin/bash
if [ $# -ne 2 ]; then
echo "Usage: $0 source_directory target_directory
exit 1
fi
source_directory = "$1"
target_directory = "$2"
if [ ! -d "$target_directory" ]; then
mkdir -p "$target_directory"
fi
find "$source_directory" -type f -exec md5sum
{} + | sort | uniq -w32 -dD |while read -r
hash file;
do
mv "$file" "$target_directory"
done
```

* Conclusion:
→ Successfully converted csv file into vcf file.
→ Converted a YT transcript to SRT format by numbering each seg, formating timestamps & assigning corr.
→ Identified the top 10 largest files
→ Removed all duplicate File.

[illegible]
27/03/24

COEP TECHNOLOGICAL UNIVERSITY, PUNE

Wellesly Road, Shivajinagar, Pune - 411 005

* Assignment 2 *

Tasks

1) Write shell scripts to find out the factorial of any number. (Take numbers from user)

```
#!/bin/bash

#Function to calculate factorial
factorial () {
      local num = $1
      local fact = 1
      for (( i=1 ; i<=num; i++)) ; do
            fact = $(( fact * i))
      done
      echo $fact
}

# Read input from user
read -p "Enter a number:" number

#Check if input is a positive integer
if [[ $number =~ ^[0-9]+$ ]] && [ $number
-ge 0 ]; then
      result = $(factorial $number)
      echo "Factorial of $number is result"
else
   echo "Please enter a positive integer."
fi
```

B) Write shell scripts to find out the fibonacci series of any number.

```
#!/bin/bash

# Function to generate Fibonacci series
fibonacci() {
        local n=$1
        local a=0
        local b=1
        echo "Fibonacci series upto $n terms:"
        for ((i=0; i<n; i++)); do
                echo -n "$a"
                local temp=$a
                a=$b
                b=$((temp + b))
        done
        echo
}

# Read number of terms from user
read -p "Enter the number of terms for
Fibonacci series: " terms

# Check if input is a positive integer
if [[ $terms =~ ^[0-9]+$ ]] && [ $terms
-ge 1 ]; then
        fibonacci $terms
else
        echo "Please enter a positive integer"
fi
```

COEP TECHNOLOGICAL UNIVERSITY, PUNE

Wellesly Road, Shivajinagar, Pune - 411 005

3) Write shell scripts to find even & odd numbers between some range of number. Take a minimum & maximum range of numbers from the user.

```
#!/bin/bash

#Function to find even & odd numbers
find_even_odd(){
        local min = $1
        local max = $2
        echo "Even numbers between $min & $max:"
        for ((num=min ; num<=max ; num++)); do
                if ((num %2 == 0 )); then
                        echo -n "$num"

                fi
        done
        echo

        echo "odd numbers between $min & $max:"
        for ((num = min; num<=max ; num++)); do
                ~~echo -n "$num"~~
                if ((num % 2 != 0 )); then
                        echo -n "$num"

                fi
        done
        echo
}

#Read range from user
read -p "Enter minimum number: " min
read -p "Enter maximum number :" max

#check if inputs are valid integers &
min is less than max
if [[ $min =~ ^[0-9] + $]] && [[ $max
=~ ^[0-9] + $ ]] && [$min -le $max]; then
        find_even_odd $min $max
else
        echo "Please enter valid integer with
minimum less than or equal to maximum"
fi
```

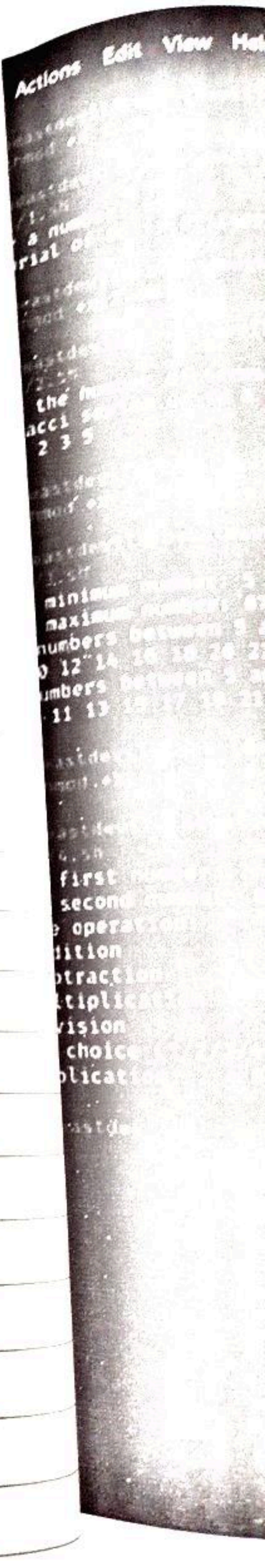
4) Write shell scripts for all arithmetic operations (addition, substraction, multiplication, division) by using the switch - case.

```
#!/bin/bash

#Read first number
read -p "Enter first number : "num1

#Read second number
read -p "Enter second number : "num2

#Read operation choice
echo "Choose operation:"
echo " 1. Addition"
echo " 2. Subtraction"
echo " 3. Multiplication"
echo "4. Division"
read -p "Enter choice (1/2/3/4): " choice

# Perform the chosen operation
case $ choice in
    1)
        result = $ (( num1 + num2))
        echo "Addition: $num1 + $num2 = $result"
        ;;
    2)
        result = $ ((num1 - num2))
        echo "Subtraction: $num1 - $num2 = $result"
        ;;
    3)
        result = ~~#~~$ (( num1 * num2))
        echo "Multiplication : $num1 * $num2 = $result"
        ;;
    4)
        # Handle division & division by zero
        if [ $num2 -ne 0 ]; then
            result = $((num1/num2))
            echo "Division : $num1 / $num2 = $result"
        else
            echo "Division by zero is not allowed."
        fi
        ;;
    *)
        echo "Invalid choice"
        ;;
esac
```

* conclusion

Covered essential shell scripting technique for finding factorial, fibonacci series, even odd & did basic arithmetic operations.

[illegible] 27/08/24



COEP TECHNOLOGICAL UNIVERSITY, PUNE

Wellesly Road, Shivajinagar, Pune - 411 005

* Assignment 3 *

1) Shell Length

Shell commands To get the length of a string

```
string = "Hello, World!"
echo ${#string}
```

2) substring Extraction

Shell commands To extract a substring:

```
string = "Hello, World!"
echo ${string:7:5}   # Extracts "World"
```

3) String Replacement

Shell Commands To replace the first occurence of a string:

```
String = "Hello, World!"
echo ${string/World/Bash}  # Replaces "World"
                             with "Bash"
```

Shell Commands To replace all occurences:

```
string = "Hello, World! Welcome to the World!"
echo ${string//World / Bash}  # Replaces all occurrences
                                of World" with "Bash"
```

4) Convert to Uppercase

To convert a string to uppercase:

```
String = "Hello, World!"
echo $(echo "$string" | tr '[:lower:]'
'[:upper:]')
```

5) convert to Lowercase

Shell commands To convert a string to lowercase:

```
string = "Hello, World!"
echo $(echo "$string" | tr [:upper:]'
'[:lower:]')
```

+ Conclusion

Learned string manipulation " in shell scrip by extracting, case changing. String length finding etc.

[illegible]
27/08/24

COEP TECHNOLOGICAL UNIVERSITY, PUNE

Wellesly Road, Shivajinagar, Pune - 411 005

* Assignment 4 *

* Write Shell scripting for below tasks:

1) Trim whitespace

```
#!/bin/bash

#Define the string with leading & trailing
whitespace
string = "      Hello, World!    "

#Trim whitespace using xargs
trimmed = $(echo "$string" | xargs)

#Output & trimmed string
echo "$trimmed"
```

2) Check if string contains a substring

```
#!/bin/bash

#Define the string & the substring
string = "Hello, World!"
substring = "World"

#Check if the string contains the substring
if [[ $string == *"substring"* ]]; then
    echo "string contains '$substring' "
else
```

```
    echo "string does not contains
    $substring"
fi
```

3) Split String by Delimiter

```
#!/bin/bash

#Define the string with coma as the delimeter
string = "apple, banana, cherry"

#Set the Internal Field Separator (IFS)
to commo
IFS = ','

#Read the string into an array
read -r -a array <<< "$string"

#Iterate over the array & print each
element
for element in "${array[@]}"; do
    echo "$element"
done
```

4) Compare Two Strings

```
#!/bin/bash

# Define the two strings
string1 = "Hello"
```

```
string2 = "World"

#compare the two strings
if [ "$string1" = "string2" ]; then
    echo "strings are equal"
else
    echo "strings are not equal"
fi
```

5) Get the character

```
#!/bin/bash

#Define the string
string = "Hello, World!"

# Get the character at index 7 ( 0 based indexing)
character = ${string:7:1}

#Print the character
echo "$character"
```

* Conclusion

Learned & do some task like trim whitespace, checking substring in string, splitting by delimeter, comparing two string & extract character etc.

[illegible]
27/08/25