Міністерство освіти України
Харківський національний університет радіоелектроніки

Звіт до лабораторної роботи № 1
з курсу «Операційні системи UNIX»

Виконав: ст. гр. ПЗПІ-16-3
Губар С.О.
Прийняв: Сокорчук І.П.

Харків 2020

Мета роботи: Вивчити структуру файлової системи ОС Unix, основні команди операційної системи, застосування каналів обміну інформацією між двома командами операційної системи. Познайомитися з Bash, та застосувати ці знання для того, щоб написати Bash скрипт, який буде збирати інформацію про систему.

Хід роботи

Створимо файл task1.sh, у початку файла вкажемо, що його слід виконувати у bash:

```
#!/bin/bash
```

Після чого реалізуємо опцію -h (--help) для нашого скрипта, перевіряючи значення змінної $1

```
if [ "$1" = "--help" ] || [ "$1" = "-h" ]; then
   echo "Usage ./task1.sh [-h | --help] [-n num] [file]";
   echo "Script gathers system information and writes it to the [file] option";
      echo "[n] parameter specifies, how many files with the similar name can be in output directory";
   exit 0;
fi
```

Перевіримо кількість аргументів, з якими було запущено скрипт, якщо їх більше трьох - виводимо помилку у stderr, та зупиняємо виконання

```
if (($# > 3)) ; then
   if [[ $LANG =~ uk_UA ]]; then
      echo "Забагато аргументів" 1>&2
   else
      echo "Too many arguments" 1>&2;
   fi
   exit 1;
fi
```

Якщо кількість аргументів не перевищує 3, перевіримо за допомогою getopts той аргумент, який передається у опції -n (він повинен бути числом більшим за одиницю):

```
while getopts "n:" OPTION; do
   if [[ "$OPTARG" =~ ^-?[0-9]+$ ]]; then
      if ! (($OPTARG >= 1)); then
         if [[ $LANG =~ uk_UA ]]; then
            echo "-n має бути числом, більшим за 1" 1>&2
         else
            echo "-n must be an integer > 1, got $OPTARG" >&2
         fi
         exit 1
      else
         number_of_files=$OPTARG
      fi
   else
      if [[ $LANG =~ uk_UA ]]; then
         echo "-n має бути числом" 1>&2
      else
         echo "-n must be an integer, got $OPTARG" >&2
      fi
      exit 1
   fi
done
```

Перевіряємо файл, у який будемо записувати інформацію. Якщо він не заданий - виставляємо значення за замовчуванням

```
# Default file
if [ -z "$outFile" ]; then
   outFile="$HOME/bash/task1.out"
fi
```

Перевіряємо директорію, у яку будемо записувати, та якщо її не має - створюємо

```
check_directory() {
```

```
    if [ ! -d $1 ]; then
        echo "Directory was not found and will be automatically created."
        mkdir -p "$1" 2>&1
    fi
    if [ $? -ne 0 ]; then
        if [[ $LANG =~ uk_UA ]]; then
            echo "Директорію не було створено" 1>&2
        else
            echo "Error creating directory" 1>&2
        fi
        exit 1
    fi
}
```

Напишемо код, який визначає кількість файлів за день з однією назвою, та видаляє один, якщо кількість дорівнює максимальній кількості, яка задається параметром -n

```
# File rotation

parent_directory=$(dirname "$output_file")
file_name=$(basename $output_file)

check_directory "$parent_directory"

current_date=$(date "+%Y%m%d")
filename_with_date="${file_name}-${current_date}"



if [ -f "$output_file" ]; then
    # get all files created today with numbers (like 0000)
    all_files_with_numbers=($(ls -v "$parent_directory" | grep -E "^$filename_with_date"))
    # number of these files
    length=$(ls -v "$parent_directory" | grep -E "^$filename_with_date" | wc -l)
```

```
    if [ $length -ne 0 ]; then
        last_created_file_name=${all_files_with_numbers[$((length - 1))]}
        # remove all text, leave only number (0000)
        last_created_number=($(echo $last_created_file_name | grep -oP "\d{4}$"))
        for ((i = $length - 1; i >= 0; i--)); do
            num=$(($i + 1))
            new_number_formatted="$(printf "%04d" $num)"
                                                mv     "$parent_directory/${all_files_with_numbers[$i]}"
"$parent_directory/${filename_with_date}-${new_number_formatted}"
        done
    fi

    # move task.out -> task1.out-date-0000 (last move)
    mv "$output_file" "${output_file}-${current_date}-0000"
fi

# Write all data
collect "$output_file"

remove_old_files
```

Висновки:

У ході виконання лабораторної роботи було написано скрипт, який валідує задані йому аргументи, підтримує аргумент --help для виводу інформації про роботу скрипта, а також містить логіку створення файла, який буде використано для запису інформації у лабораторній роботі №2. Також скрипт містить логіку визначення максимальної кількості файлів, які можна зберігати, та автоматично видаляє старі файли, щоб створити новий.