QXD0088: Lista de Exercícios #2

Guilherme Santos Rodrigues
guilhermerodrigues@alu.ufc.br

Universidade Federal do Ceará — August 24, 2023

Introduction

This project comprises a series of exercises designed to familiarize students with terminal usage. The finalized document is accessible on my personal GitHub Gist, and detailed descriptions for each function are provided in the subsequent sections.

1 Setup the environment

To initiate the creation of the necessary files, folders, and links required for this assessment, we will first establish the foundational directory structure. This structure will serve as the groundwork for successfully completing the remaining exercises.

script.sh

```
#!/bin/bash

rootFolder="atividades"
levelOneFolder="atividade2"
professorsFolder="professores"
classesFolder="disciplinas"
historicFolder="historico"

clearProject() {
    rm -rf $rootFolder
}

createInitialStructure() {
    levelTwoFolders=("professores" "disciplinas" "historico")

    mkdir "$rootFolder"
    mkdir "$rootFolder/$levelOneFolder"

    for folder in "${levelTwoFolders[@]}"; do
        mkdir "$rootFolder/$levelOneFolder/$folder"
    done
}

clearProject
createInitialStructure
tree .
```

2 Create the professors

The `createProfessors()` function is a vital tool in this assessment, enabling the automated extraction and categorization of professor details from a university's webpage. By leveraging the curl command, the function retrieves HTML content from a designated URL. It then employs a combination of `grep` and `sed` commands to identify and extract professor names based on a defined pattern. These names are subsequently processed in a loop, resulting in the creation of individual text files for each professor. This function exemplifies how shell scripting can streamline data collection and organization tasks, contributing to a deeper understanding of text manipulation and file handling techniques.

script.sh

```
createProfessors() {
    professorsURL="https://www.quixada.ufc.br/docente/"

    curl -s $professorsURL > docentes.html

    # Extract professor names using grep and sed from an HTML file
    professor_names=$(grep -o '<a␣href="https://www\.quixada\.ufc\.br
       /docente/[^"]*"␣class="btn␣btn-info␣btn-xs">' docentes.html |
        sed -r 's_^<a␣href="https://www\.quixada\.ufc\.br/docente
           /([^"]*)/"␣class="btn␣btn-info␣btn-xs">_\1_')

    # Loop through professor names and display them
    for name in $professor_names; do
        touch "$rootFolder/$levelOneFolder/$professorsFolder/$name.
           txt"
    done

    rm docentes.html
}
```

3 Create the classes

The `createClasses()` function fetches content from a URL containing class information. It iterates through the content, sanitizes and formats it by removing accents, converting to lowercase, and replacing spaces with underscores. It then creates corresponding directories and files for each class, including a text file and a historical subdirectory. This function streamlines the organization of class data into a structured directory hierarchy.

script.sh

```
createClasses() {
    url="https://gist.githubusercontent.com/guiRodrigues/
        c5793e3dffe48f4ed6b040d94b15fc6e/raw/8299
        e01078276e285124b9b4a22893bb042e095d/disciplinas.txt"

    curl -s "$url" |
    while read -r content || [[ -n "$content" ]]; do
        content=$(echo "$content" | sed 's#^/##')

        # Create a folder for each content
        content_dir="$OUTPUT_DIR/$(echo␣"$content"␣|␣tr␣'␣'␣'_')"

        # Remove the leading '/'
        content=$(echo "$content_dir" | sed 's#^/##')

        # Replace accented characters with non-accented characters
        converted_name=$(echo "$content" | iconv -f UTF-8 -t ASCII//
            TRANSLIT | tr -d -c '[:alnum:]_')

        # Convert to lowercase and replace underscores with spaces
        directory_name=$(echo "$converted_name" | tr '[:upper:]' '[:
            lower:]')

        touch "$rootFolder/$levelOneFolder/$classesFolder/
            $directory_name.txt"
        mkdir "$rootFolder/$levelOneFolder/$historicFolder/
            $directory_name"
    done
}
```

4 Create the history with Symbolic links

The `createHistory()` function orchestrates the creation of a historical record. This function works by iterating through designated folders, each representing a historical period. Within each historical period folder, symbolic links are established to relevant source text files from a separate classes folder. Additionally, a `createProfessorLink()` sub-function is utilized to create symbolic links to randomly selected professor files, fostering an association between historical periods and professors. The resulting setup efficiently links historical content to associated course details and professors, offering a comprehensive historical overview.

script.sh

```
createHistory() {
    createProfessorLink() {
        folder="$1"

        # Get a list of files in the folder
        files=("$rootFolder/$levelOneFolder/$professorsFolder"/*)

        # Get a random index within the range of file count
        random_index=$((RANDOM % ${#files[@]}))

        # Get the random file name
        random_file="${files[random_index]}"

        # Create a symbolic link to the professor
        ln -s "$random_file" "$folder/professor"
    }

    for folder in "$rootFolder/$levelOneFolder/$historicFolder"/*; do
        if [ -d "$folder" ]; then
            # Extract the folder name
            folder_name=$(basename "$folder")

            # Create a symbolic link to the source txt file in each
               folder
            ln -s "$rootFolder/$levelOneFolder/$classesFolder/
               $folder_name.txt" "$folder/programa"

            createProfessorLink $folder
        fi
    done
}
```

5 Conclusion and final result

In conclusion, this document has introduced a set of exercises designed to familiarize students with terminal operations. The final version, available on my personal GitHub Gist, provides detailed explanations for each function. The journey began with setting up the necessary environment, establishing the groundwork for subsequent tasks. The `createProfessors()` function demonstrated efficient professor extraction from a university webpage using text manipulation techniques. The `createClasses()` function streamlined class data organization by fetching content and creating directories. Lastly, the `createHistory()` function orchestrated historical record creation, linking periods, courses, and professors through symbolic links. This comprehensive script showcases the potency of shell scripting in automating various tasks and enhancing data management within a terminal framework.

In order to run the script, you type `chmod +x script.sh` and `./script.sh`. These commands will make the script executable and then run it. The terminal output will display the generated content, directories, and processed information based on the functions described in this document.

Command Line

```
$ chmod +x script.sh
$ ./script.sh

└── atividades
    └── atividade2
        ├── disciplinas
        │   ├── arquitetura_de_computadores.txt
        │   ├── arquitetura_de_software.txt
        │   ├── estruturas_de_dados.txt
        │   └── ...
        ├── historico
        │   ├── arquitetura_de_computadores
        │   │   ├── professor -> atividades/../samy-soares-passos-de-sa.txt
        │   │   └── programa -> atividades/../arquitetura_de_computadores.txt
        │   ├── arquitetura_de_software
        │   │   ├── professor -> atividades/../wagner-guimaraes-al-alam.txt
        │   │   └── programa -> atividades/../arquitetura_de_software.txt
        │   ├── estruturas_de_dados
        │   │   ├── professor -> atividades/../tania-saraiva-de-melo-pinheiro.txt
        │   │   └── programa -> atividades/../estruturas_de_dados.txt
        │   └── ...
        └── professores
            ├── alisson-barbosa-de-souza.txt
            ├── andre-ribeiro-braga.txt
            ├── andreia-liborio-sampaio.txt
            └── ...
```