Network Administration and System Administration

Spring 2024

Homework 2: Shell Script

B12902080 資工一吳威錡

本作業除了參考額外標記超連結的網址外，部分題目亦參考 ChatGPT, 並與 B12902054 彭詳睿、B12902055 楊翔宇 討論，特別感謝楊翔宇親自生了一些測資讓我們能額外進行測試。

有紀錄中的所有參考網址：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19

Issue

我原本在自己電腦上寫此次的作業 (版本：Linux 6.5.0-21-generic #21~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC)，在上傳到工作站後發現有許多指令的版本問題導致原本全部 accepted 的程式碼會產生 wrong answer (例如工作站判斷 binary files 是要使用中文的" 二元碼檔"，但本機是要用英文的"Binary files")，上傳的程式碼檔案針對" Binary files" 和"Symbolic link" 皆使用中文與英文判斷，另外也將 LC_ALL 設爲英文。

Source Code

```bash
#!/bin/bash
#set -x
export LC_ALL="en_US.UTF-8"

print_help() {
    echo "usage: ./compare.sh [OPTION] <PATH A> <PATH B>"
    echo "options:"
    echo "-a: compare hidden files instead of ignoring them"
    echo "-h: output information about compare.sh"
    echo "-l: treat symlinks as files instead of ignoring them"
    echo "-n <EXP>: compare only files whose paths follow the REGEX <EXP>"
    echo "-r: compare directories recursively"
}

ropt=false
aopt=false
```

```
nopt=false
lopt=false
fe=true
exp=""
path_a=""
path_b=""

check() {
    local size="$#"
    if [[ "$size" -lt 2 ]]; then
        print_help
        exit 1
    fi
    for arg in "${@: -2}"; do
        if [ ! -e "$arg" ]; then
            print_help
            exit 1
        fi
        if [ ! -d "$arg" ] && [ ! -f "$arg" ] && [ ! -L "$arg" ]; then
            print_help
            exit 1
        fi
    done
    while getopts ":ahlrn:?" opt; do
        case $opt in
            a)
                aopt=true
                ;;
            h)
                print_help
                exit 1
                ;;
            l)
                lopt=true
                ;;
            n)
                nopt=true
                if [[ "$OPTARG" =~ "$exp" ]]; then
                    exp="$OPTARG"
                else
                    print_help
                    exit 1
                fi
                ;;
            r)
```

```
62                  ropt=true
63                  ;;
64              \?)
65                  print_help
66                  exit 1
67                  ;;
68          esac
69      done
70      shift $((OPTIND -1))
71      size="$#"
72      if [[ "$size" -ne 2 ]]; then
73          print_help
74          exit 1
75      fi
76      path_a="$1"
77      path_b="$2"
78      if [[ "$ropt" == "false" && ( "$nopt" == "true" || "$aopt" == "true" )
    ]]; then
79          print_help
80          exit 1
81      fi
82      if [[ "$ropt" == "true" && (! -d "$path_a" || ! -d "$path_b") ]]; then
83          print_help
84          exit 1
85      fi
86      if [[ "$ropt" == "false" && ( -d "$path_a" || -d "$path_b") ]]; then
87          if [[ ( -L "$path_a" && -f "$path_b") || ( -f "$path_a" && -L "
    $path_b" ) ]]; then
88              :
89          else
90              print_help
91              exit 1
92          fi
93      fi
94      if [[ "$lopt" == "false" && ( -L "$path_a" || -L "$path_b") ]]; then
95          print_help
96          exit 1
97      fi
98  }
99
100 check_exist() {
101     fe=true
102     if [[ -h "$1" && "$lopt" == "false" ]]; then
103         fe=false
104     fi
```

```
105     if echo "$2" | grep -qE '/\.[^/]*'; then
106         if [[ "$aopt" == "false" ]]; then
107             fe=false
108         fi
109     fi
110     if [[ ! -e "$1" ]]; then
111         fe=false
112     fi
113     if [[ -d "$1" && ( ! -h "$1" ) ]]; then
114         fe=false
115     fi
116     if [[ ! "$2" =~ "$exp" ]] && [[ "$nopt" == "true" ]]; then
117         fe=false
118     fi
119 }
120 
121 cmp_file() {
122     local diffo_output=$(diff -d --no-dereference "$1" "$2")
123     local diff_output=$(diff -d -u --no-dereference "$1" "$2" | grep -vE '
    ^(---|\+\+\+|@@)' 2>&1)
124     local result=$(echo "$diffo_output")
125     local x
126     if [[ ! -n "$result" ]]; then
127         return
128     fi
129     if [[ $result == *"Binary files"* || $result == *"二元碼檔"* || $result
     == *"symbolic link"* || $result == *"Symbolic link"* || $result == *"符
    號連結"* ]]; then
130         x=100
131     else
132         local a=$(echo "$diff_output" | grep -c "^-" )
133         local b=$(echo "$diff_output" | grep -c "^+" )
134         local tota=$(grep -a "^" "$1" | wc -l)
135         local c=$((tota-a))
136         local max=$((b > a ? b : a))
137         if [[ $max == 0 && $c == 0 ]]; then
138             x=0
139         else
140             x=$((100 * max / (max + c)))
141         fi
142     fi
143     echo "changed ${x}%"
144 }
145 
146 cmp_dir() {
```

```
147     declare -a file_status=()
148     acom=""
149     lcom="-type f"
150     if [[ "$aopt" == "false" ]]; then
151         acom="-not -path '*/\.*'"
152     fi
153     if [[ "$lopt" == "true" ]]; then
154         lcom="\( -type f -o -type l \)"
155     fi
156     ec1="cd $1 && find ./ $lcom $acom"
157     ec2="cd $2 && find ./ $lcom $acom"
158     en1=$(eval "$ec1")
159     en2=$(eval "$ec2")
160     while IFS= read -r file1; do
161         rel_path="${file1#\./}"
162         fa="$1/""$rel_path"
163         fb="$2/""$rel_path"
164         check_exist "$fb" "$rel_path"
165         if [[ ! -n $file1 || ( "$nopt" == "true" && ! $rel_path =~ "$exp" )
    ]]; then
166             continue
167         fi
168         if [[ "$fe" == "true" ]]; then
169             result="$(cmp_file "$fa" "$fb")"
170             if [[ -n "$result" ]]; then
171                 file_status+=("$rel_path! $result")
172             fi
173         else
174             file_status+=("$rel_path delete")
175         fi
176     done < <(echo "$en1")
177     while IFS= read -r file2; do
178         rel_path="${file2#\./}"
179         fa="$1/""$rel_path"
180         fb="$2/""$rel_path"
181         check_exist "$fa" "$rel_path"
182         if [[ ! -n $file2 || ( "$nopt" == "true" && ! $rel_path =~ "$exp" )
    ]]; then
183             continue
184         fi
185         if [[ "$fe" == "false" ]]; then
186             file_status+=("$rel_path create")
187         fi
188     done < <(echo "$en2")
189     LC_ALL=C
```

```
    IFS=$'\n' sorted_status=($(sort <<<"${file_status[*]}"))
    for status in "${sorted_status[@]}"; do
        case $status in
            *create)
                echo "create ${status% create}"
                ;;
            *delete)
                echo "delete ${status% delete}"
                ;;
            *)
                outp="$(echo "${status}" | sed 's/!/:/g' )"
                echo "$outp"
                ;;
        esac
    done
}

check "$@"

remove_slash() {
    path="$1"
    while [[ "${path: -1}" == '/' ]]; do
        path="${path:0:-1}"
    done
}

remove_slash "$path_a"
path_a="$path"
remove_slash "$path_b"
path_b="$path"

if [[ "$ropt" == "true" && ( -d "$path_a" && -d "$path_b") ]]; then
    cmp_dir "$path_a" "$path_b"
elif diff -qr "$path_a" "$path_b" >/dev/null; then
    :
else
    cmp_file "$path_a" "$path_b"
fi
```

Listing 1: Source Code

subtask 1

1. 做題過程

根據題目敘述，我列出了幾個需要輸出方框中內容的情況：

1. 有輸入 -h
2. 使用沒有定義的參數
3. -n 的後面沒有接 regex 字串
4. 有 -a -n 參數但沒有 -r 參數
5. 有 -r 參數但 <PATH A>, <PATH B> 有一個以上不是目錄
6. 沒有 -r 參數但 <PATH A>, <PATH B> 有一個以上是目錄
7. 最後兩個參數不是 <PATH A>, <PATH B>
8. 沒有 -l 但 <PATH A>, <PATH B> 有一個以上是 symlink
9. 不存在 <PATH A> 或 <PATH B>
10. 在參數中出現無意義的字串

根據以上情況寫出程式碼，並且自己生測資確認上述情況都會輸出方框中內容。

2. 程式碼解釋

- 5-13 行：印出錯誤訊息
- 15-22 行：初始化變數
- 26-29 行：如果所有輸入參數不及兩個，代表根本沒有輸入 <PATH A>, <PATH B>，觸發第 7 種情況，輸出錯誤訊息
- 30-39 行：如果最後兩項參數不符合輸入檔案的格式或無法執行，代表可能觸發第 7, 9 種情況，輸出錯誤訊息
- 40-69 行：讀取輸入的參數，並更新變數紀錄供後續程式使用，如果有輸入 -h (第 1 種情況)、讀取未定義的參數 (第 2 種情況)、-n 後字串不符合 *regex* 格式 (第 3 種情況)，輸出錯誤訊息
- 70-77 行：將 [OPTION] 部分移出，使 <PATH A>, <PATH B> 成爲陣列內唯二的元素，如果移除 [OPTION] 部分後參數不及兩個，代表可能觸發第 7, 10 種情況，輸出錯誤訊息

- 78-81 行：如果有 -a -n 參數但沒有 -r 參數，觸發第 4 種情況，輸出錯誤訊息
- 82-85 行：如果有 -r 參數但 <PATH A>, <PATH B> 有一個以上不是目錄，觸發第 5 種情況，輸出錯誤訊息
- 86-93 行：如果沒有 -r 參數但 <PATH A>, <PATH B> 有一個以上是目錄，觸發第 6 種情況，輸出錯誤訊息 (但如果其中一個是 link，即便它指到的是目錄，但根據題目規定應視爲檔案，不在此限)
- 94-97 行：如果沒有 -l 但 <PATH A>, <PATH B> 有一個以上是 symlin，觸發第 8 種情況，輸出錯誤訊息

subtask 2

1. 做題過程

將兩檔案比較實做成 cmp_file 函式。

2. 程式碼解釋

- 122 行：使用 `diff -d` 函式比較兩檔案，用來判斷結果是否不需處理 (兩檔案相同) 或是爲 ${x}:=100 的情況
- 123 行：使用 `diff -d -u` 函式比較兩檔案，用來計算 a, b, c 的值
- 126-128 行：如果使用 `diff` 指令比較後爲空檔案，代表兩檔案完全相同，不須輸出
- 129-130 行：如果使用 `diff` 指令比較後輸出 Binary files 或 Symbolic link，根據題目敘述，`${x}:=100`(因中英文版本問題，這邊條件同時列出中英文)
- 131-141 行：計算兩檔案不同時的 ${x} 值
- 143 行：輸出 ${x} 值

subtask 3, 4, 5

1. 做題過程

將兩目錄比較實做成 cmp_dir 函式。先根據使用者設定參數決定使用的 `cd`, `find` 指令參數，並分別讀取 <PATH A>, <PATH B> 內的所有目錄、檔案、鏈結進行檢查、並與另一個 PATH 對應之檔案進行比較，將結果存至陣列後排序輸出。

2. 程式碼解釋

- 100-119 行：檢查該路徑與變數是否符合設定的參數規範，如果不符合就將變數 fe 設爲 false
- 147-159 行：初始化變數、設定使用指令 `cd`, `find` 的參數
- 160-176 行：跑過所有 <PATH A> 的檔案、目錄、鏈結等，檢查 <PATH B> 是否有存在相同且合法的檔案，若有則進行 cmp_file 函式計算 ${x} 值；若無則試情況決定是要輸出 delete 還是不需輸出
- 177-188 行：跑過所有 <PATH A> 的檔案、目錄、鏈結等，檢查 <PATH B> 是否有存在不相同且合法的檔案，若有則輸出 create
- NOTE: 爲了排序方便，放入陣列的格式爲 $filename create, $filename delete, $filename! changed x%，這樣使用一次 `sort` 即可照題目要求排序
- 189-190 行：排序
- 191-204 行：改回正確格式輸出
- 207 行：執行 check 函式檢查輸入的參數是否合法，是否需要輸出錯誤訊息
- 209-219 行：處理有後綴"/" 的問題
- 221-222 行：如果兩者皆是目錄，則使用 cmp_dir 函式比對
- 223-224 行：如果兩者完全相同，則不進行任何操作 (此部分與先前程式有些重複)
- 225-226 行：如果兩者皆是檔案，則使用 cmp_file 函式比對