Министерство науки и высшего образования Российской Федерации
Новосибирский государственный технический университет

Управление ресурсами в вычислительных системах
Лабораторная работа №1

Факультет: прикладной математики и информатики
Группа: ПМ-13
Студенты: Исакин Д.А.
Вострецова Е.В.
Преподаватели: Стасышин В. М.
Сивак М. А.

Новосибирск
2024

1. Условие (Вариант №10)

Разработать программу на языке C и Shell.

Программа выводит имена тех каталогов в каталоге, которые в себе не содержат каталогов. Имя каталога задано параметром программы.

2. Анализ задачи

Получаем список каталогов в заданной директории. В цикле заходим в каждый из них и определяем есть ли в этом каталоге другие каталоги или нет. Если их нет, то выводим имя данного каталога. Повторяем данную операцию рекурсивно, спускаясь по дереву каталогов.

3. Используемые программные средства

Shell — средства:

/bin/ls -A $cur_dir 2>/dev/null — выводит все директории или файлы в том числе и скрытые, ошибки игнорируются

/usr/bin/find $cur_dir -maxdepth 1 -type d -name '*' -not -path $cur_dir 2>/dev/null — поиск всех директорий в каталоге, ошибки игнорируются

>&2 — временное перенапрвление потока ошибок

echo -n — вывод в консоль флаг -n отмена вывода символа перевода строки

C — средства:

Функции:

DIR *opendir(char *dirname) – открытие папки dirname

struct dirent *readdir(DIR *dirptr) – чтение содержимого папки, где dirptr – дескриптор папки

int closedir(DIR *ptr) – закрытие папки

`void * memmove( void * destptr, const void * srcptr, size_tnum );` - Переместить блок памяти. Функция копирует num байтов из блока памяти источника, на который ссылается указатель srcptr, в блок памяти назначения, на который указывает указатель destptr

char *strcat(char * destination, const char * source) – добавляет строку source в конец строкиdestination

int strcmp(const char * string1, const char * string2) – сравнивает строки string1 и string2

int stat(const char *_file_name_, struct stat *_buf_); - Эти функции возвращают информацию об указанном файле. Для этого не

требуется иметь права доступа к файлу, хотя потребуются права поиска во всех каталогах, указанных в полном имени файла.

Структуры:

struct dirent – имя файлв и индексный дескриптор

struct stat — структура с характеристиками файла

4. Спецификация

Shell - специйикация

Программа находится в папке /home/daniil/Desktop/WorkSpace/УпРесы/lab1
Чтобы запустить скрипт, нужно использовать команду
“./shel.sh %path%”, где %path% - название директории.
В результате работы скрипта, будет выведен список всех нужных директорий.
Возможные ошибки предусмотрены и описаны при вводе команды “./shel.sh –help”.

C — специйикация

Программа находится в папке /home/daniil/Desktop/WorkSpace/УпРесы/lab1
Чтобы запустить скрипт, нужно использовать команду
“./lab1 %path%”, где %path% - название директории.
В результате работы скрипта, будет выведен список всех нужных директорий.
Возможные ошибки предусмотрены и описаны при вводе команды “./lab1 –help”.
Для сборки набрать в терминале
make all
Для очистки
make clean

5. Исходный код

Текст скрипта на языке Shell

```
#!/bin/bash

# Script input param
DIRECTORY="$1"
ARGCOUNT=$#

# Inputs validation Code
CORRECT=0        # OK!
NO_ARG_ERR=1     # No input parameters specified
DIRECTORY_ERR=2  # Incorrectly specified directory
MANY_ARG_ERR=3   # Many argument
HELP=4           # Print help

# Input parameter check function.
# Returns code that indicates the status of the check
function inputCheckStatus {

   local status=$CORRECT;

   if [ $ARGCOUNT -gt 1 ]; then
      status=$MANY_ARG_ERR
   fi

   if [ -z "$DIRECTORY" ]; then
      status=$NO_ARG_ERR
   else

      if [ "$DIRECTORY" == "-help" ]; then
         status=$HELP
      elif [ ! -d "$DIRECTORY" ]; then
         status=$DIRECTORY_ERR
      fi

   fi

   echo "$status"
}

# Function of displaying errors on the screen by their code.
# The output is carried out in stderr
function printErrorByStatus {
        err_code="$1"

        >&2 echo -n "Error: "
        case $err_code in
                $NO_ARG_ERR) >&2 echo "directory not specified!";;
                $DIRECTORY_ERR) >&2 echo "\"$DIRECTORY\" is not a directory!";;
      $MANY_ARG_ERR) >&2 echo "Many argument pass, you must pass 1 arg:
dirname";;
```

```
        esac
}


# Function of search and print directory without directorys
function dirSearch {
    cur_dir="$1"

    if [[ -z "$(/bin/ls -A $cur_dir 2>/dev/null)" ]]; then
                    echo " is empty folder: $cur_dir"
        else
        dir_list="$(/usr/bin/find $cur_dir -maxdepth 1 -type d -name '*' -not -path
$cur_dir 2>/dev/null)"

        for dir in $dir_list; do
            local nested_dir="$(/usr/bin/find $dir -maxdepth 1 -type d -name '*' -not -
path $dir 2>/dev/null)"
            if [[ -z $nested_dir ]]; then
                echo $dir
            else
                dirSearch $dir # Recursive find subdirs
            fi
        done
    fi
}


# Main programm
input_status=$(inputCheckStatus)                                    # Getting inputs status

if [ "$input_status" -eq "$CORRECT" ]; then             # Status is correct
        dirSearch "$DIRECTORY"                    # Start the main function
else
    if [ "$input_status" -eq "$HELP" ]; then
        echo "1) Enter the name of the directory for which you want to display folders
that do not contain subdirectories."
        echo "2) If the directory you entered does not exist, an appropriate error
message will be displayed."
        echo "3) If an incorrect number of parameters is entered (not equal to 1), an
error message is displayed."
    else                                                                                # Error
status
            $(printErrorByStatus "$input_status")               # Display the error
    fi
fi
```

Текст программы на языке C

```
#include <dirent.h> // Структура папки
#include <stdio.h>  // Стандартный ввод/вывод
#include <stdlib.h> // Стандартная библиотека
#include <string.h> // Строки
#include <errno.h>  // Проверка существования папки
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

#define OK 0            //
#define NO_ARG_ERR 1    // No input parameters specified
#define DIRECTORY_ERR 2 // Incorrectly specified directory
#define MANY_ARG_ERR 3  // Many argument

#define false 0
#define true 1
typedef int32_t bool;

/*
   @param: const char* str - строка
   @return: int32_t - 1 - стока "." или ".." 0 - в противном случае
*/
int32_t isSystemDir(const char *str);

/*
   @param: char *pathDir - Путь до каталога
   @return: void
   @result: Печть самого вложенного каталога
*/
void printTheInternalCatalog(char *pathDir);

/*
   @param:
      int32_t argc - количество аргументов
      char **argv - список аргументов командной строки
   @return: void
   @result: Проверка входных параметров и вывод соответсвующих сообщение
об ошибках или действий для параметра  --help
*/

/*
   @param: const char* __name - имя каталога
   @return: DIR* - дескриптор открытого каталога
   @result: Безопасное открытие и соответсвующая обработка ошибок
*/
DIR *openDir(const char *__name, bool checkIsDir);

/*
   @param: DIR *__dirp
   @return: void
```

```
   @result: Закрытие директории, если не удачно то выход с соответсвующим
кодом
*/
void closeDir(DIR *__dirp);

void checkInputArgument(int32_t argc, char **argv);

int32_t main(int32_t argc, char **argv)
{

   checkInputArgument(argc, argv);
   char *path = argv[1];

   int32_t AnyFile = 0;                    // Счетчик каких либо файлов в
директории
   DIR *dir = openDir((const char *)path, true); // Открываем переданный каталог

   struct dirent *entry;

   while (entry = readdir(dir))
   {
      if (isSystemDir(entry->d_name) == 0 && entry->d_type == 4)
      {
         char *pathDir = (char *)calloc((strlen(path) + 1 + strlen(entry->d_name) +
1), sizeof(char)); // +2 так как strlen - не учитывает \0
         memmove(pathDir, path, strlen(path));                                    //
гарантирует корректное поведение для строк
         strcat(pathDir, "/");                                                 // в конец
помещется \0
         strcat(pathDir, entry->d_name);                                          // в
конец помещется \0
         printTheInternalCatalog(pathDir);
         free(pathDir); // Переменная локальная тоже не совсем нужно
      }
      if (isSystemDir(entry->d_name) == 0)
         AnyFile++;
   }

   closeDir(dir);

   // Если ни одного подкаталога в каталоге не обнаружено печатаем имя этого
каталога
   if (AnyFile == 0)
      printf(" is empty folder!");

   return 0;
}

void checkInputArgument(int32_t argc, char **argv)
{
   if (argc > 2)
   {
      fprintf(stderr,"Many argument pass, you must pass 1 arg: dirname\n");
      exit(MANY_ARG_ERR);
```

```
    }
    else if (argc < 2)
    {
        fprintf(stderr,"directory not specified!\n");
        exit(NO_ARG_ERR);
    }
    else if (strcmp(argv[1], "-help") == 0)
    {
        printf("Instruction:\n");
        printf("1) Enter the name of the directory for which you want to display folders
that do not contain subdirectories.\n");
        printf("2) If the directory you entered does not exist, an appropriate error
message will be displayed.\n");
        printf("3) If an incorrect number of parameters is entered (not equal to 1), an
error message is displayed.\n");
        exit(OK);
    }

    return;
}

DIR *openDir(const char *__name, bool checkIsDir)
{
    bool isDir = true;
    DIR *dir = NULL;

    if (checkIsDir == true)
    {
        struct stat statBuf;
        if (stat(__name, &statBuf) == 0)
        {
            if (!S_ISDIR(statBuf.st_mode)) isDir = false;
        }
        else if (errno == ENOENT)
        {
            fprintf(stderr,"directory not specified!\n");
            exit(DIRECTORY_ERR);
        }
    }

    if (isDir == true)
    {
        dir = opendir(__name); // Открываем переданный каталог
                               // printf("try open: %s, size: %ld\n", __name, strlen(__name));
        if (errno == ENOENT)
        {
            fprintf(stderr,"Directory not found. Open error\n");
            exit(DIRECTORY_ERR);
        }
        else if (errno == ENOMEM)
        {
          fprintf(stderr,"Memory error. Open error\n");
            exit(DIRECTORY_ERR);
        }
```

```
    }
    else
    {
        fprintf(stderr, "%s is not a directory!\n", __name);
        exit(DIRECTORY_ERR);
    }

    return dir;
}

void closeDir(DIR *__dirp)
{
    // Закрытие каталога
    // printf("Close Dir: %p\n", __dirp);
    if (closedir(__dirp) != 0)
    {
        perror("Некорректное закрытие каталога");
        exit(DIRECTORY_ERR);
    }
    return;
}

int32_t isSystemDir(const char *str)
{
    int32_t res = 0;
    if (strcmp(str, ".") == 0)
        res = 1;
    if (strcmp(str, "..") == 0)
        res = 1;
    return res;
}

void printTheInternalCatalog(char *pathDir)
{
    int32_t DirCount = 0; // Счетчик каталогов в папке
    // Open dir
    DIR *dir = openDir((const char *)pathDir, false);

    struct dirent *entry;

    while (entry = readdir(dir))
    {
        if (isSystemDir(entry->d_name) == 0 && entry->d_type == 4)
        {
            DirCount++; // Увеличиваем количество счетчика каталогов в вызове
            // Формируем путь к каталогу
            char *newPath = (char *)calloc(strlen(pathDir) + 1 + strlen(entry->d_name)
+ 1, sizeof(char)); // +2 так как strlen - не учитывает \0
            memmove(newPath, pathDir, strlen(pathDir));
// гарантирует корректное поведение для строк
            strcat(newPath, "/");                                                    // в конец
помещется \0
            strcat(newPath, entry->d_name);                                          // в
конец помещется \0
```

```
            printTheInternalCatalog(newPath);
            free(newPath); // Вообще не совсем нужно так как переменная
локальная
        }
    }

    // Закрытие каталога
    closeDir(dir);

    // Если ни одного подкаталога в каталоге не обнаружено печатаем имя этого
каталога
    if (DirCount == 0)
        printf("%s\n", pathDir);
    // free(pathDir);

    return;
}
```

Содержимое файла Makefile

```
# Makefile for lab #1
all: lab1

lab1: lab1.o
	gcc -std=c11 lab1.o -o lab1

lab1.o: lab1.c
	gcc -std=c11 -c lab1.c

clean:
	rm -rf *.o lab1
```

6. Набор тестов (Тестирование C-программа)

Тест 1

Описание: пустая папка

. ├── shel.sh ├── testDir❯ ./lab1 testDir is empty folder!	❯ ./lab1 testDir is empty folder!

Тест 2

Описание: папка, содержащая файлы (директорий нет => выаод пусто)

├── shel.sh ├── testDir │ ├── file1.txt │ └── file2.txt	❯ ./lab1 testDir is empty folder!

Тест 3

Описание: бинарное дерево папок с глубиной 3

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ └── file2.txt	❯ ./lab1 testDir testDir/f21/f32 testDir/f21/f31 testDir/f22/f32 testDir/f22/f31

Тест 4

Описание: со скрытыми каталогами

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt	❯ ./lab1 testDir testDir/f21/f32 testDir/f21/f31 testDir/f22/f32 testDir/f22/f31 testDir/.shadow_dir1

Тест 5

Описание: тест ввод не дирректория

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt ├── testFile	❯ ./lab1 testFile testFile is not a directory!

Тест 6

Описание: много параметров

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt ├── testFile	❯ ./lab1 testDir res Many argument pass, you must pass 1 arg: dirname

Тест 7

Описание: отсутствие параметров

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt ├── testFile	❯ ./lab1 directory not specified!

7. Набор тестов (Тестирование Shell-скрипт)

Тест 1

Описание: пустая папка

. ├── shel.sh └── testDir	❯ ./shel.sh testDir is empty folder!

Тест 2

Описание: папка, содержащая файлы (директорий нет => выаод пусто)

├── shel.sh
└── testDir
  ├── file1.txt
  └── file2.txt
	❯ ./shel.sh testDir
	Empty folder: testDir

Тест 3

Описание: бинарное дерево папок с глубиной 3

└── testDir
  ├── f21
  │ ├── f31
  │ └── f32
  ├── f22
  │ ├── f31
  │ └── f32
  ├── file1.txt
  └── file2.txt
	❯ ./shel.sh testDir
	testDir/f21/f32
	testDir/f21/f31
	testDir/f22/f32
	testDir/f22/f31

Тест 4

Описание: со скрытыми каталогами

└── testDir
  ├── f21
  │ ├── f31
  │ └── f32
  ├── f22
  │ ├── f31
  │ └── f32
  ├── file1.txt
  ├── file2.txt
  └── .shadow_dir
  ├── file1.txt
  └── file2.txt
	❯ ./shel.sh testDir
	testDir/f21/f32
	testDir/f21/f31
	testDir/f22/f32
	testDir/f22/f31
	testDir/.shadow_dir

Тест 5

Описание: тест ввод не дирректория

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt ├── testFile	❯ ./shel.sh testFile Error: "testFile" is not a directory!

Тест 6

Описание: много параметров

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt ├── testFile	❯ ./shel.sh testDir res Error: Many argument pass, you must pass 1 arg: dirname

Тест 7

Описание: отсутствие параметров

├── testDir │ ├── f21 │ │ ├── f31 │ │ └── f32 │ ├── f22 │ │ ├── f31 │ │ └── f32 │ ├── file1.txt │ ├── file2.txt │ └── .shadow_dir │ ├── file1.txt │ └── file2.txt ├── testFile	❯ ./shel.sh Error: directory not specified!