Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа**

**по курсу «Языки и методы программирования»**

**ll Семестр**

**Задание 6**

**Обработка последовательной файловой структуры на языке Си**

|  |  |
| --- | --- |
| Студент: | Аксенов А.Е. |
| Группа: | М8О-108Б-18 |
| Преподаватель: | Поповкин А.В. |
|  |  |
| Оценка: |  |
| Дата: | .06.2019 |

**Москва, 2019 г.**

**Содержание**

1. Постановка задачи………………………………………………….3
2. Общий метод решения……………………………………………..3
3. Общие сведения о программе……………………………………...4
4. Описание логической структуры………………………………….4
5. Таблица переменных и констант…………………………………..5
6. Входные/выходные данные………………………………………..5
7. Программа на Си…………………………………………………...6
8. Демонстрация работы программы………………………………...8
9. Вывод ……………………………………………………………….9
10. Литература ……………………………………………………...9
11. **Постановка задачи**

Разработать последовательную структуру данных для представление простейшей базы данных на файлах СП Си в соответствии с заданным вариантом. Составить программу генерации внешнего нетекстового файла заданной структуры, содержащего представительный набор записей. Распечатать содержимое сгенерированного файла в виде таблицы и выполнить над ним заданное действие для 2-3 значений параметров запроса p и распечатать результат.

Вариант задания: №1

Содержимое и структуры файла:

Сведения о составе комплектующих личных ПЭВМ в студенческой группе: фамилия владельца, число и тип процессоров, Объем памяти, тип видеоконтроллера (встроенный, внешний, AGP, PCI) и объем видеопамяти, тип (SCSI/IDE, ATA/SATA), число и емкость винчестеров, количество интегрированных контроллеров и внешних (периферийных) устройств, операционная система.

**Действиe:**

Найти всех владельцев двухпроцессорных компьютеров, имеющих не более ***p*** внешних устройств.

1. **Общий метод решения**

Работа идет с четырьмя программами СИ, тремя библиотеками (содержат описание структур компьютера, процессоров и винчестеров), текстовым файлом (хранятся данные) и бинарным файлом (туда записываются данные из текстового файла, а затем считываются оттуда).

Первые три программы читают данные из текстового файла

Последняя программа записывает данные в бинарный файл, затем считывает с него и проверяет компьютеры владельцев на наличие не более ***p*** внешних устройств.

1. **Общие сведения о программе**

Программное и аппаратное обеспечения для запуска данной программы на ПК не ограничено в выборе. Операционная система семейства Unix — Linux Ubuntu 16.04.5. Язык программирования Си. Строк в программе proc.c: 42, drive.c: 97, read.c: 158, main.c: 105.

Имена файлов:

1. def.h;
2. proc.h
3. drive.h
4. read.h
5. proc.c
6. drive.c
7. read.c
8. main.c
9. in.t
10. db.dat
11. **Описание логической структуры**

read.c:

Текстовой файл in.t открывается с помощью функции fopen; затем в цикле с помощью функций fscanf, fgets происходит запись данных из текстового файла в переменные.

proc.c:

Вспомогательная программа, считывающая данные

drive.c:

Вспомогательная программа, считывающая данные

main.c:

Текстовой файл in.t открывается с помощью функции fopen; затем в файле read.c происходит запись данных из текстового файла в переменные. А потом данные с переменных переходят в файл db.dat В конце переписи данных файлы закрываются функцией fclose.

Далее db.dat cнова открывается, но уже для чтения данных в переменные. После этого файл закрывается функцией fclose. Далее происходит проверка, принадлежат ли комплектующие компьютера одной фирме. Если это так, то данные выводятся в терминал.

1. **Таблица переменных и констант**

Описание переменных структуры в proc.h:

|  |  |
| --- | --- |
| char company[50]; | Производитель процессора |
| char name[50]; | Название процессора |
| double freq; | Частота процессора |

Описание переменных структуры в drive.h:

|  |  |
| --- | --- |
| char ext; | Внешний или внутренний |
| char name[50]; | Название диска |
| char type; | Тип диска |
| size\_t size; | Объём памяти диска |

Описание переменных структуры в read.h:

|  |  |
| --- | --- |
| char owner\_name[50]; | Фамилия владельца |
| size\_t proc\_count; | Количество процессоров |
| proc \*processor; | Структура процессора |
| size\_t mem\_v; | Объём видеопамяти |
| char videoname[50]; | Название видеокарты |
| char videotype[50]; | Тип видеокарты |
| size\_t vidmem\_v; | Объём видеопамяти |
| size\_t vin\_count; | Количество дисков |
| drive \*storage; | Структура диска |
| size\_t ctrl\_dev\_count; | Количество внутренних контролеров и внешних устройств |
| char osystem[50]; | Операционная система |

В программах read.c, proc.c, drive.c main.c происходит работа со структурами read.h, drive.h и proc.c, в main.c добавились переменные для проверки фирменности комплектующих.

1. **Входные данные**
2. Для main.c:

первый аргумент - текстовый файл для чтения с данными о компьютерах;

второй аргумент - бинарный файл для записи;

1. **Выходные данные**

1. Вывод данных о компьютерах с количеством внешних устройств не более ***p***.

1. **Программа на Си**

**def.h:**

#ifndef DEF\_H  
#define DEF\_H  
  
#define LEN 50  
  
#endif

**proc.h:**

#ifndef PROC\_H  
#define PROC\_H  
  
#include <stdio.h>  
#include <stdint.h>  
#include "def.h"  
  
typedef struct {  
 char company[LEN];  
 char name[LEN];  
 double freq;  
} proc;  
  
int read\_proc (FILE \*f, proc \*pr);  
void print\_proc(proc \*pc);  
int write\_proc(FILE \*f, proc \*pr);  
  
#endif

**drive.h:**

#ifndef DRIVE\_H  
#define DRIVE\_H  
  
#include <stdio.h>  
#include "def.h"  
  
typedef struct {  
 char ext;  
 char name[LEN];  
 char type;  
 size\_t size;  
} drive;  
  
int read\_drive (FILE \*f, drive \*st);  
void print\_drive(drive \*st);  
int write\_drive(FILE \*f, drive \*);  
  
#endif

**read.h:**

#ifndef READ\_H  
#define READ\_H  
  
#include "def.h"  
#include "drive.h"  
#include "proc.h"  
  
typedef struct {  
 char owner\_name[LEN];  
 size\_t proc\_count;  
 proc \*processor;  
 size\_t mem\_v;  
 char videoname[LEN];  
 char videotype[LEN];  
 size\_t vidmem\_v;  
 size\_t vin\_count;  
 drive \*storage;  
 size\_t ctrl\_dev\_count;  
 char osystem[LEN];  
} pcdata;  
  
void init\_data (pcdata \*data);  
int read\_data (FILE \*f, pcdata \*data);  
void print\_data(pcdata \*data);  
void delet\_data(pcdata \*data);  
void bin\_wrt\_data(FILE \* f, pcdata \* data);  
int bin\_rd\_data (FILE \* f, pcdata \* data);  
  
#endif

**proc.c:**

#include "proc.h"  
  
#include <stdlib.h>  
  
void print\_proc(proc \*pr) {  
 if (pr) {  
 printf("\nProsessor from %s\n", pr->company);  
 printf("%s type\n", pr->name);  
 printf("Of %lg GHz is frequency\n", pr->freq);  
 }  
}  
  
int read\_proc(FILE \*f, proc \*pr) {  
 int r = 0;  
  
 if (pr) {  
 fscanf(f, "%s", pr->company);  
  
 fscanf(f, "%s", pr->name);  
 fscanf(f, "%lg", &pr->freq);  
 }  
  
 return r;  
}  
  
int write\_proc(FILE \*f, proc \*pr) {  
 int r = 0;  
  
 if (pr) {  
 fwrite(pr->company, LEN, 1, f);  
  
 fwrite(pr->name, LEN, 1, f);  
 fwrite(&pr->freq, sizeof(int), 1, f);  
 }  
  
 return r;  
}

**drive.c:**

#include "drive.h"  
  
#include <stdlib.h>  
  
enum {  
 ATA,  
 SATA,  
 SCSI,  
 IDE  
};  
  
void print\_drive(drive \*st) {  
 if (st) {  
 char \*t;  
 printf("\nDrive %s\n", st->ext ? "External" : "Internal");  
 printf("From %s\n", st->name);  
  
 switch (st->type) {  
 case IDE:  
 t = "IDE ";  
 break;  
 case SCSI:  
 t = "SCSI";  
 break;  
 case SATA:  
 t = "SATA";  
 break;  
 case ATA:  
 t = "ATA ";  
 break;  
 default:  
 t = "Unknown";  
 break;  
 }  
  
 printf("%s type\n", t);  
 printf("Of %lu GB is size\n", st->size);  
 }  
}  
  
int read\_drive(FILE \*f, drive \*st) {  
 int r = 0;  
 char buff[LEN];  
  
 if (st) {  
 fscanf(f, "%s", buff);  
  
 st->ext = (buff[0] == 'E');  
  
 fscanf(f, "%s", st->name);  
  
 fscanf(f, "%s", buff);  
  
 switch (buff[0] + buff[1]) {  
 case 'I' + 'D':  
 st->type = IDE;  
 break;  
 case 'A' + 'T':  
 st->type = ATA;  
 break;  
 case 'S' + 'A':  
 st->type = SATA;  
 break;  
 case 'S' + 'C':  
 st->type = SCSI;  
 break;  
 }  
 fscanf(f, "%lu", &st->size);  
 }  
  
 return r;  
}  
  
int write\_drive(FILE \*f, drive \*st) {  
 int r = 0;  
  
 if (st) {  
 fwrite(&st->ext, LEN, 1, f);  
  
 fwrite( st->name, LEN, 1, f);  
  
 fwrite(&st->type, LEN, 1, f);  
 fwrite(&st->size, sizeof(int), 1, f);  
 }  
  
 return r;  
}

**read.c:**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include "read.h"  
  
void init\_data(pcdata \*data) {  
 data->processor = NULL;  
 data->proc\_count = 0;  
 data->storage = NULL;  
 data->vin\_count = 0;  
}  
  
void endl(FILE \*f) {  
 while (fgetc(f) != '\n')  
 ;  
}  
  
/\* Fix newlines \*/  
void fxnl(char \*str) {  
 if (str && str[0])  
 str[strlen(str) - 1] = '\0';  
}  
  
int read\_data(FILE \*f, pcdata \*data) {  
 int ret = 0;  
 char buff[10];  
 if (data) {  
 size\_t i;  
  
 if (fscanf(f, "%s", buff) == 1 && buff[0] == '@') {  
 endl(f);  
 fgets (data->owner\_name, LEN, f);  
 fxnl (data->owner\_name);  
 if (fscanf(f, "%lu", &data->proc\_count) == 1) {  
 data->processor = (proc \*)malloc(sizeof(proc) \* data->proc\_count);  
 for (i = 0; i < data->proc\_count; ++i)  
 read\_proc(f, data->processor + i);  
 }  
 fscanf(f, "%lu", &data->mem\_v);  
 endl(f);  
 fgets (data->videoname, LEN, f);  
 fgets (data->videotype, LEN, f);  
 fxnl (data->videoname);  
 fxnl (data->videotype);  
 fscanf(f, "%lu", &data->vidmem\_v);  
 endl(f);  
  
 if (fscanf(f, "%lu", &data->vin\_count) == 1) {  
 data->storage = (drive \*)malloc(sizeof(drive) \* data->vin\_count);  
 for (i = 0; i < data->vin\_count; ++i)  
 read\_drive(f, data->storage + i);  
 }  
 fscanf(f, "%lu", &data->ctrl\_dev\_count);  
 endl(f);  
 fgets (data->osystem, LEN, f);  
 fxnl (data->osystem);  
 }  
 else  
 ret = -1;  
 }  
  
 return ret;  
}  
  
void print\_data(pcdata \*data) {  
 if (data) {  
 size\_t i;  
  
 printf("Surname %s\n", data->owner\_name);  
 for (i = 0; i < data->proc\_count; ++i)  
 print\_proc(data->processor + i);  
  
 printf("\n");  
 printf("Memo %ld GB\n", data->mem\_v);  
 printf("Video: \n");  
 printf("Name %s\n", data->videoname);  
 printf("Type %s\n", data->videotype);  
 printf("Memo %ld GB\n", data->vidmem\_v);  
 for (i = 0; i < data->vin\_count; ++i)  
 print\_drive(data->storage + i);  
  
 printf("\n");  
 printf("Controllers and Devices %ld\n", data->ctrl\_dev\_count);  
 printf("OC %s\n", data->osystem);  
 }  
}  
  
void delet\_data(pcdata \*data) {  
 if (data) {  
 free(data->processor);  
 data->proc\_count = 0;  
 free(data->storage);  
 data->vin\_count = 0;  
 }  
}  
  
void bin\_wrt\_data(FILE \*f, pcdata \* data){  
 if (data) {  
 size\_t i;  
  
 fwrite( data->owner\_name, LEN, 1, f);  
 fwrite(&data->proc\_count, sizeof(size\_t), 1, f);  
  
 for (i = 0; i < data->proc\_count; ++i)  
 write\_proc(f, data->processor + i);  
  
 fwrite(&data->mem\_v, sizeof(size\_t), 1, f);  
 fwrite( data->videoname, LEN, 1, f);  
 fwrite( data->videotype, LEN, 1, f);  
 fwrite(&data->vidmem\_v, sizeof(size\_t), 1, f);  
 fwrite(&data->vin\_count, sizeof(size\_t), 1, f);  
  
 for (i = 0; i < data->vin\_count; ++i)  
 write\_drive(f, data->storage + i);  
  
 fwrite(&data->ctrl\_dev\_count, sizeof(size\_t), 1, f);  
 fwrite( data->osystem, LEN, 1, f);  
 }  
}  
  
int bin\_rd\_data(FILE \*f, pcdata \*data){  
 int res = 1;  
  
 if (data) {  
 size\_t i;  
  
 res = fread(data->owner\_name, LEN, 1, f);  
 fread(&data->proc\_count, sizeof(size\_t), 1, f);  
  
 data->processor = (proc \*)malloc(data->proc\_count \* sizeof(proc));  
  
 for (i = 0; i < data->proc\_count; ++i)  
 read\_proc(f, data->processor + i);  
  
 fread(&data->mem\_v, sizeof(size\_t), 1, f);  
 fread( data->videoname, LEN, 1, f);  
 fread( data->videotype, LEN, 1, f);  
 fread(&data->vidmem\_v, sizeof(size\_t), 1, f);  
 fread(&data->vin\_count, sizeof(size\_t), 1, f);  
  
 data->storage = (drive \*)malloc(data->vin\_count \* sizeof(drive));  
  
 for (i = 0; i < data->vin\_count; ++i)  
 read\_drive(f, data->storage + i);  
  
 fread(&data->ctrl\_dev\_count, sizeof(size\_t), 1, f);  
 fread( data->osystem, LEN, 1, f);  
 }  
  
 if (res < 0)  
 res = 0;  
  
 return res;  
}

**main.c:**

#include <stdio.h>  
#include "read.h"  
  
void brandList (char \* dname);  
char \*brand\_check(pcdata \*data);  
  
void readNwrite(char \*iname, char \*oname){  
 FILE \*inf = iname ? fopen(iname, "r") : stdin;   
 int ret = 0;   
  
 printf("Input is %s\n", iname);   
 printf("Output is %s\n", oname);  
 printf("\n"); if (inf) {   
 FILE \*ouf = oname ? fopen(oname, "wb") : stdout;   
  
 if (ouf) {   
 FILE \*dbin = fopen(oname, "r+");   
 pcdata data; init\_data(&data);  
 while (!read\_data(inf, &data)) {  
 bin\_wrt\_data(ouf, &data);  
 //print\_data(&data);  
 }  
 fclose (ouf);  
 }  
 }  
}  
  
char \*brand\_check(pcdata \*data){  
 int isb = 1;  
 size\_t i;  
 char \*co = NULL;  
 char vidcompany[LEN];  
  
 sscanf(data->videoname, "%s", vidcompany);  
  
 for (i = 0; i < data->proc\_count && isb; ++i)  
 if (co) {  
 if (strcmp(co, data->processor[i].company))  
 isb = 0;  
 }  
 else  
 co = data->processor[i].company;  
  
 if (isb) {  
 if (co) {  
 if (strcmp(co, vidcompany))  
 isb = 0;  
 }  
 else  
 co = vidcompany;  
 }  
  
 if (isb) {  
 for (i = 0; i < data->vin\_count && isb; ++i)  
 if (co) {  
 if (strcmp(co, data->storage[i].name))  
 isb = 0;  
 }  
 else  
 co = data->storage[i].name;  
 }  
  
 if (!isb)  
 co = NULL;  
  
 return co;  
}  
  
void brandList(char \* dname) {  
 FILE \*dbin = fopen(dname, "r");  
  
 printf("DB: %s\n", dname);  
  
 if (dbin) {  
 pcdata data;  
 char \*brand;  
  
 printf ("Brand computers: \n");  
  
 while (bin\_rd\_data(dbin, &data)) {  
 //print\_data(&data);  
 if ((brand = brand\_check(&data))) {  
 printf("%s\n", brand);  
 print\_data(&data);  
 delet\_data(&data);  
 }  
 }  
  
 fclose (dbin);  
 }  
}  
  
int main(int argc, char \*\*argv) {  
 if (argc == 3){  
 readNwrite(argv[1], argv[2]);  
 }  
 else  
 printf("Insufficient number of arguments\n");  
 brandList(argv[2]);  
}

1. **Демонстрация работы программы**

user@PSB133S01ZFH:~/course6$ gcc \*.c -o main

user@PSB133S01ZFH:~/course6$ ./main in.t db.dat

Input is in.t

Output is db.dat

DB: db.dat

Brand computers:

Surname Walter

Prosessor from AMD

Ryzen-5-2600 type

Of 2.8 GHz is frequency

Memo 2 GB

Video:

Name AMD Radeon R5 435

Type External

Memo 3 GB

Drive Internal

From AMD

IDE type

Of 4 GB is size

Controllers and Devices 4

OC Linux Ubuntu 16.04.5

1. **Вывод**

Данное задание курсовой работы позволяет:

1. Научиться работать со структурами, задавать их в библиотеках;

2. Работать с файлами (запись и чтение бинарных файлов);

3. Создавать базы данных.

Эти навыки внесли большой вклад в мой опыт прикладного программирования.

1. **Литература**
2. Методические указания к выполнению курсовых работ. Зайцев В. Е.

<https://learnc.info/c/text_files.html>

1. Google Диск — О файлах и функциях

https://drive.google.com/drive/folders/0B65wxlnb9Q\_0WnJYRWoxdUhoMUk

1. Пособие по Си - strtok

http://www.cplusplus.com/reference/cstring/strtok/?kw=strtok