杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

Linux 系统及应用作业报告

作业 5: 使用 Shell 命令实现一种简单加密

学院 卓越学院

学号 23040447

姓名 陈文轩

专业 智能硬件与系统 (电子信息工程)

2025 年 5 月 11 日

1 解密密文与明文推导

1.1 问题分析

已知密文为：

- Yfqp nx hmjfu
- Bmjwj ymjwj nx f xmjqg ymjwj nx f bfd
- Xmtb rj ymj htij

加密方法为：每个英文字母替换为 ASCII 字母表中后移 n 个位置的字母，大小写区分，标点符号和空格不变。例如，$n = 1$ 时，A 变为 B，Z 变为 A。目标是确定 n 值并解密明文。

1.2 解密思路

以第一段密文“Yfap nx hmjfu”为例，假设其明文为常见英文单词，尝试推导 n。注意到密文中有“nx”，可能对应明文中的常见双字母词如“is”。若“nx”解密为“is”，则：- n 解密为 i：n（ASCII 110）向前 n 个位置为 i（ASCII 105），则 $110 - n = 105$，$n = 5$。- x 解密为 s：x（ASCII 120）向前 n 个位置为 s（ASCII 115），同样 $120 - n = 115$，$n = 5$。

1.3 解密结果

- n 值为 5。- 明文为：

- Talk is cheap
- Where there is a shell there is a way
- Show me the code

2 脚本编写

2.1

以下是实现加密和解密功能的 Shell 脚本代码：

```bash
#!/bin/bash

# Function to encrypt text
encrypt() {
    local text="$1"
    local shift="$2"
    local result=""
    for ((i = 0; i < ${#text}; i++)); do
        char="${text:i:1}"
        if [[ "$char" =~ [A-Z] ]]; then
            # Uppercase letters
            result+=$(printf "\\$(printf '%03o' $(( ( $(printf '%d' "'$char") - 65 + shift ) % 26 + 65 )))")
        elif [[ "$char" =~ [a-z] ]]; then
            # Lowercase letters
            result+=$(printf "\\$(printf '%03o' $(( ( $(printf '%d' "'$char") - 97 + shift ) % 26 + 97 )))")
        else
            # Non-alphabetic characters (unchanged)
            result+="$char"
        fi
    done
    echo "$result"
}

# Function to decrypt text
decrypt() {
    local text="$1"
    local shift="$2"
    local result=""
    for ((i = 0; i < ${#text}; i++)); do
        char="${text:i:1}"
        if [[ "$char" =~ [A-Z] ]]; then
            # Uppercase letters
```

```
            result+=$(printf "\\$(printf '%03o' $(( ( $(printf '%d' "'$char") - 65 - shift + 26 ) % 26 + 65 )))")
        elif [[ "$char" =~ [a-z] ]]; then
            # Lowercase letters
            result+=$(printf "\\$(printf '%03o' $(( ( $(printf '%d' "'$char") - 97 - shift + 26 ) % 26 + 97 )))")
        else
            # Non-alphabetic characters (unchanged)
            result+="$char"
        fi
    done
    echo "$result"
}

# Parse command-line arguments
while [[ $# -gt 0 ]]; do
    case "$1" in
    -s)
        input_string="$2"
        shift 2
        ;;
    -k)
        shift_value="$2"
        shift 2
        ;;
    -if)
        input_file="$2"
        shift 2
        ;;
    -of)
        output_file="$2"
        shift 2
```

```
        ;;
    -d)
        mode="decrypt"
        shift
        ;;
    *)
        echo "Invalid option: $1"
        exit 1
        ;;
    esac
done

# Validate shift value
if [[ -z "$shift_value" || ! "$shift_value" =~ ^[0-9]+$ ]];
   then
    echo "Error: Shift value (-k) must be a positive integer."
    exit 1
fi

# Perform encryption or decryption
if [[ -n "$input_string" ]]; then
    if [[ "$mode" == "decrypt" ]]; then
        result=$(decrypt "$input_string" "$shift_value")
    else
        result=$(encrypt "$input_string" "$shift_value")
    fi
    echo "$result"
elif [[ -n "$input_file" ]]; then
    if [[ ! -f "$input_file" ]]; then
        echo "Error: Input file not found."
        exit 1
    fi
    input_content=$(cat "$input_file")
    if [[ "$mode" == "decrypt" ]]; then
```

```
96          result=$(decrypt "$input_content" "$shift_value")
97      else
98          result=$(encrypt "$input_content" "$shift_value")
99      fi
100     if [[ -n "$output_file" ]]; then
101         echo "$result" >"$output_file"
102     else
103         echo "$result"
104     fi
105 else
106     echo "Error: No input provided. Use -s for string input or
        -if for file input."
107     exit 1
108 fi
```

Code Listing 1: 加密与解密脚本

2.2 代码解释

上述脚本实现了一个简单的加密和解密工具，基于凯撒密码（Caesar Cipher）的原理。以下是脚本的主要功能和实现细节：

- **加密函数（encrypt）：**
  - 输入一个字符串和一个位移值（shift）。
  - 遍历字符串中的每个字符：
    - 如果是大写字母（A-Z），通过公式 (ASCII 值 $-65+$ shift)%26 $+65$ 计算加密后的字符。
    - 如果是小写字母（a-z），通过公式 (ASCII 值 $-97+$ shift)%26 $+97$ 计算加密后的字符。
    - 非字母字符（如空格、标点符号）保持不变。
  - 最终返回加密后的字符串。
- **解密函数（decrypt）：**
  - 输入一个字符串和一个位移值（shift）。

  - 遍历字符串中的每个字符：
    - 如果是大写字母（A-Z），通过公式 (ASCII 值 $-65-$ shift $+26)\%26+65$ 计算解密后的字符。
    - 如果是小写字母（a-z），通过公式 (ASCII 值 $-97-$ shift $+26)\%26+97$ 计算解密后的字符。
    - 非字母字符保持不变。
  - 最终返回解密后的字符串。

- **命令行参数解析：**
  - 使用 `while` 循环和 `case` 语句解析命令行参数。
  - 支持的参数包括：
    - `-s`：指定输入字符串。
    - `-k`：指定加密/解密的位移值。
    - `-if`：指定输入文件。
    - `-of`：指定输出文件。
    - `-d`：指定解密模式（如果不加 `-d`，默认为加密模式）。

- **加密/解密逻辑：**
  - 如果提供了字符串输入（`-s`），直接调用加密或解密函数处理字符串。
  - 如果提供了文件输入（`-if`），读取文件内容并调用加密或解密函数处理。
  - 如果指定了输出文件（`-of`），将结果写入文件；否则直接输出到终端。

- **错误处理：**
  - 如果未提供必要的参数（如 `-s` 或 `-if`），脚本会提示错误并退出。
  - 如果位移值（`-k`）不是正整数，脚本会提示错误并退出。
  - 如果输入文件不存在，脚本会提示错误并退出。

2.3 脚本测试

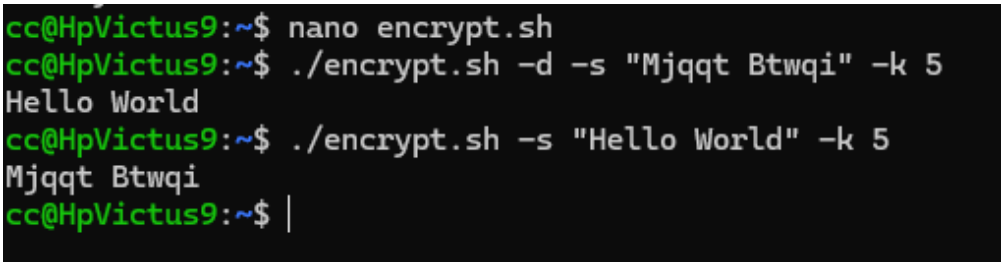

图 1 WSL 指令测试结果

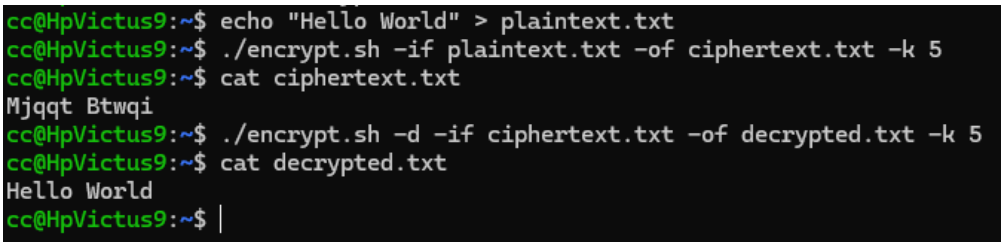

图 2 WSL 文件测试结果