МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Програмної інженерії

Звіт
з лабораторної роботи № 1
з дисципліни: «Скриптові мови програмування»
з теми: «Розробка Bash-скрипту для конвертації розкладу CIST NURE у формат Google Календар»

Виконав:
ст. гр. ПЗПІ-23-2
Ситник Є. С.

Перевірив:
ст. викл. каф. ПІ
Сокорчук І. П.

Харків – 2025

1 РОЗРОБКА BASH-СКРИПТУ ДЛЯ КОНВЕРТАЦІЇ РОЗКЛАДУ CIST NURE У ФОРМАТ GOOGLE КАЛЕНДАР

1.1 Мета роботи

- Ознайомитись з основними командами Bash для роботи з файлами та текстовими даними.
- Навчитись використовувати команди «select», «ls», «awk», «sort», «uniq», «tee», «cat», «sed», «iconv», тощо.
- Розробити Bash-скрипт для перетворення CSV файла розкладу занять у формат для імпорту в Google Календар.

1.2 Хід роботи

1.2.1 Огляд завдання

Завдання полягало у розробці скрипта для обробки експортованого з сайту cist.nure.ua розкладу занять у форматі CSV та перетворення його у формат, сумісний з Google Календарем. Скрипт повинен підтримувати різні параметри командного рядка, вибір файлів та груп студентів, а також забезпечувати належне форматування даних для успішного імпорту в Google Календар.

1.2.2 Реалізація

Реалізований Bash-скрипт складається з декількох основних функціональних блоків.

Спочатку реалізовано дві допоміжні функції. Функція «error()» відповідає за виведення повідомлень про помилки та завершення роботи скрипта із заданим кодом виходу. Вона приймає два параметри: текст помилки та код помилки, після чого перенаправляє текст помилки у потік помилок («stderr») за допомогою «>&2».

```
error() {
    echo "Error: $1." >&2;
    exit $2;
}
```

Функція «select_from()» забезпечує інтерактивний вибір елементів із списку через меню. Вона обробляє як випадок, коли у списку лише один елемент (автоматично його вибирає), так і випадок з декількома елементами, коли

користувачу надається меню вибору. Команда «select» дозволяє створити меню для вибору опцій, з яких користувач може вибрати потрібний варіант або вийти з вибору за допомогою опції «quit».

```
select_from(){
    if [ -n "$2" -a $(echo "$1" | wc -l) -eq 1 ]; then echo "$1"; return 0; fi

    select selected in quit $1; do
        case "$selected" in
            '') continue ;;
            'quit') return 1 ;;
            *) echo "$selected"; return 0 ;;
        esac
    done
}
```

Скрипт підтримує наступні ключі командного рядка: «–version» для виведення інформації про версію скрипта, «–help» для виведення довідки про використання скрипта, та «-q» або «–quiet» для вимкнення виведення даних в стандартний потік виведення. Це реалізовано через перевірку значення першого аргументу командного рядка.

```
if [ "$1" = "--version" ]; then echo 'cist converter 1.0'; exit 0; fi
if [ "$1" = "--help" ]; then
    echo "Usage: $0 [--help | --version] | [-q|--quiet] [[академ_група] файл_із_cist.csv]"
    echo 'Convert schedule.csv from CIST format into Google Calendar format'
    echo -e "  --help \t\tPrint this help message and exit"
    echo -e "  --version \t\tPrint version info and exit"
    echo -e "  -q, --quiet \t\tDo not print the result to stdout"
    echo -e "  академ_група \t\tOptional group name"
    echo -e "  файл_із_cist.csv \tCSV file to process"
    exit 0;
fi
```

Для підтримки опції «–quiet» використовується перенаправлення стандартного потоку виведення через файловий дескриптор 3. Команда «exec 3>&1» створює новий файловий дескриптор, що вказує на стандартний потік виведення, а «exec 3>/dev/null» перенаправляє його в «/dev/null» для режиму «–quiet», блокуючи виведення результату на екран.

```
exec 3>&1;
if [ "$1" = "-q" -o "$1" = "--quiet" ]; then exec 3>/dev/null; shift; fi
```

Якщо файл не вказано через аргументи командного рядка, скрипт шукає доступні CSV файли в поточній директорії за шаблоном «TimeTable_??_??_20??.csv» за допомогою команди «ls» та пропонує вибрати один з них,

використовуючи функцію «select_from». Після вибору файлу скрипт перевіряє його наявність та права доступу, використовуючи умовні оператори з флагами «-f» (існування файлу) та «-r» (можливість читання).

```
if [ -z "$csv" ]; then csv=$(select_from "$csvs") || exit 0; fi

if [ ! -f "$csv" ]; then error "file '$csv' is not exist" 1; fi
if [ ! -r "$csv" ]; then error "permission to read file '$csv' denied" 2; fi
```

Далі скрипт зчитує вміст файлу за допомогою команди «cat», заміняє символи повернення каретки на переноси рядків за допомогою команди «sed ’s/r/n/g’» та конвертує текст з Windows-1251 в UTF-8 за допомогою «iconv -f cp1251 -t utf8». Потім формується список доступних академічних груп за допомогою команди «awk», яка обробляє рядки файлу CSV з використанням змінної FPAT для коректного розбиття полів даних, розділених комами. Отриманий список сортується («sort») та з нього видаляються дублікати («uniq»).

```
csv_data=$( cat "$csv" | sed 's/\r/\n/g' | iconv -f cp1251 -t utf8 )

groups=$(
    echo "$csv_data" \
    | awk -v FPAT='[^,]*|"[^"]*"' '
        NR > 1 {
            gsub(/^"/, "", $1);
            if (split($1, res, " - ") != 1) print res[1]
        }
    ' \
    | sort \
    | uniq
)
```

Якщо група не вказана через аргументи командного рядка, скрипт пропонує вибрати групу зі списку за допомогою функції «select_from». Якщо група не знайдена у вибраному файлі, скрипт повідомляє про це користувача та пропонує вибрати іншу групу.

```
if [ -z "$group" ]; then group=$(select_from "$groups" "skip") || exit 0; fi
```

Основний процес перетворення даних здійснюється за допомогою команди «awk», яка обробляє CSV дані, фільтрує рядки відповідно до обраної групи, та форматує дані для Google Календаря. Всередині «awk» визначаються дві функції: «time_fmt()» для перетворення часу з формату «HH:MM:SS» у формат «HH:MM AM/PM», та «date_fmt()» для перетворення дати з формату «DD.MM.YYYY» у формат «MM/DD/YYYY». Обидві функції використовують «strftime()» та

«mktime()» для конвертації форматів. Для кожного рядка, що відповідає обраній групі, скрипт форматує поля «Subject», «Start Date», «Start Time», «End Date», «End Time» та «Description» відповідно до вимог Google Календаря. Результат зберігається у файл з префіксом «Google_» та виводиться на екран за допомогою команди «tee», якщо не вказано ключ «–quiet».

```
echo "$csv_data" \
| awk -v FPAT='[^,]*|"[^"]*"' -v pattern="$([ -n "$group" ] && echo "^\"$group - ")" '
    function time_fmt(time) {
        gsub(/:|"/, " ", time)
        return strftime("%I:%M %p", mktime("1970 01 01" time))
    }

    function date_fmt(date) {
        gsub(/"/, "", date)
        split(date, dp, ".")
        return strftime("%m/%d/%Y", mktime(dp[3] " " dp[2] " " dp[1] " 00 00 00"))
    }

    NR==1 { print "Subject,Start Date,Start Time,End Date,End Time,Description" }

    NR!=1 && $1 ~ pattern {
        gsub(pattern "|^\"|\"$", "", $1)

        printf("\"%s; №%d\",%s,%s,%s,%s,%s\n",
            $1, ++lessons[$2], date_fmt($2), time_fmt($3), date_fmt($4), time_fmt($5), $12)
    }
' \
| tee "Google_$(basename "$csv")" >&3
```

1.3 Висновки

У ході виконання лабораторної роботи було розроблено скрипт командного інтерпретатора Bash для перетворення формату CSV файлів розкладу занять із системи CIST ХНУРЕ у формат, придатний для імпорту в Google Календар.

Скрипт має гнучкий інтерфейс командного рядка з підтримкою різних опцій та аргументів, а також забезпечує належну обробку помилок. Він демонструє ефективне використання базових команд Bash для роботи з файлами та обробки текстових даних, включаючи «select», «ls», «awk», «sort», «uniq», «tee», «cat», «sed» та «iconv».

Основні навички, отримані при виконанні роботи:

- Використання перенаправлення потоків даних в Bash
- Створення інтерактивних меню за допомогою команди «select»
- Обробка структурованих текстових даних за допомогою «awk»
- Конвертація форматів дати та часу
- Робота з кодуваннями символів

Розроблений скрипт повністю відповідає поставленим вимогам та може бути використаний для автоматизації процесу імпорту розкладу занять із системи CIST у Google Календар.

ДОДАТОК А

Повний текст розробленого скрипта

```
#!/bin/bash

error() {
    echo "Error: $1." >&2;
    exit $2;
}

select_from(){
    if [ -n "$2" -a $(echo "$1" | wc -l) -eq 1 ]; then echo "$1"; return 0; fi

    select selected in quit $1; do
        case "$selected" in
            '') continue ;;
            'quit') return 1 ;;
            *) echo "$selected"; return 0 ;;
        esac
    done
}

if [ "$1" = "--version" ]; then echo 'cist converter 1.0'; exit 0; fi
if [ "$1" = "--help" ]; then
    echo "Usage: $0 [--help | --version] | [-q|--quiet] [[академ_група] файл_із_cist.csv]"
    echo 'Convert schedule.csv from CIST format into Google Calendar format'
    echo -e "  --help \t\tPrint this help message and exit"
    echo -e "  --version \t\tPrint version info and exit"
    echo -e "  -q, --quiet \t\tDo not print the result to stdout"
    echo -e "  академ_група \t\tOptional group name"
    echo -e "  файл_із_cist.csv \tCSV file to process"
    exit 0;
fi

exec 3>&1;
if [ "$1" = "-q" -o "$1" = "--quiet" ]; then exec 3>/dev/null; shift; fi

if [ $# -eq 2 ]; then group="$1"; shift; fi
if [ $# -eq 1 ]; then csv="$1"; fi

csvs=$(ls TimeTable_??_??_20??.csv 2>/dev/null | sort)

if [ -z "$csv" ]; then csv=$(select_from "$csvs") || exit 0; fi

if [ ! -f "$csv" ]; then error "file '$csv' is not exist" 1; fi
if [ ! -r "$csv" ]; then error "permission to read file '$csv' denied" 2; fi

csv_data=$( cat "$csv" | sed 's/\r/\n/g' | iconv -f cp1251 -t utf8 )

groups=$(
```

```
    echo "$csv_data" \
    | awk -v FPAT='[^,]*|"[^"]*"' '
        NR > 1 {
            gsub(/^"/, "", $1);
            if (split($1, res, " - ") != 1) print res[1]
        }
    ' \
    | sort \
    | uniq
)

if [ -z "$group" ]; then group=$(select_from "$groups" "skip") || exit 0; fi

if ! echo "$groups" | grep "$group"; then
    echo "No '$group' group in '$csv'."
    group=$(select_from "$groups" "skip") || exit 0;
fi

echo "$csv_data" \
| awk -v FPAT='[^,]*|"[^"]*"' -v pattern="$([ -n "$group" ] && echo "^\"$group - ")" '
    function time_fmt(time) {
        gsub(/:|"/, " ", time)
        return strftime("%I:%M %p", mktime("1970 01 01" time))
    }

    function date_fmt(date) {
        gsub(/"/, "", date)
        split(date, dp, ".")
        return strftime("%m/%d/%Y", mktime(dp[3] " " dp[2] " " dp[1] " 00 00 00"))
    }

    NR==1 { print "Subject,Start Date,Start Time,End Date,End Time,Description" }

    NR!=1 && $1 ~ pattern {
        gsub(pattern "|^\"|\"$", "", $1)

        printf("\"%s; №%d\",%s,%s,%s,%s,%s\n",
            $1, ++lessons[$2], date_fmt($2), time_fmt($3), date_fmt($4), time_fmt($5), $12)
    }
' \
| tee "Google_$(basename "$csv")" >&3
```