B.M.S. College of Engineering

P.O. Box No.: 1908 Bull Temple Road,

Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

UNIX System Programming

23IS3AEUSP

Project Report

On

CURRENCY CONVERTER

Submitted By:

Adi Narayan Prasad G, 1BM23IS007
Arya Y P, 1BM23IS039
Ashrith S Jain, 1BM23IS040

Submitted To:

Rashmi R

AY 2023-2024

B.M.S. College of Engineering

P.O. Box No.: 1908 Bull Temple Road,

Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

Certified that the Project has been successfully presented at **B.M.S College Of Engineering** by **Adi Narayan Prasad G, Arya Y P, Ashrith S Jain** bearing USN: **1BM23IS007, 1BM23IS039 and 1BM23IS040** in partial fulfilment of the requirements for the III Semester degree in **Bachelor of Engineering in Information Science & Engineering** of **Visvesvaraya Technological University, Belgaum** as a part of project for the Course– Unix System Programming Course Code – **23IS3AEUSP** during academic year **2023-2024**.

Faculty Name – Rashmi R

Designation – Assistant Professor

Department of ISE, BMSCE.

Table of Contents

Chapter No.	Chapters	Page No.
Abstract		1
1	**Introduction**	1
2	**Problem Statement**	2
3	**API's and Process Control used**	3
4	**Implementation**	4-5
5	**Result**	6
	References	7

Abstract

This project aims to develop a Unix-based currency converter using shell scripting and external APIs to convert one currency to another based on real-time exchange rates. By leveraging external APIs, the converter will dynamically fetch exchange rates, enabling users to make currency conversions directly from the command line. The project will be implemented in Bash, utilizing tools like curl for HTTP requests and jq for parsing JSON data. The system will be designed to handle user input, process the conversion, and output the result in a clear format.

Introduction

Currency conversion is a critical task in global business, travel, and online shopping, where users often need to know the current exchange rate between two currencies. While graphical user interface (GUI)-based tools are common for currency conversion, many power users prefer the simplicity and speed of command-line interfaces. A Unix-based system, renowned for its scripting capabilities, provides an excellent environment for building such a tool. This project aims to create a currency converter application that retrieves real-time exchange rates from an external API, and utilizes shell scripting to perform the necessary calculations.

The tool utilizes utilities native to Unix systems, such as **curl** for fetching live exchange rates and **jq** for processing and displaying data. The result is a fast, easy-to-use, and scalable currency converter that can be run directly in the terminal without the need for complex installations or graphical user interfaces (GUIs).

The Unix shell provides a flexible environment for text processing and system management, while external APIs offer up-to-date currency exchange data. The combination of these technologies will help users seamlessly convert currencies with minimal effort. The implementation will focus on simplicity, efficiency, and accuracy, making use of available resources within the Unix ecosystem.

Problem statement

Currency conversion is crucial for international transactions, travel, and online shopping. While graphical user interface (GUI)-based currency converters are widely available, they tend to be resource-intensive and require navigating through multiple screens. Unix-based users, especially developers and system administrators, often prefer lightweight, efficient command-line tools for automation and quick access. However, there is a gap in the market for a simple, real-time, and command-line-based currency converter that integrates seamlessly with external APIs to fetch up-to-date exchange rates.

The key issues this project aims to solve are:

- Lack of simple, efficient command-line tools for currency conversion on Unix-based systems.
- Existing solutions either lack live data or are complex to configure.
- Difficulty in handling errors like invalid currency codes or API connectivity issues in existing tools.
- Prompt users for input (amount, source currency, target currency) in a simple and intuitive manner giving an user-friendly experience.
- Ensure the tool works on all Unix-based systems, including Linux and macOS providing cross platform compatability.
- A need for real-time conversion with minimal setup and maximum accessibility for Unix power users.

This project will address these challenges by providing a lightweight, real-time currency converter that operates entirely in the Unix shell environment, fetching live data from an external API and displaying the results with ease and accuracy.

API’s and Process Control used

API:

To retrieve real-time currency exchange rates, the project will utilize an external API that provides up-to-date financial data. One commonly used API for such tasks is the "ExchangeRate-API" or "Fixer.io," which gives access to current exchange rates for multiple currencies. These APIs allow for easy integration with shell scripts by making HTTP requests and returning data in a structured format like JSON.

- ExchangeRate-API: This API offers free and paid versions, depending on the number of requests per month. It supports over 150 currencies and provides exchange rates in real-time.
- Fixer.io: This API also provides current exchange rates, with a straightforward API response in JSON format. It has a free tier with limited usage, and higher-tier plans offer additional functionality.

Process Control:

The project will make use of standard Unix shell scripting tools to handle user input, call external APIs, and process the response. The control flow of the script will be as follows:

1. User Input: The user provides the amount and the source and target currencies.
2. API Call: The script makes an HTTP request to the chosen API using curl or wget.
3. Data Parsing: The returned JSON data will be parsed using jq or awk to extract the required exchange rate.
4. Calculation: The script performs the currency conversion based on the extracted rate.
5. Output: The result is printed to the terminal.

Process control mechanisms in shell scripting such as conditional statements, loops, and functions will be employed to ensure smooth execution, handle potential errors (e.g., invalid input or API downtime), and provide useful feedback to the user.

Implementation

```
#!/bin/bash
# Function to fetch exchange rates and store in a local file
fetch_exchange_rates() {
    echo "Fetching latest exchange rates..."
    curl -s "https://api.exchangerate-api.com/v4/latest/USD" -o rates.json
    if [ $? -ne 0 ]; then
        echo "Error: Unable to fetch exchange rates. Check your internet
connection."
        exit 1
    fi
    echo "Exchange rates updated successfully."
}
# Function to convert currency from INR to another currency
convert_from_inr() {
    echo "Enter the amount in INR: "
    read amount

    if ! [[ "$amount" =~ ^[0-9]+(\.[0-9]+)?$ ]]; then
        echo "Error: Invalid amount. Please enter a numeric value."
        return
    fi

    echo "Enter the target currency (e.g., USD, EUR, GBP): "
    read target_currency

    # If the user enters INR as the target currency, no conversion is needed
    if [ "$target_currency" == "INR" ]; then
        echo "No conversion needed. The amount is already in INR: $amount
INR"
        return
    fi

    # Fetch the INR to USD rate first, then get the rate for the target currency
    inr_to_usd=$(grep -o "\"INR\":[0-9.]*" rates.json | awk -F: '{print $2}')

    if [ -z "$inr_to_usd" ]; then
        echo "Error: Unable to find INR to USD conversion rate."
```

```
        return
    fi

    # Extract exchange rate for the target currency (from USD)
    rate=$(grep -o "\"$target_currency\":[0-9.]*" rates.json | awk -F: '{print $2}')

    if [ -z "$rate" ]; then
        echo "Error: Invalid currency code or data unavailable."
        return
    fi

    # Perform the conversion (INR -> USD -> Target currency)
    converted_amount=$(echo "$amount / $inr_to_usd * $rate" | bc)
    echo "Converted Amount: $converted_amount $target_currency"
}

# Main menu function
main_menu() {
    while true; do
        echo "============================="
        echo "   Currency Converter Menu  "
        echo "============================="
        echo "1. Convert INR to another currency"
        echo "2. Update exchange rates"
        echo "3. Exit"
        echo "============================="
        echo -n "Enter your choice: "
        read choice

        case $choice in
            1)  if [ ! -f rates.json ]; then
                    echo "Exchange rates not found. Fetching now..."
                    fetch_exchange_rates
                fi
                convert_from_inr ;;
            2) fetch_exchange_rates ;;
            3) echo "Exiting Currency Converter. Goodbye!"
                exit 0 ;;
            *)
                echo "Invalid choice. Please try again."
```

```
            ;;
        esac
    done
}

# Start the script
main_menu
```

Result

```
adi@adi-VirtualBox:~/prog$ bash lab.sh
============================
   Currency Converter Menu
============================
1. Convert INR to another currency
2. Update exchange rates
3. Exit
============================
Enter your choice: 1
Enter the amount in INR:
5000
Enter the target currency (e.g., USD, EUR, GBP):
USD
Converted Amount: 58 USD
============================
   Currency Converter Menu
============================
1. Convert INR to another currency
2. Update exchange rates
3. Exit
============================
Enter your choice: 2
Fetching latest exchange rates...
Exchange rates updated successfully.
============================
   Currency Converter Menu
============================
1. Convert INR to another currency
2. Update exchange rates
3. Exit
============================
Enter your choice: 3
Exiting Currency Converter. Goodbye!
adi@adi-VirtualBox:~/prog$ flameshot gui
```

References

1. ExchangeRate-API: Provides free and paid versions of an API for accessing live currency exchange rates. https://www.exchangerate-api.com/

2. jq Manual: A lightweight and flexible command-line JSON processor used for parsing API responses. https://stedolan.github.io/jq/

3. curl: A tool for transferring data with URLs, used in Unix systems to interact with APIs. https://curl.se/

Font: Times New Roman

Headings font size: 18

Running font size: 14

Line spacing: 1.15