**Московский авиационный институт**

(национальный исследовательский университет)

**Факультет № 8 «Информационные технологии и прикладная математика»**

**Кафедра 806 «Вычислительная математика и программирование»**

КУРСОВОЙ ПРОЕКТ

по дисциплине «Языки и методы программирования»

Задание 6. Обработка последовательной

файловой структуры на языке Си.

|  |  |
| --- | --- |
| Студент: | Петров И. О. |
| Группа: | М8О-106Б-21 |
| Преподаватель: | Дубинин А.В. |
| Подпись: |  |
| Оценка: |  |

Москва, 2022

# Постановка задачи

Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП Си. Составить программу для генерации внешнего нетекстового файла заданной структуры, содержащего представительный набор записей (15-20). Распечатать содержимое сгенерированного файла в виде таблицы и выполнить над ним заданное действие для 2-3 значений параметров запроса, распечатать результат.

|  |  |  |  |
| --- | --- | --- | --- |
| **Вариант 6** | | | |
| Содержимое | Сведения о составе комплектующих | | |
| и | личных ПЭВМ в студенческой | | |
| структура файла | группе: фамилия владельца, число и | | |
|  | тип процессоров, объём памяти, тип | | |
|  | видеоконтроллера (встроенный, | | |
|  | внешний, AGP, PCI) и объём | | |
|  | видеопамяти, тип (SCSI/IDE, | | |
|  | ATA/SATA), число и ёмкость | | |
|  | винчестеров, количество | | |
|  | интегрированных контроллеров и | | |
|  | внешних (периферийных) устройств, | | |
|  | операционная система. | | |
| Задание | Распечатать сведения  компьютерах-серверах станциях. | и | обо всех рабочих |

# Теоретическая часть

Открытие файла осуществляется с помощью функции , которая возвращает указатель на структуру типа FILE, который можно использовать для последующих операций с файлом:

* FILE *file\_name* – создаём файл с названием “*file\_name*”;
* *file\_name* = (file\_path, format) – открываем файл в нужном формате, здесь:
  + file\_path – путь до открываемого файла или его имя, если находимся в нужной директории;
  + format – способ доступа к файлу (флаг);

|  |  |
| --- | --- |
| Форматы | |
| “r” | Режим открытия файла для чтения. Файл должен существовать. |
| “w” | Режим создания пустого файла для записи. Если файл с таким именем уже существует его содержимое стирается, и файл  рассматривается как новый пустой файл. |
| “a” | Дописать в файл. Операция добавления данных в конец файла. Файл  создается, если он не существует. |
| “r+” | Режим открытия фала для обновления чтения и записи. Этот файл  должен существовать. |
| “w+” | Создаёт пустой файл для чтения и записи. Если файл с таким именем  уже существует его содержимое стирается, и файл рассматривается как новый пустой файл. |
| “a+” | Открыть файл для чтения и добавления данных. Все операции записи выполняются в конец файла, защищая предыдущее содержания файла от случайного изменения. Файл создается, если он не  существует. |

Выше указанные спецификаторы режима доступа к файлу используются только в текстовых файлах. Для того, чтобы открыть двоичный файл, символ b должен быть включен в режим доступа. Этот дополнительный

символ b может быть добавлен в конец строки, что даёт следующие режимы доступа к бинарным файлам: rb, wb, ab, r+b, w+b, a+b или может быть вставлен между буквой и знаком +, в случае со смешанными режимами: rb+, wb+, ab+.

* *fclose(file\_name)* – закрывает поток, открытый при помощи функции

*fopen().*

* *fwrite(const void \*p, size, n, FILE \*file)* – функция записывает n элементов длиной size байт из буфера, заданного указателем р, в поток file. Возвращает число записанных элементов.
* *fread(void \*p, size, n, FILE \*file)* – функция считывает n элементов длиной size байт в буфер, заданный указателем p, из потока file. Функция возвращает количество прочитанных элементов, которое может быть меньше n, если при чтении произошла ошибка или встретился конец файла.
* *feof(file)* – функция проверяет, достигнут ли конец потока file. Возвращается ноль, если конец файла НЕ был достигнут.

# Серверы и рабочие станции

Серверы и рабочие станции от персональных компьютеров отличаются в основном более высокой мощностью и производительостью.

Рабочая станция часто используется для решения сложных прикладных, с чем не справится обычный компьютер, также она подключена к локальной сети.

Сервер может выполнять разные функции, хранить информацию, обрабатывать многочисленные запросы извне, обслуживать офисную технику и многое другое.

По комплектации сервер и рабочую станцию отличают типы процессоров (в них больше ядер, они надёжнее), для них устанавливается своя операционная система. Может быть больше одного процессора. В зависимости от назначения к ним может быть подключено много периферийных устройств, у

них может быть много жёстких или твёрдотельных дисков с большой

ёмкостью.

typedef struct Computer { char surname[str\_len]; int processors;

char proc\_type[str\_len]; int memory\_size;

# Основная часть Структура

char video\_controller\_type[str\_len]; int vid\_mem\_size;

char HMDD\_type[str\_len]; int HMDD\_number;

int HMDD\_size;

int controllers\_number; int devices\_number; char OS[str\_len];

} comp;

# Описание функций

|  |  |
| --- | --- |
| void create\_new() { char name[str\_len]; FILE \*thisfile;  printf("Please, enter a name of the file you want to create or renew:**\n**");  scanf("%s", name);  thisfile = fopen(name, "rb"); if (thisfile == NULL) {  thisfile = fopen(name, "wb");  printf("File %s was successfully created**\n**", name); fclose(thisfile);  } else {  thisfile = fopen(name, "wb");  printf("File %s has become empty**\n**", name); fclose(thisfile);  }  } | Функция создаёт бинарный файл, если он до этого не существовал.  Если файл  существовал, предыдущие записи в нём удалятся. |

|  |  |
| --- | --- |
| void add() { FILE \*thisfile;  char name[str\_len]; int num;  comp pc;  printf("Enter a name of the file where you want add tuple:**\n**");  scanf("%s", name);  printf("How many tuples you want to add?**\n**"); scanf("%d", &num);  thisfile = fopen(name, "ab"); for (int i = 0; i < num; i++) {  printf("Enter the surname of pc's owner, press underlines instead of whitespaces:**\n**");  scanf("%s", pc.surname);  printf("Enter the number of processors:**\n**"); scanf("%d", &pc.processors);  printf("Enter type of processors, press underlines instead of whitespaces**\n**");  scanf("%s", pc.proc\_type);  printf("Enter size of memory**\n**"); scanf("%d", &pc.memory\_size);  printf("Enter type of video controller, press underlines instead of whitespaces**\n**");  scanf("%s", pc.video\_controller\_type);  printf("Enter size of video memory**\n**"); scanf("%d", &pc.vid\_mem\_size);  printf("Enter type of hard disks, press underlines instead of whitespaces**\n**");  scanf("%s", pc.HMDD\_type);  printf("Enter number of hard disks**\n**"); scanf("%d", &pc.HMDD\_number);  printf("Enter size of hard disks**\n**"); scanf("%d", &pc.HMDD\_size);  printf("Enter number of controllers**\n**"); | Функция  добавления записей в конец файла.  Предыдущие не изменяются.  Происходит  считывание из буфера, дальше данные записываются в бинарный файл. |

|  |  |
| --- | --- |
| scanf("%d", &pc.controllers\_number);  printf("Enter number of devices**\n**"); scanf("%d", &pc.devices\_number);  printf("Enter OS, press underlines instead of whitespaces:**\n**");  scanf("%s", pc.OS);  fwrite(&pc, sizeof(comp), 1, thisfile); printf("Tuple was added**\n**");  }  fclose(thisfile);  } |  |
| void delite() { FILE \*thisfile;  char name[str\_len];  printf("Enter a name of the removing file**\n**"); scanf("%s", name);  thisfile = fopen(name, "rb"); if (!thisfile) {  printf("This file doesn't exist in the directory**\n**");  } else {  remove(name); fclose(thisfile);  printf("File %s was delited**\n**", name);  }  } | Функция удаляет файл, если он до этого существовал. |
| void help() {  printf("Enter '-c' to create new file**\n**Enter '-a' to add tuple to new or existing file**\n**");  printf("Enter '-h' for help**\n**Enter '-r' to remove file**\n**Enter '-p' to print file**\n**Enter '-f' to use function**\n**");  } | Подсказка, какие флаги нужно ввести в  командную строку |

void print() { comp pc; FILE \*thisfile;

char name[str\_len];

printf("Enter a name of the file you want to print**\n**"); scanf("%s", name);

thisfile = fopen(name, "rb"); if (thisfile == NULL) {

printf("This file doesn't exist in the directory**\n**"); return;

|  |
| --- |
| } else {  printf("  **\n**");  printf("| Surname |Number of CPUs| CPUs' type  |Memory|VideoControllerType|Video memory| HDD's type |HDD's Number|HDD's size|Contr. numb.|Devices| OS |**\n**");  while (fread(&pc, sizeof(comp), 1, thisfile) && feof(thisfile) == 0) {  fprintf(stdout, "|%-15s|%-14d|%-16s|%-6d|%-19s|%-12d|%-12s|%-12d|%-10d|%-12d|%- 7d|%-14s|**\n**",  pc.surname, pc.processors, pc.proc\_type, pc.memory\_size, pc.video\_controller\_type, pc.vid\_mem\_size, pc.HMDD\_type, pc.HMDD\_number, pc.HMDD\_size, pc.controllers\_number, pc.devices\_number, pc.OS);  }  fclose(thisfile);  printf("  **\n**");  }  } |
| Функция открывает файл на чтение, выводит по строкам. |
| void func() { comp pc; FILE \*thisfile;  char name[str\_len];  char type1[] = "AMD\_Opteron"; char type2[] = "Intel\_Xeon"; char type3[] = "Intel\_Atom"; char type4[] = "Intel\_Itanium"; char os1[] = "Free\_BSD";  char os2[] = "Windows\_Server"; char os3[] = "CentOS";  char os4[] = "Ubuntu\_Server"; char os5[] = "Gentoo";  char os6[] = "Fedora";  char os7[] = "OS\_X\_Server"; char os8[] = "OpenBSD"; char os9[] = "Solaris";  char os10[] = "HO-UX"; char os11[] = "AIX";  printf("Enter a name of the file where you want to use function:**\n**"); scanf("%s", name);  thisfile = fopen(name, "rb"); if (thisfile == NULL)  printf("This file doesn't exist in the directory**\n**"); else {  printf("  **\n**");  printf("| Surname |Number of CPUs| CPUs' type |

|  |
| --- |
| |Memory|VideoControllerType|Video memory| HDD's type |HDD's Number|HDD's size|Contr. numb.|Devices| OS |**\n**");  while (fread(&pc, sizeof(comp), 1, thisfile) && feof(thisfile) == 0) { if  ((((strcmp(pc.proc\_type, type1) == 0) || (strcmp(pc.proc\_type, type2) == 0) || (strcmp(pc.proc\_type, type3) == 0) || (strcmp(pc.proc\_type, type4) == 0)) &&  ((strcmp(pc.OS, os1) == 0) || (strcmp(pc.OS, os2) == 0) || (strcmp(pc.OS, os3) == 0) || (strcmp(pc.OS, os4) == 0) ||  (strcmp(pc.OS, os5) == 0) || (strcmp(pc.OS, os6) == 0) || (strcmp(pc.OS, os7) == 0) || (strcmp(pc.OS, os8) == 0) || (strcmp(pc.OS, os9) == 0) || (strcmp(pc.OS, os10) == 0) || (strcmp(pc.OS, os11) == 0))) &&  ((pc.processors > 1) || (pc.memory\_size >= 16) || (pc.HMDD\_number > 1) || (pc.HMDD\_size >= 1024) || (pc.devices\_number > 15))) {  fprintf(stdout, "|%-15s|%-14d|%-16s|%-6d|%-19s|%-12d|%-12s|%-12d|%-10d|%- 12d|%-7d|%-14s|**\n**",  pc.surname, pc.processors, pc.proc\_type, pc.memory\_size, pc.video\_controller\_type, pc.vid\_mem\_size, pc.HMDD\_type, pc.HMDD\_number, pc.HMDD\_size, pc.controllers\_number, pc.devices\_number, pc.OS);  }  }  fclose(thisfile);  printf("  **\n**");  }  } |
| Функция проверяет файл на существование, а потом проверяет  принадлежность к серверу или рабочей станции по ранее указанным параметрам. |

**Интерфейс**

int main(int argc, char const \*argv[])

{

if (argc != 2) {

printf("Please, enter flag**\n**"); help();

exit(1);

}

else if (!strcmp(argv[1], "-c")) { create\_new();

}

else if (!strcmp(argv[1], "-a")) { add();

}

else if (!strcmp(argv[1], "-p")) { print();

}

else if (!strcmp(argv[1], "-r")) {

delite();

}

else if (!strcmp(argv[1], "-f")) { func();

}

else if (!strcmp(argv[1], "-h")) { help();

}

else {

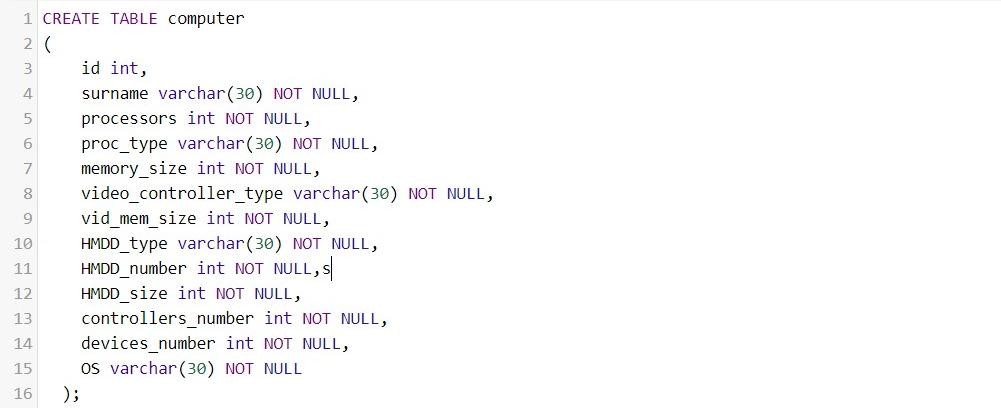
help();

}

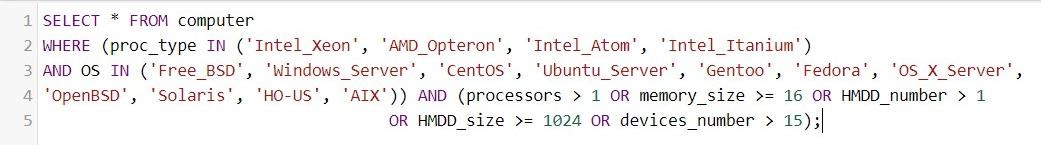
return 0;

}

SQL таблица

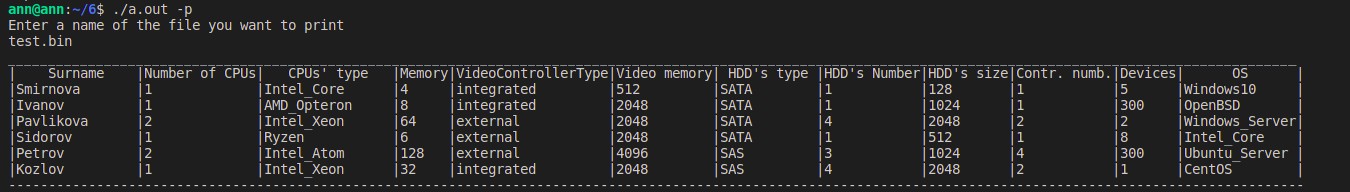


SQL запрос

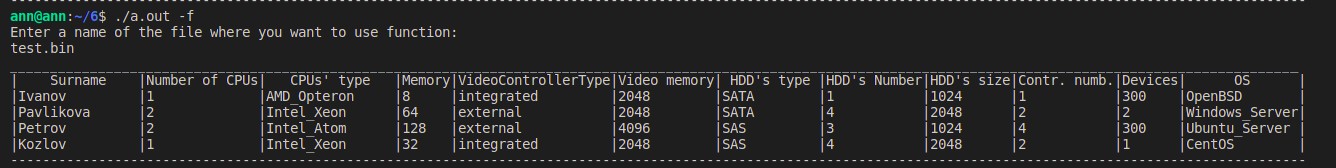


# Протокол

Созданный датасет



Выполнение запроса



# Заключение

В задании №6 курсовой работы я научился работать с файлами и обрабатывать файловые структуры. Я построил простейшую СУБД на файлах с возможностью создания таблицы, добавления в неё записей, удаления таблицы, распечатки таблицы и вывода данных о серверах и рабочих станциях. Полученные знания работы с файлами и структурами, а также основы работы с базами данных обязательно пригодятся мне в будущем.

# Список литературы

1. <http://cppstudio.com/post/1253/>
2. <https://www.avk-company.ru/articles/11/>
3. [http://sqlfiddle.com/#!9/f23fe0/3](http://sqlfiddle.com/%23!9/f23fe0/3)
4. <https://itvdn.com/ru/blog/article/m-sql#5>
5. <https://sql-language.ru/select-where.html>