БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Кафедра информатики

Факультет: ИНО
Специальность: ИиТП

Индивидуальная практическая работа № 1
по дисциплине “Операционные системы и среды”
Вариант 6
“Средства пользовательского уровня в среде Unix/Linux”

Выполнил студент: Дубейковский А.А.
Группа № 893551
Зачётная книжка № 75350046

Условие

Методическая часть

Цель работы – практическое изучение способов реализация типичных задач пользователя системы средствами *shell*, утилит и скриптовых языков

Выполнение работы предполагает написание сценария (скрипта) или нескольких взаимосвязанных сценариев для командного интерпретатора *shell* и, возможно, других интерпретаторов, демонстрирующего выполнение поставленной задачи. Необходимо обеспечить корректное поведение сценариев с различными входными данными и обработку типичных ошибок.

Задания состоят из трех подзаданий.

Подзадание **1** (общее) – поиск файлов в файловой системе с обходом дерева директориев. Алгоритм поиска (обхода дерева) реализуется средствами *shell*, без специализированных утилит (*find* и т.п.), но с использованием произвольных внешних средств для сравнения, вычислений и т.д.

Подзадание **2** (специальное) – действия над найденными файлами. Ограничений на набор используемых команд (утилит) не накладывается.

Подзадание **3** (общее) – оформление выводимых результатов. Ограничений на набор используемых команд (утилит) не накладывается.

Рекомендуется по возможности сопровождать файлы скриптом *make* (*makefile*), автоматизирующим тестовые и демонстрационные запуски.

У подзадания 1 всего два варианта, т.к. мой вариант 6й по зачётной книжке, я взял чётный из данных.

Вариант подзадания № 1.2.

Поиск файла по имени, образец поиска задан списком (имена проверяются на совпадение с любым из этого списка). Способ передачи списка – ряд аргументов в командной строке. Поиск начинается от текущего директория.

Вариант подзадания № 2.6.

Подсчет контрольной суммы файлов (суммы значений байтов или слов): каждого найденного файла и общей по всем файлам. Вместо суммирования можно использовать другую доступную функцию.

У подзадания 3 только один вариант

Вариант подзадания № 3.1.

Дополнить результаты работы сценария «подписью», выводимой в поток *stderr* и включающей:

– имя стартового каталога (актуальное значение);
– имя пользователя (актуальное значение);
– текущая дата;
– время выполнения сценария.

Если результатом являются новые файлы, то в их содержимое «подпись» не включается.

Скрипт и пояснения к нему

Все необходимые пояснения написаны в комментариях в скрипте, который будет показан тут в двух вариантах (как фото из sublime text для удобного восприятия и чтения, и как текст для копирования). А также скрипт и тесты к нему (готовое дерево директорий и файлы в них) выложены на гитхабе: https://github.com/ElephantT/MyBashScripts/tree/main/MyRecursiveFind

Код из sublime text (удобный для чтения):

```bash
#!/bin/bash

# ENG

# Operating systems and environments - individual practical work №1
# Aleksandr Dubeykovsky
# 1. Find all files in subdirectories that have names like arguments, that are passed to the script
#   1.1 restriction - you can not use shell's 'find' and etc
# 2. Find size of each file and size of all files together, that have been found
# 3. Print results and signature, signature should be prited in stderr
# Example on how to execute script: $ bash osis_ipr1_Dubeykovsky.sh file1.txt file7.txt file4.txt
# Search will start from directory where the script is stored

# RU

# ОСИС ИПР 1
# Александр Дубейковский - зачётная книжка 75350046, группа 893551, Вариант 6
# 1. Найти все файлы из списка, переданного как аргументы (не используя find и подобные функции shell)
# 2. Подсчитать общий размер найденных файлов и каждого из них
# 3. Вывести результаты, а так же 'подпись', 'подпись' в stderr
# Пример использования: $ bash osis_ipr1_Dubeykovsky.sh file1.txt file7.txt file4.txt
# Поиск будет начинаться с директории, где расположен скрипт

# var to save all pathes of found files by given filename requirements
declare -a list_of_found_filenames=()

myRecursiveFind() {
  # first argument - list of filenames to look for
  # second argument - directory path, where we check each object:
  #   if file fits search requirements - show it
  #   if not - skip
  #   if it is another directory - use same function for it

  # loop by all ls results of our current position on directory tree
  for obj in $(ls "${@: -1}")
  do
    # check if object is file or dir
    if [[ -f "${@: -1}/$obj" ]]
    then
      # we've found file
      # check if we are looking for this current file
      for name_to_check in ${@:1:$#-1}
      do
        # echo "$name_to_check"
        if [[ $obj == $name_to_check ]]
        then
          list_of_found_filenames+=("${@: -1}/$obj")
          echo "found filename: "$obj""
          echo "found path: ""${@: -1}/$obj"""
          echo "size = "$(stat -c%s "${@: -1}/$obj")" bytes"
          echo "---"
        fi
      done
    else
      # we've found dir
      # recursive call on dir
      myRecursiveFind ${@:1:$#-1} "${@: -1}/$obj"
    fi
  done
}
```

```
61
62
63  sumFilesSizes() {
64    # first argument - array with pathes of files
65    let FILE_SIZES=0
66    for file in $@
67    do
68      FILE_SIZES=$(($FILE_SIZES+$(stat -c%s "$file")))
69    done
70    printf "$FILE_SIZES"
71  }
72
73
75
76  let start=`date +%s`
77
78  declare -a list_of_filenames_to_find=()
79  for FILE_NAME in $@
80  do
81    list_of_filenames_to_find+=("$FILE_NAME")
82  done
83
84  printf "\nstart of search:\n---\n"
85  myRecursiveFind ${list_of_filenames_to_find[@]} "$(pwd)"
86  printf "end of search\n\n"
87
88  sumFilesSizes ${list_of_found_filenames[@]}
89
90  # i use sleep for 1 second, because for some tests script works almost instantly
91  sleep 1
92  let end=`date +%s`
93  declare -a signature=(
94    "current directory: $(pwd)"
95    "name of current user: $(whoami)"
96    "current date: $(date +%D)"
97    "runtime in seconds: $(($end-$start))"
98  )
99
100 # when we write in shell to stderr, it uses standart write fucntion, which doesn't use red color for text,
101 # but still writes to stderr, so i used some shell functionality to color output in red,
102 # so stderr output would be visually different
103 for signature_str in "${signature[@]}"
104 do
105   echo -e "\e[01;31m$signature_str\e[0m" >&2
106 done
107
```

Код скрипта для копирования:

```
#!/bin/bash

# ENG

# Operating systems and environments - individual practical work №1
# Aleksandr Dubeykovsky
# 1. Find all files in subdirectories that have names like arguments, that are passed to the script
#   1.1 restriction - you can not use shell's 'find' and etc
# 2. Find size of each file and size of all files together, that have been found
# 3. Print results and signature, signature should be prited in stderr
# Example on how to execute script: $ bash osis_ipr1_Dubeykovsky.sh file1.txt file7.txt file4.txt
# Search will start from directory where the script is stored

# RU

# ОСИС ИПР 1
```

```
# Александр Дубейковский - зачётная книжка 75350046, группа 893551, Вариант 6
# 1. Найти все файлы из списка, переданного как аргументы (не используя find и подобные функции shell)
# 2. Подсчитать общий размер найденных файлов и каждого из них
# 3. Вывести результаты, а так же 'подпись', 'подпись' в stderr
# Пример использования: $ bash osis_ipr1_Dubeykovsky.sh file1.txt file7.txt file4.txt
# Поиск будет начинаться с директории, где расположен скрипт

# var to save all pathes of found files by given filename requirements
declare -a list_of_found_filenames=()

myRecursiveFind() {
  # first argument - array of filenames to look for
  # second argument - directory path, where we check each object:
  #   if file fits search requirements - show it
  #   if not - skip
  #   if it is another directory - use same function for it

  # loop by all ls results of our current position on directory tree
  for obj in $(ls "${@: -1}")
  do
    # check if object is file or dir
    if [[ -f "${@: -1}/$obj" ]]
    then
      # we've found file
      # check if we are looking for this current file
      for name_to_check in ${@:1:$#-1}
      do
        # echo "$name_to_check"
        if [[ $obj == $name_to_check ]]
        then
          list_of_found_filenames+=("${@: -1}/$obj")
          echo "found filename: "$obj""
          echo "found path: ""${@: -1}/$obj""""
          echo "size = "$(stat -c%s "${@: -1}/$obj")" bytes"
          echo "---"
        fi
      done
    else
      # we've found dir
      # recursive call on dir
      myRecursiveFind ${@:1:$#-1} "${@: -1}/$obj"
    fi
```

```
  done
}


sumFilesSizes() {
  # first argument - array with pathes of files
  let FILE_SIZES=0
  for file in $@
  do
    FILE_SIZES=$(($FILE_SIZES+$(stat -c%s "$file")))
  done
  printf "Size of all found files = "
  printf "$FILE_SIZES"
  printf " bytes\n"
}


let start=`date +%s`

declare -a list_of_filenames_to_find=()
for FILE_NAME in $@
do
  list_of_filenames_to_find+=("$FILE_NAME")
done

printf "\nstart of search:\n---\n"
myRecursiveFind ${list_of_filenames_to_find[@]} "$(pwd)"
printf "end of search\n\n"

sumFilesSizes ${list_of_found_filenames[@]}

# i use sleep for 1 second, because for some tests script works almost instantly
sleep 1
let end=`date +%s`
declare -a signature=(
  "current directory: $(pwd)"
  "name of current user: $(whoami)"
  "current date: $(date +%D)"
  "runtime in seconds: $(($end-$start))"
)

# when we write in shell to stderr, it uses standart write fucntion, which doesn't use red color for text,
# but still writes to stderr, so i used some shell functionality to color output in red,
# so stderr output would be visually different
```

```
for signature_str in "${signature[@]}"
do
  echo -e "\e[01;31m$signature_str\e[0m" >&2
done
```

Тесты

Тесты запускались на ОС семейства Linux – Ubuntu 20.04.3 LTS (Focal Fossa).

Структура директорий и их содержание для тестов:

```
(base) elephant@seaelephant:~/University/OS/osis_ipr$ ls
file1.txt  folder2  hw.sh  osis_ipr1_Dubeykovsky.sh
(base) elephant@seaelephant:~/University/OS/osis_ipr$ cd folder2/ ; ls
file2.txt  file3.txt  folder10  folder3
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2$ cd folder10/ ; ls
file4.txt
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2/folder10$ cd .. ; cd folder3/ ; ls
file4.txt  file5.txt
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2/folder3$
```

Запустим скрипт:

```
(base) elephant@seaelephant:~/University/OS/osis_ipr$ ls
file1.txt  folder2  hw.sh  osis_ipr1_Dubeykovsky.sh
(base) elephant@seaelephant:~/University/OS/osis_ipr$ cd folder2/ ; ls
file2.txt  file3.txt  folder10  folder3
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2$ cd folder10/ ; ls
file4.txt
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2/folder10$ cd .. ; cd folder3/ ; ls
file4.txt  file5.txt
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2/folder3$ cd ..
(base) elephant@seaelephant:~/University/OS/osis_ipr/folder2$ cd ..
(base) elephant@seaelephant:~/University/OS/osis_ipr$ bash osis_ipr1_Dubeykovsky.sh file1.txt file7.txt file4.txt

start of search:
---
found filename: file1.txt
found path: /home/elephant/University/OS/osis_ipr/file1.txt
size = 11 bytes
---
found filename: file4.txt
found path: /home/elephant/University/OS/osis_ipr/folder2/folder10/file4.txt
size = 3970 bytes
---
found filename: file4.txt
found path: /home/elephant/University/OS/osis_ipr/folder2/folder3/file4.txt
size = 83 bytes
---
end of search

Size of all found files = 4064 bytes
current directory: /home/elephant/University/OS/osis_ipr
name of current user: elephant
current date: 01/02/22
runtime in seconds: 1
(base) elephant@seaelephant:~/University/OS/osis_ipr$
```

Данный скрипт, а также директории и файлы для тестов (из примера выше) можно скачать на гит-хабе и запустить на своём компьютере в linux shell: https://github.com/ElephantT/MyBashScripts/tree/main/MyRecursiveFind