МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з лабораторної роботи № 1 з дисципліни

скриптові мови програмування

Виконала: Перевірив:

ст. гр. ПЗПІ-23-5 ст. викладач кафедри ПІ

Самосватова А. Р. Сокорчук Ігор Петрович

Харків 2025

ЗАВДАННЯ

Ознайомитися з основними командами Bash для роботи з файлами та текстовими даними, навчитися використовувати команди select, ls, awk, sort, uniq, tee, cat, sed, iconv тощо. Розробити Bash-скрипт для перетворення CSV-файла розкладу занять, експортованого з сайту cist.nure.ua, у формат, придатний для імпорту в Google Календар. Скрипт повинен вміти працювати з вибраним користувачем або автоматично визначеним CSV-файлом розкладу, сортувати та фільтрувати дані, а також форматувати їх у відповідності до вимог Google Календаря.

У результаті має створюватися файл Google\_TimeTable\_??\_??\_20??.csv з відформатованими даними. Скрипт має обробляти помилки (відсутність файлу, групи, проблеми з доступом або обробкою), бути виконуваним для всіх користувачів, але редагованим лише власником.

ХІД РОБОТИ

Створюємо новий скрипт з назвою pzpi-23-5-samosvatova-anastasiia-lab1,

використовуючи команду touch та відкриваємо файл у текстовому редакторі nano. Оголошуємо інтерпретатор, який вказує що скрипт виконується через Bash. Оголошуэмо змінні.

1. #!/bin/bash

2.

3. version="1.0"

4. quiet=false

5. group=""

6. file=""

Далі у скрипті визначається функція showHelp, яка призначена для виведення короткої інструкції щодо використання скрипта. У тілі функції за допомогою команд echo виводиться правильний синтаксис виклику скрипта, а також доступні параметри, такі як --help для виведення довідки, --version для відображення версії скрипта та -q або --quiet для активації тихого режиму.

8. showHelp() {

9. echo "Використання: $0 [опції] [група] [файл]"

10. echo "Опції:"

11. echo " --help Показати це повідомлення і вийти"

12. echo " --version Показати інформацію про версію і вийти"

13. echo " -q, --quiet Приховати стандартний вивід"

14. }

Після визначення функції showHelp, скрипт переходить до обробки параметрів командного рядка. Це відбувається в циклі while [[ "$#" -gt 0 ]], який проходиться по всіх переданих аргументах. Усередині циклу використовується оператор case, що перевіряє значення кожного аргументу. Якщо аргумент дорівнює --help, викликається функція showHelp, після чого скрипт завершують свою роботу (exit 0). Якщо аргумент — --version, то виводиться версія скрипта і також відбувається вихід. Для всіх інших аргументів передбачено окрему обробку. Якщо аргумент не починається з -, не є ще визначеним file і закінчується на .csv, то він записується у змінну file як шлях до CSV-файлу. Якщо аргумент не є опцією і ще не встановлена змінна group, тоді цей аргумент розглядається як назва академічної групи. Якщо жодна з умов не спрацювала, виводиться повідомлення про невідому опцію, і скрипт завершують з помилкою (exit 1).

16. while [[ "$#" -gt 0 ]]; do

17. case $1 in

18. --help)

19. showHelp

20. exit 0

21. ;;

22. --version)

23. echo "$0 версія $version"

24. exit 0

25. ;;

26. -q|--quiet)

27. quiet=true

28. ;;

29. \*)

30. if [[ -z "$file" && "$1" != -\* && "$1" == \*.csv ]]; then

31. file="$1"

32. elif [[ -z "$group" && "$1" != -\* ]]; then

33. group="$1"

34. else

35. echo "Невідома опція: $1" >&2

36. exit 1

37. fi

38. ;;

39. esac

40. shift

41. done

Після обробки параметрів перевіряється, чи було задано файл. Якщо змінна file порожня, скрипт виводить повідомлення “Оберіть CSV-файл:” і за допомогою конструкції select формує меню з файлів, що відповідають шаблону TimeTable\_??\_??\_20??.csv. Файли відсортовані за іменем. Якщо користувач обрав файл (тобто змінна file більше не пуста), скрипт переходить до наступного етапу. Якщо обрано невірно — повідомляє про помилку і просить спробувати ще раз.

43. if [[ -z "$file" ]]; then

44. echo "Оберіть CSV-файл:"

45. select file in $(ls TimeTable\_??\_??\_20??.csv 2>/dev/null | sort); do

46. [[ -n "$file" ]] && break

47. echo "Невірний вибір. Спробуйте ще раз."

48. done

49. fi

Далі відбувається перевірка доступності обраного або вказаного CSV-файлу. За допомогою конструкції [[ ! -r "$file" ]] перевіряється, чи існує файл і чи доступний він для читання. Якщо файл відсутній або недоступний, скрипт виводить повідомлення про помилку і завершує роботу.

51. if [[ ! -r "$file" ]]; then

52. echo "Помилка: файл '$file' не знайдено або він недоступний." >&2

53. exit 1

54. fi

Після перевірки файл переходить до попередньої обробки. Спочатку за допомогою регулярного виразу із команди grep із назви файлу витягується дата у форматі DD\_MM\_YYYY — ця дата буде використана в імені вихідного файлу. Потім виконується конвертація вмісту CSV-файлу з кодування Windows-1251 у UTF-8 за допомогою утиліти iconv, а також замінюються символи \r на \n за допомогою sed. Результат записується у тимчасовий файл temp\_utf8.csv, який надалі використовується як основний файл. Якщо конвертація не вдається — скрипт виводить помилку і завершується.

56. dateSuffix=$(basename "$file" | grep -o "[0-9]\{2\}\_[0-9]\{2\}\_20[0-9]\{2\}")

57. utf8File="temp\_utf8.csv"

58.

59. sed 's/\r/\n/g' "$file" | iconv -f cp1251 -t UTF-8 > "$utf8File" || {

60. echo "Помилка: не вдалося конвертувати файл '$file' у UTF-8." >&2

61. exit 1

62. }

63.

64. file="$utf8File"

На наступному етапі перевіряється, чи була вказана група. Якщо ні — скрипт аналізує CSV-файл і за допомогою команди awk виділяє всі унікальні значення назв груп із першої колонки (назви предметів). Значення сортуються і заносяться у масив groups. Якщо у масиві лише один елемент, то ця група автоматично призначається як активна. Якщо ж груп декілька, користувачу пропонується обрати одну з них у вигляді меню через select.

66. if [[ -z "$group" ]]; then

67. mapfile -t groups < <(awk -F',' 'NR > 1 { split($1, a, " "); gsub(/"/, "", a[1]); print a[1] }' "$file" | sort | uniq)

68. if [[ ${#groups[@]} -eq 1 ]]; then

69. group="${groups[0]}"

70. echo "Єдина доступна група: $group"

71. else

72. echo "Оберіть академічну групу:"

73. select group in "${groups[@]}"; do

74. [[ -n "$group" ]] && break

75. echo "Невірний вибір. Спробуйте ще раз."

76. done

77. fi

78. fi

Після вибору групи виводиться її назва (якщо не увімкнено тихий режим) і формується ім’я вихідного файлу у вигляді Google\_TimeTable\_дата.csv, де дата — раніше витягнута частина з імені вхідного файлу.

80. $quiet || echo "Ви вибрали: $group"

82. outputFile="Google\_TimeTable\_${dateSuffix}.csv"

Основна обробка даних виконується у кілька етапів. Спочатку команда awk фільтрує рядки CSV-файлу відповідно до вказаної групи. З кожного рядка витягується інформація про предмет, дату та час початку і завершення заняття, а також опис. Деякі поля очищуються від лапок і символів ;, замінюються роздільники на ,. Результат записується у сформований CSV-файл.

84. awk -F',' -v grp="$group " 'BEGIN { OFS = ","; }

85. NR > 1 && $1 ~ grp {

86. gsub(/;/, ",", $0)

87. gsub(/"/, "", $1); gsub(/"/, "", $2); gsub(/"/, "", $3); gsub(/"/, "", $5); gsub(/"/, "", $12)

88. for (i = 2; i <= NF; i++) {

89. gsub(/"/, "", $i)

90. if ($i ~ /^[0-9]{2}\.[0-9]{2}\.[0-9]{4}$/) {

91. startDate = "\"" $i "\""

92. startTime = "\"" $(i+1) "\""

93. endDate = "\"" $i "\""

94. endTime = "\"" $(i+3) "\""

95. description = $(i+10)

96. break

97. }

98. }

99. subject = "\"" $1 "\""

100. gsub(/"/, "", description)

101. description = "\"" $(i+10) "\""

102. print subject, startDate, startTime, endDate, endTime, description

103. }' "$file" > "$outputFile"

Після цього за допомогою іншої команди awk дати, записані у форматі DD.MM.YYYY, перетворюються у формат MM/DD/YYYY, який приймає Google Календар. Ці дані записуються у тимчасовий файл, який потім замінює основний.

105. awk -F',' 'BEGIN {OFS=","}

106. NR > 0 {

107. split($2, date, ".");

108. $2 = date[2] "/" date[1] "/" date[3];

109. gsub(/"/, "", $2);

110. $2 = "\"" $2 "\"";

111.

112. split($4, date, ".");

113. $4 = date[2] "/" date[1] "/" date[3];

114. gsub(/"/, "", $4);

115. $4 = "\"" $4 "\"";

116.

117. print $1, $2, $3, $4, $5, $6;

118. }' "$outputFile" > "$outputFile.tmp" && mv "$outputFile.tmp" "$outputFile"

Після зміни формату дат скрипт виконує сортування записів у файлі по колонці дати та часу початку — це дозволяє впорядкувати заняття у хронологічному порядку, що дуже зручно при імпорті в календар.

120. sort -t',' -k2,2 -k3,3 "$outputFile" -o "$outputFile"

На завершальному етапі скрипт проводить нумерацію занять. За допомогою двох асоціативних масивів lectureNumber і labNumber відстежується кількість лекцій і лабораторних з кожного предмету. Якщо у полі опису зустрічається позначка Лб, скрипт вважає це лабораторною і додає нумерацію у форматі №N, де N — номер заняття. Якщо ж позначка відсутня — вважається, що це лекція, і їй також призначається відповідний номер.

125. awk -F',' -v grp="$group " 'BEGIN { OFS = ","; print "Subject,Start Date,Start Time,End Date,End Time,Description"

126. }

127. NR > 1 {

128. subjectName = $6

129.

130. if (subjectName ~ /Лб/) {

131. if (!(subjectName in labNumber)) labNumber[subjectName] = 1

132. number = int((labNumber[subjectName] + 1) / 2)

133. labNumber[subjectName]++

134. gsub(/"/, "", $1)

135. sub(/ПЗПІ-[0-9]+-[0-9]+ - /, "", $1)

136. $1 = "\"" $1 " №" number "\""

137. } else {

138. if (!(subjectName in lectureNumber)) lectureNumber[subjectName] = 0

139. lectureNumber[subjectName]++

140. gsub(/"/, "", $1)

141. sub(/ПЗПІ-[0-9]+-[0-9]+ - /, "", $1)

142. $1 = "\"" $1 " №" lectureNumber[subjectName] "\""

143. }

Крім того, час занять перетворюється з 24-годинного формату (наприклад, 14:30) у 12-годинний з вказанням AM або PM, що відповідає вимогам Google Календаря. Усі відформатовані значення записуються назад у файл.

145. gsub(/"/, "", $3)

146. split($3, time, ":")

147. hour = time[1]; minute = time[2]

148. if (hour == 0) { hour = 12; suffix = "AM" }

149. else if (hour == 12) { suffix = "PM" }

150. else if (hour > 12) { hour -= 12; suffix = "PM" }

151. else { suffix = "AM" }

152. $3 = "\"" hour ":" minute " " suffix "\""

153.

154. gsub(/"/, "", $5)

155. split($5, time, ":")

156. hour = time[1]; minute = time[2]

157. if (hour == 0) { hour = 12; suffix = "AM" }

158. else if (hour == 12) { suffix = "PM" }

159. else if (hour > 12) { hour -= 12; suffix = "PM" }

160. else { suffix = "AM" }

161. $5 = "\"" hour ":" minute " " suffix "\""

Якщо скрипт не був запущений у тихому режимі, його фінальні дії — це виведення вмісту згенерованого файлу у консоль за допомогою tee, а також повідомлення про успішне створення файлу для Google Календаря. В кінці скрипт завершують командою exit 0, що означає успішне виконання.

166. $quiet || tee /dev/tty < "$outputFile" > /dev/null

167. $quiet || echo "Файл '$outputFile' готовий для імпорту в Google Календар."

168.

169. exit 0

ВИСНОВОК

Розроблений Bash-скрипт дозволяє перетворювати CSV-файл розкладу занять з сайту cist.nure.ua у формат, придатний для імпорту в Google Календар. Скрипт автоматично обирає або дозволяє вибрати файл, конвертує його з Windows-1251 у UTF-8, очищує дані та форматує їх згідно вимог календаря. Він сортує записи за датою та часом, нумерує заняття, перетворює час у 12-годинний формат і генерує вихідний файл. Скрипт обробляє помилки, підтримує тихий режим і має налаштовані права доступу, що робить його безпечним та зручним у використанні.

ДОДАТОК А

Посилання на YouTube

Відеозапис презентації результатів лабораторної роботи: <https://youtu.be/X3rDSNStuBU>

00:00 - Привітання

00:27 - Призначення першого рядка

00:37 - Оголошення змінних

01:10 - Обробка параметрів командного рядка

02:53 - Перевірка існування файлу

03:16 - Конвертація файлу у UTF-8

03:35 - Вибір групи, якщо її не вказано

04:27 - Формування нового CSV для Google Calendar

05:42 - Нумерація занять

06:45 - Перетворення часу у 12-годинний формат

07:20 - Виведення результату

08:15 - Робота скрипта

ДОДАТОК Б

GitHub репозиторій: <https://github.com/NureSamosvatovaAnastasiia/smp-pzpi-23-5-samosvatova-anastasiia/blob/main/Lab1/smp-pzpi-23-5-samosvatova-anastasiia-lab1/smp-pzpi-23-5-samosvatova-anastasiia-lab1-code>