

Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Е.В. Красавин, Е.А. Черепков

РАЗРАБОТКА ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ ПОД FREEBSD

Методические указания к домашней работе
по дисциплине «Операционные системы»

Калуга – 2019

УДК 004.62
ББК 32.972.1
К78

Методические указания составлены в соответствии с учебным планом КФ МГТУ им. Н.Э. Баумана по направлению подготовки 09.03.04 «Программная инженерия» кафедры «Программного обеспечения ЭВМ, информационные технологии».

Методические указания рассмотрены и одобрены:


- Кафедрой «Программного обеспечения ЭВМ, информационных технологий» (ИУ4-КФ) протокол № 51.4/04 от 20 ноября 2019 г.

Зав. кафедрой ИУ4-КФ

 к.т.н., доцент Ю.Е. Гагарин

- Методической комиссией факультета ИУ-КФ протокол № 5 от «25» ноября 2019 г.


Председатель методической
комиссии факультета ИУ-КФ

 к.т.н., доцент М.Ю. Адкин

- Методической комиссией

КФ МГТУ им.Н.Э. Баумана протокол № 3 от «3» 12 2019 г.

Председатель методической
комиссии КФ МГТУ им.Н.Э. Баумана

 д.э.н., профессор О.Л. Перерва



Рецензент:

к.т.н., доцент кафедры ИУ6-КФ

 А.Б. Лачихина

Авторы

к.т.н., доцент кафедры ИУ4-КФ
ассистент кафедры ИУ4-КФ

 Е.В. Красавин
 Е.А. Черепков

Аннотация

Методические указания к домашней работе по курсу «Операционные системы» включают основные сведения о программировании в операционной системе FreeBSD. Содержат описание принципов разработки, отладки и запуска программ в ОС FreeBSD.

Предназначены для студентов 3-го курса бакалавриата КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

© Калужский филиал МГТУ им. Н.Э. Баумана, 2019 г.
© Е.В. Красавин, Е.А. Черепков, 2019 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	4
ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ.....	5
КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ	6
РАЗРАБОТКА ВО FREEBSD	8
СБОРКА И ЗАПУСК ПРОГРАММЫ.....	11
ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ	15
КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ	22
ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ	23
ОСНОВНАЯ ЛИТЕРАТУРА.....	24
ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА	24

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой дисциплины «Операционные системы» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Методические указания, ориентированные на студентов 3-го курса направления подготовки 09.03.04 «Программная инженерия», содержат основные сведения о программировании в операционной системе FreeBSD.

Методические указания составлены для ознакомления студентов с основополагающими понятиями и принципами написания программ в UNIX. Для выполнения домашней работы студенту необходимы минимальные навыки программирования.

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения домашней работы является получение практических навыков по написанию и отладке программ.

Основными задачами выполнения домашней работы являются:

1. Научиться разрабатывать, компилировать и отлаживать программы под ОС FreeBSD.

Результатами работы являются:

1. Исполняемый файл, содержащий программу, разработанную согласно варианту;
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

FreeBSD – это операционная система, подобная UNIX, находящаяся в открытом доступе для всех пользователей. Она широко применяется в компаниях и провайдерах услуг Интернета, во встроенных устройствах и в любом другом месте, где важна надежность. Операционная система FreeBSD – это результат непрерывного, в течение более тридцати лет, процесса разработки, исследований и доработки. FreeBSD основана на 4.4BSD-Lite и предназначена для компьютеров Intel (x86 и Itanium®), AMD64, Alpha™ и Sun UltraSPARC®. Ведется работа по портированию и на другие архитектуры.

FreeBSD является самой настоящей операционной системой, поставляемой со всеми исходными текстами.

Несомненно то, что использование так называемых открытых систем является требованием организации вычислительного процесса в современных условиях. Однако не существует коммерческого продукта, более открытого, чем то, которое включает полные исходные тексты всей операционной системы, включая ядро и все системные демоны, программы и утилиты. Вы можете модифицировать любую часть FreeBSD, если это требуется вам, вашей организации или фирме.

Так как FreeBSD основывается на 4.4BSD, стандартной промышленной версии UNIX, компилировать и запускать программы достаточно легко. FreeBSD также включает большую коллекцию пакетов и коллекцию портов, что обеспечивает лёгкость компиляции и установки уже откомпилированного программного обеспечения на вашей рабочей машине или сервере. Имеется также всё увеличивающееся количество коммерческих приложений, написанных для FreeBSD.

Этапы разработки программы

- *Спецификация* (определение требований к программе): На данном этапе происходит подробное описание исходных данных, осуществляется формулировка требований к

получаемому результату, рассматриваются всевозможные поведения программы при возникновении особых случаев (к примеру, если ввели неверные данные), происходит разработка диалоговых окон, которые обеспечат взаимодействие пользователя и самой программы.

- *Разработка алгоритма:* На этом этапе программист определяет последовательность необходимых действий, которые впоследствии нужно выполнить для получения желаемого результата. Результат данного этапа разработки программы — подробное словесное описание алгоритма программы, либо блок-схема алгоритма.
- *Кодирование:* После проведения [спецификации](#) и составления [алгоритма](#) решения, используемый алгоритм в итоге будет записан на необходимом языке программирования (Pascal, Delphi, C++ и др.). Результатом этапа кодирования является готовая программа.
- *Отладка:* На данном этапе программист занимается отладкой программы, то есть поиском и устранением ошибок. Последние делятся на две группы: алгоритмические и синтаксические (ошибки в тексте исходной программы). Из этих двух групп ошибок наиболее легко устранить синтаксические ошибки, тогда как алгоритмические ошибки определить достаточно трудно. Этап отладки считается законченным лишь тогда, когда исходная программа работает корректно и правильно при одном или двух наборах первичных данных.
- *Тестирование:* Тестирование программы очень важно, поскольку в большинстве случаев программисты создают программы не для личного применения, а чтоб их программой пользовались другие. На этапе тестирования разработчик проверяет поведение программы при большом числе наборов входных данных, как верных, так и специально подобранных неверных.

РАЗРАБОТКА НА ПЛАТФОРМЕ FREEBSD

FreeBSD обеспечивает прекрасную среду для ведения разработок. С базовой системой поставляются компиляторы C, C++, языка Fortran и ассемблера, не говоря уж об интерпретаторе Perl и наборе классических утилит Unix, таких, как sed и awk. Если этого недостаточно, то в Коллекции портов имеется еще больше компиляторов и интерпретаторов. FreeBSD практически полностью соответствует таким стандартам, как POSIX и ANSI C, а также собственной идеологии BSD, так что возможно писать приложения, которые без особых или с незначительными модификациями будут компилироваться и работать на широком спектре платформ.

Программа представляет собой набор инструкций, которые предписывают компьютеру выполнить различные действия; иногда действие, которое нужно выполнить, зависит от того, что случилось в результате выполнения предыдущей операции. Этот раздел дает вам обзор двух основных способов, которыми вы можете выдавать инструкции, или «команды», как их обычно называют. В одном способе используется [интерпретатор](#), а в другом [компилятор](#).

Интерпретаторы

В случае использования интерпретатора язык представляет собой оболочку, в которой вы набираете команды, а оболочка их выполняет. Для более сложных программ вы можете набрать команды в файле и использовать интерпретатор чтобы загрузить файл и выполнить команды из него. Если что-то идет не так, интерпретатор передает управление отладчику, с помощью которого можно решить проблему.

Плюсом этого подхода является возможность сразу увидеть результаты выполнения команд, а ошибки могут быть быстро исправлены. Недостаток является то, что для запуска программ на другом компьютере у него должен быть точно такой же интерпретатор. С точки зрения производительности, интерпретаторы могут

использовать много памяти, и, как правило, генерируемый ими код не так эффективен, как код, генерируемый компиляторами.

Использование интерпретационных языков является хорошим началом. Такой способ работы обычно используется с такими языками как Lisp, Smalltalk, Perl и Basic. Можно также отметить, что оболочка Unix (sh, csh) является интерпретатором, и многие пользователи пишут «скрипты» для облегчения выполнения различных частых задач на своих машинах. На самом деле, частью философии Unix было предоставление множества маленьких вспомогательных программ, которые можно связать вместе в «скриптах» оболочки для выполнения полезных действий.

Ниже приводится список интерпретаторов, которые доступны в виде пакетов FreeBSD, с кратким обсуждением некоторых самых популярных интерпретационных языков. Для работы пакета, как правило, требуется полностью функциональная система FreeBSD версии 2.1.0 и выше.

Интерпретаторы, существующие для FreeBSD:

- BASIC
- Lisp
- Perl
- Scheme
- Icon
- Logo
- Python

Компиляторы

Компиляторы достаточно сильно отличаются от интерпретаторов. Сначала код записывается в файл (или файлы), с помощью редактора. Затем необходимо запустить компилятор и проверить работает ли программа. Если она не компилируется, необходимо вернуться к редактированию и исправить ошибки; если код компилируется, и программа выполняется не успешно, можно запустить ее в отладчике для пошаговой проверки.

Этот процесс позволяет выполнять множество действий, которые затруднительно или невозможно сделать в интерпретаторе, к примеру, написать код, тесно взаимодействующий с операционной системой или даже написать собственную операционную систему. Это также полезно, если нужно написать очень эффективный код, так как компилятор может затратить время на оптимизацию кода, что может оказаться неудобным для интерпретатора. Кроме того, запустить программу на другом компьютере, написанной для компилятора, обычно проще, чем для интерпретатора, полагая, что используется та же операционная система.

В число компиляционных языков входят Pascal, C и C++. C и C++ являются более сложными языками со множеством возможностей, и больше подходят для опытных программистов; Pascal, с другой стороны, разрабатывался как язык для обучения, поэтому достаточно прост в освоении. В базовую поставку системы FreeBSD поддержка Pascal не включена, однако в коллекции портов имеется компилятор GNU Pascal Compiler (gpc).

Так как цикл редактирование-компиляция-запуск-отладка неудобен при использовании отдельных программ, многие производители коммерческих компиляторов предлагают интегрированные среды разработки (сокращённо IDE - Integrated Development Environment). В базовую поставку FreeBSD IDE не входит, однако в дереве портов имеется evel/kdevelop, и в этих целях обычно используют Emacs.

СБОРКА И ЗАПУСК ПРОГРАММЫ

Сборка

Для [компиляции](#) потребуется программа gcc (cc, c++, g++).

```
% gcc foobar.cpp
```

Имеется огромное количество параметров для gcc, все они описаны на справочной странице. Вот несколько из самых важных, с примерами их использования.

-o

Имя выходного файла. Если вы не используете этот параметр, gcc сгенерирует выполнимый файл с именем a.out.

```
% gcc foobar.c           исполняемый файл называется a.out
```

```
% gcc -o foobar foobar.c исполняемый файл называется foobar
```

-c

Выполнить только компиляцию файла, без компоновки. Полезно для программ, когда вы хотите просто проверить синтаксис, или при использовании Makefile.

```
% gcc -c foobar.c
```

В результате будет сгенерирован объектный файл (не исполнимый файл) с именем foobar.o. Он может быть скомпонован с другими объектными файлами для получения исполнимого файла.

-g

Создать отладочную версию исполнимого файла. Этот параметр указывает компилятору поместить в выполнимый файл информацию о том, какая строка какого файла с исходным текстом какому вызову

функции соответствует. Отладчик может использовать эту информацию для вывода исходного кода при пошаговом выполнении программы, что очень полезно; минусом является то, что вся эта дополнительная информация увеличивает объём, занимаемый программой. Обычно компиляция с параметром `-g` происходит при работе над программой, а затем, при компиляции финальной версии, без параметра `-g`.

```
% gcc -g foobar.c
```

При этом будет сгенерирована отладочная версия программы.

`-O`

Создать оптимизированную версию исполняемого файла. Компилятор прибегает к различным ухищрениям для того, чтобы сгенерировать исполняемый файл, обработка которого быстрее, чем обычно. После опции `-O` можно добавить число, указывающее качество оптимизации, но использование этого параметра не защищено от ошибок оптимизации компилятора. Например, известно, что версия компилятора `cc`, поставляемая с FreeBSD версии 2.1.0, при некоторых условиях генерирует неверный код при использовании опции `-O2`.

Обычно оптимизацию используют при компиляции финальной версии.

```
% gcc -O -o foobar foobar.c
```

По этой команде будет создана оптимизированная версия программы `foobar`.

Отладка

Отладчик, поставляемый с FreeBSD, называется gdb (GNU debugger). Он запускается при исполнении команды

```
% gdb progname
```

Обычно его запуск производится из Emacs. Так же можно сделать это так:

```
M-x gdb RET progname RET
```

Использование отладчика позволяет запустить программу в пошаговом режиме построчно, что позволит изучить значения переменных, изменить их, исполнить программу до определенного места, а затем остановиться, и так далее. Так же существует возможность подключиться к уже работающей программе или загрузить файл дампа для изучения причины ошибки в программе. Возможно даже отладить ядро, хотя этот процесс является более сложным, чем отладка пользовательских приложений.

В [gdb](#) имеется достаточно хорошая встроенная система помощи, а также набор info-страниц.

Чтобы получить максимальный результат от использования gdb, нужно откомпилировать программу с параметром -g. Отладчик будет работать и без этой опции, но будет показано только название текущей функции, а не ее исходный код.

Если на экране при запуске gdb будет выведено сообщение:

```
... (no debug symbols found) ...
```

Необходимо определить была ли произведена компиляция программы с опцией -g.

В gdb нужно ввести команду `break main`. Это укажет отладчику пропустить предварительный подготовительный код программы и начать сразу с необходимого участка кода. Теперь необходимо ввести

команду `gdb` для запуска программы – она начнет выполняться с подготовительного кода и затем будет остановлена отладчиком при вызове `main()`.

Теперь можно выполнять программу построчно по шагам, нажимая `n`. Если в программе используется вызов функции, то можно перейти в нее при нажатии клавиши `s`. Для того чтобы вернуться из пошагового выполнения функции необходимо нажать клавишу `f`. Можно также использовать команды `up` и `down` для просмотра вызывающей подпрограммы.

`gdb` выводит стек вызовов всякий раз, когда происходит вызов или возвращение из функции, даже если используются команды `up` и `down` для продвижения по стеку. При этом выводится имя функции и значения ее аргументов, что помогает отслеживать, где находится курсор и что происходит. (Стек является областью, где программа сохраняет информацию о передаваемых функциям аргументах и о том, куда нужно перейти после возврата из функции).

Запуск

Для запуска программы файлу необходимо установить права на исполнение

```
% chmod +x progname
```

После этого, программу можно запустить

```
% ./progname
```

ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ

1. Изучить краткий теоретический материал и составить алгоритм работы программы, согласно варианту.
2. Создать текстовый файл и написать в нем исходный код программы на языке C/C++, обеспечивающий требуемый функционал.
3. Скомпилировать и запустить программу. Протестировать работоспособность и наличие требуемой функциональности программы.

Варианты заданий:

Вариант 1

Написать комплекс программ для нахождения определенного интеграла методом трапеций. Диапазон задает пользователь с клавиатуры, функции предопределены в программе и выбираются при помощи меню. Результат заносится в файл в виде: функция, диапазон и результат. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 2

Написать комплекс программ для нахождения определенного интеграла методом прямоугольников. Диапазон задает пользователь с клавиатуры, функции предопределены в программе и выбираются при помощи меню. Результат заносится в файл в виде: функция, диапазон и результат. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 3

Написать комплекс программ для решения квадратного уравнения. Пользователь либо вводит значения с клавиатуры, либо значения считываются из файла. Результат выводится на экран, заносится в файл

и файл отправляется с клиентского приложения на сервер (протокол TCP). В программе использовать меню.

Вариант 4

Написать комплекс программ для реализации простого калькулятора. В программе использовать меню. Результат действий занести в файл и отправить с клиентского приложения на сервер (протокол TCP).

Вариант 5

Написать комплекс программ для перевода чисел из одной системы счисления в другую. Для выбора системы счисления использовать меню. Результат перевода заносится в файл и отправляется с клиентского приложения на сервер (протокол UDP).

Вариант 6

Написать комплекс программ для разложения числа на простые множители. Число и результат операции сохранить в файл и отправить с клиентского приложения на сервер (протокол UDP). Число пользователь вводит либо с клавиатуры, либо считывает из файла.

Вариант 7

Написать комплекс программ для работы с массивами. Предусмотреть сортировку, поиск и создание массива. Каждое действие над массивом сохранять в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP). В программе использовать меню.

Вариант 8

Написать комплекс программ для преобразования десятичных чисел в римскую систему счисления. Результат выдать на экран и сохранить в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 9

Написать комплекс программ для подсчета числа дней между двумя датами. Диапазон дат задается пользователем либо с клавиатуры, либо находится в файле. Реализовать меню для программы и сохранить входные и выходные данные в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 10

Создать базу данных с информацией о пользователях: идентификатор, имя пользователя, имя учетной записи, группа. Используя меню добавить пользователя в базу данных, редактировать информацию о нем, удалить запись в базе данных и обнулить базу данных, оставив только заголовок таблицы. Синхронизировать клиентскую базу данных с сервером (протокол TCP).

Вариант 11

Написать комплекс программ, реализующий калькулятор с основными арифметическими функциями и функциями сравнения. Для реализации использовать меню. Результаты вычислений занести в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 12

Написать комплекс программ, реализующий работу со строками. Пользователь поочередно вводит с клавиатуры 2 строки. Их необходимо вывести в обратном порядке и объединить. Результат операции занести в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 13

Написать комплекс программ, реализующий частотный словарь. Текст для анализа загружается из файла или вводится с клавиатуры. Режим ввода анализируемого текста выбирается в меню. Результат

частотного анализа занести в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 14

Написать комплекс программ, реализующий работу со строками. Дана строка текста длиной не более 80 символов, состоящая из слов, разделенных пробелом, в конце – точка. Написать программу, которая определяет номера слов, в которых содержится более трех символов «А». Строка считывается из файла или вводится с клавиатуры. Выбор режима ввода строки определяется в меню. Результат работы программы сохраняется в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 15

Написать комплекс программ, реализующий работу с записями. Необходимо составить список студентов группы из 15 человек, содержащий информацию: фамилию, год поступления, итоги последней сессии (3 экзамена), используя тип - запись. Вычислить средний балл по каждому предмету и вывести список студентов, имеющих оценки выше среднего балла. Список студентов заносится в файл и при вычислении среднего балла, данный считывается из него. Результат работы программы сохраняется в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 16

Написать комплекс программ, реализующий работу с записями. Составить список владельцев автомобилей (не более 10), указав фамилию, марку (модель) автомобиля и его номер. Напечатать список владельцев автомобилей данной марки и их номера, а также количество автомобилей каждой модели. Результат работы программы сохраняется в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 17

Написать комплекс программ, реализующий работу с множествами. Дан текст из строчных латинских букв, за которыми следует точка. Напечатать все буквы, входящие в текст не менее двух раз. Результат работы программы сохраняется в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 18

Написать комплекс программ, реализующий работу с множествами. Необходимо найти все целые числа, лежащие в диапазоне от 1 до 1600, которые представимы в виде $x^2 + y^2$, где x и y – целые числа ($m, n \geq 0$). Результат работы программы сохраняется в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 19

Написать комплекс программ для нахождения определенного интеграла по формуле Симпсона. Диапазон задает пользователь с клавиатуры, функции предопределены в программе и выбираются при помощи меню. Результат заносится в файл в виде: функция, диапазон и результат. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 20

Написать комплекс программ для решения квадратного уравнения. Пользователь либо вводит значения с клавиатуры, либо значения считываются из файла. Результат выводится на экран, заносится в файл и файл отправляется с клиентского приложения на сервер (протокол UDP). В программе использовать меню.

Вариант 21

Написать комплекс программ для реализации простого калькулятора. В программе использовать меню. Результат действий

занести в файл и отправить с клиентского приложения на сервер (протокол UDP).

Вариант 22

Написать комплекс программ для перевода чисел из одной системы счисления в другую. Для выбора системы счисления использовать меню. Результат перевода заносится в файл и отправляется с клиентского приложения на сервер (протокол TCP).

Вариант 23

Написать комплекс программ для разложения числа на простые множители. Число и результат операции сохранить в файл и и отправить с клиентского приложения на сервер (протокол TCP). Число пользователь вводит либо с клавиатуры, либо считывает из файла.

Вариант 24

Написать комплекс программ для работы с массивами. Предусмотреть сортировку, поиск и создание массива. Каждое действие над массивом сохранять в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP). В программе использовать меню.

Вариант 25

Написать комплекс программ для преобразования десятичных чисел в римскую систему счисления. Результат выдать на экран и сохранить в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 26

Написать комплекс программ для подсчета числа дней между двумя датами. Диапазон дат задается пользователем либо с клавиатуры, либо находится в файле. Реализовать меню для программы и сохранить

входные и выходные данные в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 27

Написать комплекс программ, реализующий калькулятор с основными арифметическими функциями и функциями сравнения. Для реализации использовать меню. Результаты вычислений занести в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 28

Написать комплекс программ, реализующий конкатенацию строк. Обе строки пользователь поочередно вводит с клавиатуры. Результат операции занести в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

Вариант 29

Написать комплекс программ, реализующий частотный словарь. Текст для анализа загружается из файла или вводится с клавиатуры. Режим ввода анализируемого текста выбирается в меню. Результат частотного анализа занести в файл. Файл отправляется от клиентского приложения на сервер (протокол TCP).

Вариант 30

Написать комплекс программ, реализующий работу с множествами. Дан текст из строчных латинских букв, за которыми следует точка. Напечатать все буквы, входящие в текст не менее двух раз. Результат работы программы сохраняется в файл. Файл отправляется от клиентского приложения на сервер (протокол UDP).

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Назовите особенности ОС FreeBSD.
2. Перечислите этапы разработки ПО.
3. Опишите этапы разработки ПО предшествующие написанию кода.
4. Опишите понятие интерпретатор.
5. Назовите интерпретаторы, встроенные в ОС FreeBSD.
6. Опишите понятие компилятор.
7. Назовите компиляторы, встроенные в ОС FreeBSD.
8. Перечислите и опишите назначение ключей команды gcc.
9. Опишите принцип работы с отладчиком.
10. Опишите действия, необходимые для запуска программы.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение домашней работы отводится 10 академических часов: 9 часов на выполнение и сдачу домашней работы и 1 час на подготовку отчета.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)):

- титульный лист;
- цель, задачи, формулировка задания;
- блок-схема работы программы;
- листинг разработанной программы;
- результаты работы программы;
- выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Вирт, Н. Разработка операционной системы и компилятора. Проект Оберон [Электронный ресурс] / Н. Вирт, Ю. Гуткнехт. — Москва: ДМК Пресс, 2012. 560 с. Режим доступа: <https://e.lanbook.com/book/39992>.
2. Войтов, Н.М. Основы работы с Linux. Учебный курс [Электронный ресурс]: учебное пособие / Н.М. Войтов. — Москва : ДМК Пресс, 2010. — 216 с. — Режим доступа: URL: <https://e.lanbook.com/book/1198>
3. Стащук, П.В. Краткое введение в операционные системы [Электронный ресурс] : учебное пособие / П.В. Стащук. — 3-е изд., стер. — Москва : ФЛИНТА, 2019. — 124 с.— URL: <https://e.lanbook.com/book/125385>

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Войтов, Н.М. Администрирование ОС Red Hat Enterprise Linux. Учебный курс [Электронный ресурс] : учеб. пособие — Москва: ДМК Пресс, 2011. 192 с. Режим доступа: <https://e.lanbook.com/book/1081>.
5. Стащук П.В. Администрирование и безопасность рабочих станций под управлением Mandriva Linux: лабораторный практикум. [Электронный ресурс]: учебно-методическое пособие / П.В. Стащук. — 2-е изд., стер. - М: Флинта, 2015. <https://e.lanbook.com/book/70397>

Электронные ресурсы:

1. Научная электронная библиотека <http://eLIBRARY.RU>.
2. Электронно-библиотечная система <http://e.lanbook.com>.
3. Электронно-библиотечная система «Университетская библиотека онлайн» <http://biblioclub.ru>.
4. Электронно-библиотечная система IPRBook <http://www.iprbookshop.ru/>
5. Losst - Linux Open Source Software Technologies <https://losst.ru>