Отчет по лабораторной работе N 21 по курсу

"Фундаментальная информатика"

Студент группы: М80-107Б-21, Павлов Иван Дмитриевич
Контакты: pavlov.id.2003@gmail.com
Работа выполнена: 02.03.2022
Преподаватель: Найденов Иван Евгеньевич

1 Тема

Программирование на интерпретируемых командных языках.

2 Цель работы

Составить программу выполнения заданных действий над файлами на языке Bash, а также на любом другом языке.

3 Задание

Вариант №18. Рекурсивный обход указанного каталога и замена всех разделителей пути / на в файлах с именем Makefile*. Программа должна удовлетворять следующим условиям:

- Программа должна обеспечивать возможность установки режима подтверждения
- Если параметры опущены, то устанавливаются некоторые стандартные значения параметров

4 Оборудование

Процессор: AMD Ryzen 5 4600H with Radeon Graphics
ОП: 7851 Мб
НМД: 256 Гб
Монитор: 1920x1080

5 Программное обеспечение

Операционная система семейства: linux (ubuntu), версия 20.04.3 LTS
Интерпретатор команд: bash, версия 5.0.17(1)-release.
Система программирования: gcc*, версия 17
Редактор текстов: emacs, версия 25.2.2
Утилиты операционной системы: awk
Прикладные системы и программы: bash, C
Местонахождение и имена файлов программ и данных на домашнем компьютере: /home/ggame/newlabs/

6 Идея и методы реализации задачи

Реализация на bash:

- Задание предполагает рекурсивный обход директории, поэтому начнем с реализации рекурсии на bash. Реализуем ее с помощью функции, внутри которой будем циклом for проверять входной аргумент (ls $1/) разделенный знаком "/") на тип данных (файл или директория), с помощью операторов -d и -f. Если при этом обнаружена директория, рекурсивно запускаем функцию.
- Если переменная filename - файл, не начинающийся на "Makefile выводим путь к файлу в консоль. Создадим строку, в которую будем рекурсивно записывать полный путь до файла и менять в нем "/"на "; с помощью sed.

- Проверяем filename на префикс "Makefile". Если это так, то выведем строку.
- Реализуем режим подтверждения и выполнение скрипта без аргументов, с помощью if.

Реализация на Си:

- Воспользуемся находящейся в библиотеке <dirent.h> структуры dirent, а также функциями opendir и readdir.
- Создадим структуру dirent* entity (сущность); с помощью entity->d_type можно узнать тип сущности (файл или папка), с помощью entity->d_name можно узнать имя файла/папки.
- В качестве потока ввода используем массив argv[].
- все остальное совпадает с реализацией на bash.

7 Примеры вывода в терминале

bash:

```
ggame@ggame:~/newlabs/lab21_r$ ./shript2.sh -p .
Continue (y/n)?y
./1/10.txt
./1/1.txt
./1/2.txt
./1/3.txt
./1/4.txt
./1/5.txt
./1/6.txt
./1/7.txt
./1/8.txt
./1/9.txt
.\1\Makefile10.txt
.\1\Makefile1.txt
.\1\Makefile2.txt
.\1\Makefile3.txt
.\1\Makefile4.txt
.\1\Makefile5.txt
.\1\Makefile6.txt
.\1\Makefile7.txt
.\1\Makefile8.txt
.\1\Makefile9.txt
./a.out
./code.c
./d1.png
./d2.png
./d3.png
./d4.png
./d5.png
./d6.png
./english_text.txt
./linal.jpg
.\Makefile_egnweighewg
.\Makefile_ttttt
./pwd.txt
./russian_text.txt
./script1.sh
./shript2.c
./shript2.sh
```

C:

```
ggame@ggame:~/newlabs/lab21_r$ gcc code.c
ggame@ggame:~/newlabs/lab21_r$ ./a.out -p .
Continue (y/n)?y
.d1.png
.pwd.txt
.russian_text.txt
.d2.png
.english_text.txt
.shript2.c
.\Makefile_ttttt
.linal.jpg
.\Makefile_egnweighewg
.code.c
```

```
.d5.png
.script1.sh
./12.txt
./11.txt
.\1\Makefile4.txt
./18.txt
.\1\Makefile7.txt
./110.txt
.\1\Makefile1.txt
./13.txt
.\1\Makefile10.txt
./17.txt
./19.txt
./15.txt
.\1\Makefile9.txt
./14.txt
./16.txt
.\1\Makefile8.txt
.\1\Makefile3.txt
.\1\Makefile2.txt
.\1\Makefile6.txt
.\1\Makefile5.txt
.d3.png
.d4.png
.a.out
.d6.png
.shript2.sh
```

8 Распечатка протокола

bash:

```
#!/usr/bin/env bash

RecursiveDirectory(){
 text=''
 for fileName in $(ls $1/)
 do
  text="$( echo $1 | sed 's/\//\\/g')"
  if [[ -f "$1/$fileName" ]]; then
   ismk=$( echo $fileName | cut -c1-8)
   if [ "$ismk" = "Makefile" ]; then
    echo "$text\\$fileName" | sed 's/\\\\/\\/g'
    text=''
    continue
   else
    echo "$1/$fileName" | sed 's/\/\//\//g'
    continue
   fi
  fi

  if [ -d "$1/$fileName" ]; then
   RecursiveDirectory "$1/$fileName"
   continue
  fi
 done
}

if [[ $1 = "-p" ]]; then
 read -p "Continue (y/n)?" cont
 if [ "$cont" = "y" ]; then
  if [ -z "$2" ]; then
   RecursiveDirectory .
  else
   RecursiveDirectory $2
  fi
  exit 0
 else
  exit 1
 fi
else
 if [ -z "$1" ]; then
  RecursiveDirectory .
 else
  RecursiveDirectory $1
```

```
 fi
fi
```

C:

```
#include <stdio.h>
#include <string.h>
#include <dirent.h>

void listFiles(const char* dirname)
{
 int n = 0;
 char text[1024] = { 0 };
 char text_1[1024] = { 0 };
 DIR* dir = opendir(dirname);
 if (dir == NULL) {
  return;
 }

 struct dirent* entity;
 entity = readdir(dir);
 while (entity != NULL) {
  if (entity->d_type == DT_REG && strcmp(entity->d_name, ".") != 0 &&
  strcmp(entity->d_name, "..") != 0) {
   if (entity->d_name[0] == 77 && entity->d_name[1] == 97 && entity->d_name[2]
   == 107 && entity->d_name[3] == 101 && entity->d_name[4] == 102 &&
   entity->d_name[5] == 105 && entity->d_name[6] == 108 && entity->d_name[7] ==
   101) {
    strcat(text, dirname);
    strcat(text, "/");
    strcat(text, entity->d_name);
    while (text[n] != '\0') {
     n++;
    }
    //n--;

    for (int i = 0; i < n; i++) {
     if (text[i] == '/') {
      printf("\\");
     } else {
      printf("%c", text[i]);
     }
    }
    printf("\n");
    memset(text,0,1024);
    memset(text_1,0,1024);
    n = 0;
   } else {
    printf("%s%s\n", dirname, entity->d_name);
   }
  }
  if (entity->d_type == DT_DIR && strcmp(entity->d_name, ".") != 0
  && strcmp(entity->d_name, "..") != 0) {
   char path[1024] = { 0 };
   strcat(path, dirname);
   strcat(path, "/");
   strcat(path, entity->d_name);
   listFiles(path);
  }
  entity = readdir(dir);
 }
 closedir(dir);
}

int main(int argc, char const *argv[])
{
 char CONT;
 if (argv[1] == NULL) {
  listFiles(".");
 } else {
  if (strcmp(argv[1], "-p") == 0) {
   printf("Continue (y/n)?");
   scanf("%c", &CONT);
   if (CONT == 'y') {
    if (argv[2] == NULL) {
     listFiles(".");
    } else {
```

```
      listFiles(argv[2]);
     }
    } else {
     return 0;
    }
   } else {
    if (argv[1] == NULL) {
     listFiles(".");
    } else {
     listFiles(argv[1]);
    }
   }
  }
  return 0;
}
```

9 Вывод

В ходе данной лабораторной работы я:

- Изучил ЯП bash и C
- Познакомился со структурой директорий в ОС linux и ее реализацией в Си
- Понял, что на языке bash скрипты работают медленнее, чем на Си. Так как сам bash был написан на C, значит, то, что можно реализовать на bash, можно реализовать на Си
- Однако bash является одним из самых простых языков для написания скриптов, он поддерживает команды терминала linux, что невозможно на Си.

Поэтому я убедился в том, что программу для выполнения действий над файлами лучше всего писать на bash.