МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ»

О.Ю. ШАМАЕВА, А.М. ЧЕРНЕЦОВ

Сборник лабораторных работ по курсу «Архитектура вычислительных систем. Часть 1»

Методические указания
для студентов, обучающихся по направлению
«Прикладная математика и информатика»

Москва Издательский дом МЭИ 2020

УДК 621.398

ББК 32.972.26-02

Ш 19

Утверждено учебным управлением НИУ «МЭИ»

в качестве учебного издания

Подготовлено на кафедре прикладной математики и искусственного интеллекта МЭИ

Рецензент: Маран М.М., канд. техн. наук, доцент кафедры прикладной математики и искусственного интеллекта ИВТИ НИУ «МЭИ»

Шамаева О.Ю.

Ш 19 Сборник лабораторных работ по курсу: «Архитектура вычислительных систем. Часть 1». Методические указания /О.Ю. Шамаева, А.М. Чернецов. – М.: Издательский дом МЭИ, 2020. – 36 с.

Сборник содержит 7 лабораторных работ. Первые две работы связаны с темой «Арифметические основы ЭВМ» и предполагают разработку и реализацию на языке высокого уровня алгоритмов обработки чисел в позиционных системах счисления. Третья и четвертая работы посвящены теме «Логические основы ЭВМ» и включают разработку и реализацию алгоритмов вычисления сложной логической функции до 4-х аргументов, а также минимизацию логических функций с последующим построением соответствующих логических схем в разных базисах. Работы №5 и №6 посвящена основам операционной среды UNIX. Лабораторная работа №7 рассматривает вопросы представления текстовой информации в компьютере.

Все работы содержат задания по вариантам, методические указания для их выполнения, а также контрольные вопросы для самопроверки.

Методические указания предназначены для студентов института Информатики и вычислительной техники НИУ «МЭИ», обучающихся по направлению 01.03.02 «Прикладная математика и информатика», а также могут быть полезны студентам, обучающимся по направлению «Информатика и вычислительная техника».

УДК 621.398

ББК 32.972.26-02

Введение

В пособии приводятся задания для 7 лабораторных работ по курсу «Архитектура вычислительных систем. Часть 1», а также методические указания для их выполнения. Теоретический материал, необходимый для выполнения работ излагается в лекционном курсе, дополнительно можно рекомендовать источники [1-6].

Первые две лабораторные работы посвящены изучению арифметических основ компьютеров [2]. В работах рассматриваются базовые системы счисления (с.с.), а именно двоичная, четверичная, восьмеричная шестнадцатеричная, а также смешанные системы, такие как двоично-десятичная и двоично-восьмеричная системы.

Также в работах моделируются алгоритмы обработки чисел, представленных в разных форматах, различных с.с. и использующих различные машинные коды. Алгоритмы обработки подробно рассматриваются на лекциях.

Лабораторные работы 3 и 4 в данном сборнике посвящены логическим основам компьютера [2], включают разработку и реализацию алгоритмов вычисления сложной логической функции, минимизацию логических функций разными методами с последующим построением логических схем, а также проверку их работоспособности. Материал подробно рассматривается на лекциях и практических занятиях.

Выполнение заданий лабораторных работ 1-3 требует от студентов знаний языков программирования высокого уровня. Студентам предлагается использовать языки программирования Pascal, C/C++, C#, однако можно применять и другие, например, Python. Кроме того, для выполнения лабораторной работы 2 необходимы базовые знания языка Ассемблер, которые студенты приобретают ранее при изучении дисциплины «Языки и методы программирования». Подробное описание языка Ассемблер можно также найти, например, в книге [5].

Лабораторные работы 5 и 6 содержат задания по основам работы в операционной системе UNIX, которые можно выполнять на любых версиях семейства UNIX-систем, от Solaris до многочисленных вариантов Linux. Студентам предлагается самостоятельно проработать материал по основам UNIX, используя электронный источник [7]. Для более глубокого знакомства с материалом можно рекомендовать источники [4,6].

Лабораторная работа 7 посвящена стандартам представления текстовой информации в компьютере [8]. Задания для студентов предполагают разработку программы перевода строк символов из одной кодировки в другую.

Лабораторная работа №1
Арифметические основы компьютеров. Часть 1
Перевод чисел из одной системы счисления в другую

Цель работы

Освоение методов и алгоритмов перевода чисел из одной позиционной системы счисления в другую позиционную систему и разработка соответствующего программного продукта.

Подготовка к работе

Изучить краткое теоретическое введение к работе и соответствующий лекционный материал.

Теоретическое введение

Для перевода чисел из одной системы счисления (***q***-ичной) в другую (***p***-ичную) предлагается использовать два базовых алгоритма перевода.

Первый алгоритм перевода переводимое число представляет в виде полинома в старой ***q***-ичной системе; далее в полученном полиноме заменяется основание ***q*** и все коэффициенты числами в новой ***p***-ой с.с.; затем выполняются арифметические операции в новой ***p***-ой с.с.

При необходимости перевода чисел из десятичной системы счисления в любую другую (**10→*q***) вычисления удобнее выполнять в 10-ной с.с., поэтому первый способ перевода становится неудобным.

Второй алгоритм перевода предполагает выполнение вычислений в старой (в ***q***-ичной) с.с. и при этом перевод осуществляется отдельно для целой и дробной части числа.

Целая часть переводимого числа делится на основание ***p*** новой с.с., само деление выполняется в ***q***-ой системе счисления, в качестве результата берутся остатки от деления, записанные с конца. Процесс получения цифр продолжать до тех пор, пока частное не станет равным нулю.

Перевод дробной части из ***q***-ичной системы счисления в ***p***-ичную сводится к умножению исходного число на ***p*** – основание новой с.с.; цифра, получившаяся в целой части произведения, является первой цифрой после запятой в искомом числе. Далее процесс умножения повторяется, но уже применительно к дробной части произведения и цифра, получившаяся в целой части произведения, становится следующей цифрой в искомом числе. Если дробная часть произведения равна нулю или достигнута заданная точность представления числа в новой с.с., то результат получен, иначе процесс умножения повторяется.

В смешанной системе счисления (<***p***-***q***>-ичной) каждая цифра числа, заданного в ***q***-ичной с.с., заменяется соответствующим ее представлением в ***p***-ичной с.с.

Например, в смешанной <2-10> с.с. каждая цифра числа в 10-ой с.с., заменяется тетрадой в 2-ой с.с. Такую запись числа используют или непосредственно, или как промежуточную форму записи между обычной десятичной его записью и машинной двоичной.

Для смешанных с.с. характерно, что количество разрядов, отводимых под каждую цифру, определяется максимальной цифрой базиса *q*-ичной с.с. В 10-ой системе это цифра 9, для представления ее в 2-ой системе требуется четыре разряда. Особый интерес представляет случай, когда $q=p^m$, где m-целое число. Тогда представление числа в p-ичной с.с. совпадает с его представлением в смешанной $<p\text{-}q>$-ичной с.с.

Например, восьмеричное число 741 в 2-ой и <2-8>-ой с.с. имеет одинаковое представление: $741_8 = 111\ 100\ 001_2. = 111\ 100\ 001_{2\text{-}8}$.

Приведенные способы перевода чисел из одной с.с. в другую представляют трудоемкий процесс, который осуществляется автоматически по стандартным подпрограммам. К ручному переводу обращаются в исключительных случаях.

Лабораторное задание

Разработать алгоритм и создать программу решения задачи перевода чисел, заданных в некоторой системе счисления в другую систему, сохранив заданную точность. Оформить полную спецификацию задачи.

Задания выполняются по вариантам.

Варианты заданий:

1. из двоичной в шестнадцатеричную
2. из двоичной в восьмеричную
3. из двоичной в четверичную
4. из десятичной в двоичную
5. из десятичной в восьмеричную
6. из десятичной в шестнадцатеричную
7. из десятичной в четверичную
8. из десятичной в двоично-десятичную
9. из десятичной в двоично-восьмеричную
10. из десятичной в троичную
11. из десятичной в семеричную
12. из десятичной в пятеричную
13. из шестнадцатеричной в пятеричную
14. из шестнадцатеричной в четверичную
15. из шестнадцатеричной в двоичную
16. из шестнадцатеричной в двоично-восьмеричную
17. из шестнадцатеричной в двоично-десятичную
18. из шестнадцатеричной в десятичную
19. из шестнадцатеричной в восьмеричную

20. из двоично-восьмеричной в десятичную
21. из двоично-восьмеричной в четверичную
22. из двоично-восьмеричной в восьмеричную
23. из двоично-восьмеричной в шестнадцатеричную
24. из двоично-восьмеричной в двоично-десятичную
25. из двоично-десятичной в шестнадцатеричную
26. из двоично-десятичной в четверичную
27. из двоично-десятичной в восьмеричную
28. из семеричной в пятеричную
29. из семеричной в двоичную
30. из семеричной в двоично-десятичную
31. из четверичной в троичную
32. из четверичной в восьмеричную
33. из четверичной в шестнадцатеричную
34. из четверичной в двоично-восьмеричную
35. из восьмеричной в двоично-десятичную

Требования

В качестве языка программирования требуется использовать языки C/C++.

Программа должна обрабатывать некоторую совокупность чисел, представленных корректно в текстовом файле. Необходимо отработать возможные аномалии при представлении исходных чисел.

Программа **не должна** использовать стандартные встроенные функции преобразования чисел.

Для контроля правильности программы полезно дополнить программу сравнением двух чисел в разных системах счисления.

Результаты также следует оформить в виде файла, в котором отразить исходные данные и полученные программой результаты.

Методические указания

Создать программный продукт в среде программирования, а также соответствующий выполняемый файл.

Оформить полную спецификацию задачи.

Для защиты выполненной работы преподавателю представляется **отчет** (описание входных/выходных данных, формы ввода/вывода, алгоритм, тесты) и **электронная версия** работающей **программы.** Структура спецификации приведена в Приложении 1.

Контрольные вопросы

1. Переведите числа из одной системы счисления в другую, сохранив заданную точность:

1.1. из десятичной в двоичную: $253,2_{10}$; $79,15_{10}$; $-6,4_{10}$;

1.2. из восьмеричной в пятеричную: $67,5_8$; $34,24_8$; $71,6_8$;
1.3. из семеричной в десятичную: $-63,2_7$; $562,32_7$; $42,5_7$;
1.4. из четверичной в восьмеричную: $0,13(21)_4$

2. Определите, что больше:
2.1. 63_8 или $49_{10;}$
2.2. 1110 1111 0011,1001 1_2 или 51_6;
2.3. $FD5_{16}$ или 8419_{10}.

3. Записать в виде условных целых чисел для размещения в 16-разрядной ячейке следующие десятичные числа: 1, -15, 38, -54, 200.

4. Следующие числа, записанные в десятичной системе счисления, представить в нормальной форме в двоичной системе и указать вид записи этих чисел в 32-разрядной ячейке памяти компьютера:

0.007, –2, 14, –34.85.

5. Укажите диапазон представления чисел с плавающей точкой в 64-разрядной сетке.

Лабораторная работа №2
Арифметические основы компьютеров. Часть 2

Реализация арифметических операций

Цель работы

Освоение методов и алгоритмов выполнения арифметических операций над числами в заданном формате представления в различных системах счисления и разработка соответствующего программного продукта.

Подготовка к работе

Изучить краткое теоретическое введение к работе и соответствующий лекционный материал.

Теоретическое введение

Рассмотрим два формата представления чисел в компьютерах:

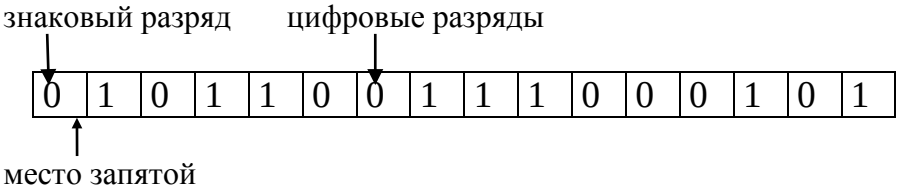
- с фиксированной запятой (или точкой);
- с плавающей запятой (или точкой).

В зависимости от этих двух способов представления чисел, различают два режима функционирования компьютеров: с фиксированной запятой и с плавающей точкой.

В режиме с фиксированной запятой место запятой, отделяющей

целую часть числа от дробной, остается постоянным для всех чисел, т.е. заранее устанавливают, какое количество разрядов отводится для целой части числа, а какое – дробной части. Диапазон представляемых чисел сравнительно невелик, поэтому этот режим неудобен для вычислительных задач с числами из достаточно широкого диапазона. Такой формат обычно используется только в малых и специализированных компьютерах.

Ячейка памяти машины, содержащая число с фиксированной запятой, имеет знаковый разряд и цифровые разряды. Например, двоичное число +0.101100111000101 в шестнадцатиразрядной ячейке запишется так:



Фиксация запятой перед старшим цифровым разрядом требует, чтобы все величины, участвующие в решении задачи, были меньше единицы. В этом случае несколько облегчается подбор масштабных коэффициентов, поскольку при выполнении операции умножения переполнение разрядной сетки исключается, так как в этом случае произведение не превосходит по абсолютной величине ни один из сомножителей. Если требуется хранить точные значения, например, значения индексов переменной, номеров ячеек и т.п., то такие числа обычно записывают в ячейки памяти как числа с фиксированной запятой, предполагая, что запятая стоит после последнего (n–1)-го разряда. Такую форму записи называют записью в виде условного целого числа. Например, число $100_{10}=1100100_2$ в форме условного целого числа запишется в 16-разрядной ячейке как: 0 000 000 001 100 100.

Запись чисел с плавающей запятой в ячейки памяти компьютера основывается на нормальной форме представления чисел – в виде произведения: $\boldsymbol{x==+/-q^p\cdot M}$, где $\boldsymbol{q}$ – основание системы счисления; $\boldsymbol{p}$ –целое число (порядок); $\boldsymbol{M}$ – правильная положительная дробь (мантисса).

Если в первом разряде мантиссы стоит цифра, отличная от нуля, то есть $\mathbf{1/}\boldsymbol{q}\mathbf{\le}\boldsymbol{M}\mathbf{<1}$, то нормальное число называют нормализованным, иначе нормальное число называют ненормализованным.

Например, десятичные числа в нормальной форме имеют вид:

$37.2=10^2\cdot 0.372$ (p=2, M=0.372);

$0.00035=10^{-3}\cdot 0.35$ (p=–3, M=0.35);

Примеры чисел в нормальной форме в двоичной системе:

$$50_{10}=\left(2^6\cdot\frac{25}{32}\right)_{10}=10^{110}\cdot 0.11001_2 \quad (p=110_2,\ M=0.11001_2);$$

$$\frac{3}{32_{10}}=\left(2^{-3}\cdot\frac{3}{4}\right)_{10}=10^{-11}\cdot 0.11_2 \quad (p=-11_2,\ M=0.11_2).$$

Приведенное определение относится ко всем числам, кроме нуля, которое в нормальной форме записывается как: $\mathbf{0} = \boldsymbol{q}^{\boldsymbol{p}} \cdot \mathbf{0}$. Это равенство выполняется при любом $\boldsymbol{p}$. Условимся считать $\boldsymbol{p} = -1\ 000\ 000_2$.

Один из способов записи числа с плавающей запятой в 32-разрядной ячейке: знак числа записывается в нулевом разряде ячейки («+» – 0, «-» – 1), в следующие семь разрядов записывается: $1\ 000\ 000_2 + p_2$, в остальные - записываются цифры мантиссы.

Например, число 5 в нормальной форме в «2» с.с.: $5_{10} = 101_2 = (10^{11} \cdot 0.101)_2$ ($p = 11_2$, $M = 0.101_2$) и в форме с плавающей запятой в 32-разрядной ячейке имеет вид:

0	1	2	3	4	5	6	7	8	9									31
0		1	0	0	0	0	1	1	1	0	1	0	0	0	. . .	0	0	0

Рассмотренная форма записи чисел с плавающей запятой обеспечивает единственность представления каждого числа в машине и обеспечивает достаточно широкий диапазон представления чисел в машине без применения масштабных коэффициентов.

В компьютерах, предназначающихся для решения широкого класса задач, обычно используются оба способа представления чисел, а при решении конкретной задачи выбирается способ, более удобный для данного вида задачи.

Лабораторное задание

Разработать алгоритм и создать программу, реализующую выполнение ***заданной*** арифметической операции над числами, заданными в определенной системе счисления (с.с.) и в определенном ***формате.*** Оформить полную спецификацию задачи.

Задания выполняются по вариантам.

Варианты заданий:

1. Сложение произвольных двоичных чисел с фиксированной точкой с использованием обратных кодов.
2. Сложение произвольных двоичных чисел с фиксированной точкой с использованием модифицированных обратных кодов.
3. Сложение произвольных двоичных чисел с фиксированной точкой с использованием дополнительных кодов.
4. Сложение произвольных двоичных чисел с фиксированной точкой с использованием модифицированных дополнительных кодов.
5. Сложение произвольных двоичных чисел с плавающей точкой с использованием дополнительных кодов.
6. Сложение произвольных двоичных чисел с плавающей точкой с использованием модифицированных дополнительных кодов.

7. Сложение произвольных двоичных чисел с плавающей точкой с использованием модифицированных обратных кодов.
8. Сложение произвольных двоичных чисел с плавающей точкой с использованием обратных кодов.
9. Алгебраическое сложение чисел в двоичной с.с.
10. Алгебраическое сложение чисел в четверичной с.с.
11. Алгебраическое сложение чисел в восьмеричной с.с.
12. Алгебраическое сложение чисел в пятеричной с.с.
13. Алгебраическое сложение в шестнадцатеричной с.с.
14. Алгебраическое сложение чисел в семеричной с.с.
15. Алгебраическое сложение чисел в троичной с.с.
16. Алгебраическое сложение в 12-ричной с.с.
17. Сложение положительных чисел в восьмеричной с.с. и дополнительным контролем сложения в двоичной с.с.
18. Деление произвольных двоичных чисел с фиксированной точкой с использованием прямых кодов.
19. Деление произвольных двоичных чисел с плавающей точкой.
20. Умножение произвольных двоичных чисел с фиксированной точкой.
21. Умножение произвольных двоичных чисел с плавающей точкой.
22. Сложение «2-10» кодированных произвольных чисел.
23. Вычитание «2-10» кодированных произвольных чисел.
24. Умножение произвольных чисел в семеричной с.с.
25. Умножение произвольных чисел в четверичной с.с.
26. Умножение произвольных чисел в восьмеричной с.с.
27. Умножение произвольных чисел в пятеричной с.с.
28. Умножение произвольных чисел в шестнадцатеричной с.с.
29. Умножение «2-10» кодированных произвольных чисел.
30. Деление «2-10» кодированных произвольных чисел.
31. Деление произвольных чисел в семеричной с.с.
32. Деление произвольных чисел в четверичной с.с.
33. Деление произвольных чисел в восьмеричной с.с.
34. Деление произвольных чисел в пятеричной с.с.
35. Деление произвольных чисел в шестнадцатеричной с.с.

Требования

В качестве языка программирования использовать языки Си/ C++, C# и язык Ассемблера.

Для основного блока программы необходимо использовать язык Ассемблера.

При выполнении арифметических действий сопроцессор эмулируется, т.е. при написании кода на ассемблере **нельзя** использовать функции сопроцессора.

Программа **не должна** использовать стандартные встроенные функции преобразования чисел.

Программа должна обрабатывать некоторую совокупность чисел, представленных корректно в текстовом файле. Необходимо отработать возможные аномалии при представлении исходных чисел.

Результаты также следует оформить в виде файла, в котором отразить исходные данные и полученные программой результаты.

Оформить полную спецификацию задачи. Структура спецификации приведена в Приложении 1.

Для защиты выполненной работы преподавателю представляется отчет и электронная версия работающей программы.

Методические указания

Исходные числа для обработки представляются в десятичной системе счисления, далее преобразуются в числа в системе счисления, заданной в варианте, над которыми и выполняется требуемая операция.

Результат представляется в той же системе счисления, в которой выполнялась операция, а также переводится и в **десятичную** систему счисления. Для контроля правильности работы программы целесообразно выполнить указанную в задании операцию непосредственно в десятичной системе.

Создать программный продукт в среде программирования, а также соответствующий выполняемый файл.

Контрольные вопросы

1. Чем определяется необходимость введения машинных кодов?
2. Как формируются прямой, обратный, дополнительный коды для положительных и отрицательных чисел?
3. Приведите алгоритм вычитания 2-х произвольных «2-10» чисел. Почему в данном алгоритме не требуется проведение 1-ой коррекции чисел?
4. Перечислите свойства обратных и дополнительных кодов.
5. Каковы преимущества при использовании модифицированных дополнительных кодов?
6. Что означает комбинация цифр «01» и «10» в знаковых разрядах результата при выполнении операции сложения в модифицированных кодах?
7. Что такое «машинная единица» и «машинный ноль»?
8. Сформулируйте алгоритм умножения двоичных чисел, представленных в формате с плавающей запятой
9. Как определяется число знаков после запятой при выполнении операции деления над числами с плавающей точкой?

10. Как определяется цифра частного (0 или 1) при делении чисел в формате с фиксированной точкой?

Лабораторная работа №3
Логические основы компьютеров. Часть 1
Вычисление значений логических функций

Цель работы

Автоматизация процессов вычисления сложных логических функций на основе представления их таблицами истинности и разработка соответствующего программного продукта.

Подготовка к работе

Изучить соответствующий лекционный материал и теоретическое введение к практическому занятию по логическим основам ЭВМ.

Лабораторное задание

Разработать алгоритм и создать программу, реализующую построения таблицы истинности для логической функции до четырех аргументов. Оформить полную спецификацию задачи.

Требования

В качестве языка программирования можно использовать языки Паскаль или Си/Си++, C#.

Программа должна вычислять значение логических функций с использованием аппарата процедур и функций.

Для каждой **базовой** логической функции необходимо создать процедуру или функцию, которые будут вызываться в процессе вычисления значений заданной сложной функции.

На входе программы – логическая функция от четырех аргументов, заданная аналитически. На выходе формируется полная таблица истинности для исходной функции, заданной аналитически.

Программу следует оттестировать на простейших примерах, а также на логических функциях, приведенных в Приложении 2.

Оформить полную спецификацию задачи.

Методические указания

Создать программный продукт в среде программирования, а также соответствующий выполняемый файл.

Для защиты выполненной работы преподавателю представляется отчет с результатами вычислений (полная таблица истинности) и электронная версия работающей программы.

Контрольные вопросы

1. Приведите определение понятия логической функции.
2. Какие существуют формы представления логических функций?
3. Сколько различных логических функций можно сгенерировать для пяти аргументов?
4. Перечислите основные свойства базовых логических функций.
5. Каким свойствам не удовлетворяют функции импликации и эквивалентности?
6. Каким свойствам не удовлетворяют функции штрих Шеффера и сложения по модулю 2?
7. Каким свойствам не удовлетворяют функции стрелка Пирса и сложения по модулю 2?
8. Проверьте выполнимость закона дистрибутивности для функций «И-НЕ» и «ИЛИ-НЕ»
9. Укажите основные законы преобразования логических функций.
10. Приведите выражение для правила де Моргана.

Лабораторная работа №4
Логические основы компьютеров. Часть 2

Минимизация логической функции и проектирование логических схем

Цель работы

Освоение алгоритмов минимизации логических функций и как результат проектирование логических схем и проверка их работоспособности.

Подготовка к работе

Изучить соответствующий лекционный материал и теоретическое введение в практическое занятие по логическим основам ЭВМ.

Лабораторное задание

По заданной таблице истинности (варианты задания соответствуют номеру заданной функции) составить описание логической функции в ДСНФ и КСНФ.

Провести минимизацию и эквивалентные преобразования полученного выражения, используя два различных метода, один из которых табличный метод с применением диаграмм Вейча (сравнить результаты) с целью приведения к двум базисам (**И-ИЛИ-НЕ**; **И-НЕ**).

Построить две логические схемы, соответствующие полученным выражениям в двух базисах. Проверить их работоспособность.

Таблица

Таблица истинности логической функции

Значения переменных X_1 X_2 X_3 X_4		Значения функции для вариантов с 1-го по 10-ый									
		Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	Y_{10}
1	0 0 0 0	1	1	1	1	1	0	0	0	0	0
2	0 0 0 1	0	0	0	1	0	1	1	1	1	1
3	0 0 1 0	1	1	1	1	1	0	0	0	0	1
4	0 0 1 1	0	0	0	0	1	0	1	1	1	1
5	0 1 0 0	0	0	1	0	0	1	1	0	1	1
6	0 1 0 1	1	1	1	1	1	0	0	0	0	0
7	0 1 1 0	0	0	0	0	0	1	1	1	1	1
8	0 1 1 1	1	1	1	1	1	0	0	1	0	0
9	1 0 0 0	1	1	1	1	1	0	0	0	0	0
10	1 0 0 1	0	0	0	0	0	1	1	1	1	1
11	1 0 1 0	1	1	1	1	1	0	0	0	0	0
12	1 0 1 1	1	0	0	0	0	1	1	1	1	0
13	1 1 0 0	0	1	0	0	0	1	1	1	0	1
14	1 1 0 1	1	1	1	1	1	1	0	0	0	0
15	1 1 1 0	0	0	0	0	0	1	1	1	1	1
16	1 1 1 1	1	1	0	1	0	0	0	0	1	0

Значения переменных X_1 X_2 X_3 X_4		Значения функции для вариантов с 11-го по 20-ый									
		Y_{11}	Y_{12}	Y_{13}	Y_{14}	Y_{15}	Y_{16}	Y_{17}	Y_{18}	Y_{19}	Y_{20}
1	0 0 0 0	1	1	1	1	1	1	0	0	0	1
2	0 0 0 1	0	0	0	1	0	1	1	1	0	1
3	0 0 1 0	1	0	0	0	0	1	1	1	1	0
4	0 0 1 1	1	1	1	1	1	0	0	0	0	0
5	0 1 0 0	0	0	0	1	0	1	0	1	1	1
6	0 1 0 1	1	1	1	1	1	0	0	0	0	0
7	0 1 1 0	0	0	0	0	1	0	1	1	1	1
8	0 1 1 1	1	1	1	1	1	0	0	0	0	0
9	1 0 0 0	0	0	0	0	0	1	1	1	1	1
10	1 0 0 1	1	1	1	1	1	0	0	1	1	1
11	1 0 1 0	0	1	1	0	0	1	0	0	1	0
12	1 0 1 1	1	1	0	0	1	1	1	1	0	0
13	1 1 0 0	0	1	1	1	0	1	1	1	1	1
14	1 1 0 1	0	0	0	1	1	1	1	0	1	0
15	1 1 1 0	1	0	1	0	0	0	1	0	1	1
16	1 1 1 1	1	1	1	0	1	0	0	1	0	1

	Значения функции для вариантов с 21-го по 30-ый

Значения переменных X_1 X_2 X_3 X_4		Y_{21}	Y_{22}	Y_{23}	Y_{24}	Y_{25}	Y_{26}	Y_{27}	Y_{28}	Y_{29}	Y_{30}
1	0 0 0 0	0	1	1	0	1	0	0	0	0	1
2	0 0 0 1	1	1	0	1	0	0	1	1	0	1
3	0 0 1 0	0	0	1	0	0	1	1	1	1	0
4	0 0 1 1	1	0	1	1	1	0	0	0	0	0
5	0 1 0 0	0	0	1	1	1	1	0	1	1	0
6	0 1 0 1	1	0	1	1	0	1	0	0	0	0
7	0 1 1 0	0	1	1	0	0	1	1	1	1	1
8	0 1 1 1	1	0	1	1	1	0	1	1	0	0
9	1 0 0 0	0	0	0	1	0	0	1	0	1	1
10	1 0 0 1	1	1	0	0	1	0	1	1	1	1
11	1 0 1 0	0	1	0	0	0	1	0	0	1	1
12	1 0 1 1	1	1	0	0	1	1	0	1	1	0
13	1 1 0 0	0	1	1	1	0	1	1	1	0	1
14	1 1 0 1	0	0	0	1	1	1	0	0	0	0
15	1 1 1 0	1	0	0	0	0	0	1	0	1	0
16	1 1 1 1	1	1	1	0	1	0	0	1	0	1

Требования

Оформить отчет по результатам минимизации исходной функции, указав все основные этапы минимизации с использованием аналитического и табличного методов. Построить две логические схемы на основе двух базисов (**И-ИЛИ-НЕ**; **И-НЕ**). Проверить правильность минимизации с помощью программы, созданной в ходе выполнения лабораторной работы 3.

Выполнить проверку работоспособности построенных логических схем.

Методические указания

Для минимизации логической функции можно разработать и реализовать программный продукт в среде программирования.

Для защиты выполненной работы преподавателю представляется отчет с результатами поэтапной минимизации, логическими схемами и таблицами по проверке работоспособности схем.

Контрольные вопросы

1. Какие законы алгебры логики используются при минимизации логических функций?
2. В чем заключается необходимость в построении таблицы покрытия при минимизации логической функции?
3. Определите понятие ДСНФ и КСНФ.
4. Что такое логически полный базис?

5. В чем преимущество табличных методов минимизации по сравнению с аналитическими и когда целесообразно их использовать?
6. Как определяется работоспособность логической схемы?
7. К какому классу схем можно отнести построенные в работе схемы?
8. Приведите пример регулярной и нерегулярной комбинационной схемы.
9. Описать таблицей истинности и построить логическую схему сумматора на основе компаратора.
10. Что надо добавить в многоразрядный сумматор последовательного действия, чтобы обеспечить его параллельное функционирование?

Лабораторная работа №5
Операционная система UNIX. Часть 1

Файловая система и работа в сети

Цель работы

Освоение работы с консолью, системы безопасности и прав доступа к файлам, системы команд. Работа в сети. Взаимодействие с Unix-сервером.

Подготовка к работе

Изучить необходимый материал по интернет-курсу «Основы операционной системы Unix» [7].

Лабораторное задание

1. Войти в систему (логин и пароль, а также IP-адрес сервера следует узнать у преподавателя).
2. Выполнить регистрацию в системе. Изменить пароль (***passwd***). Выполнить просмотр информации о работающих пользователях (***who***).
3. Познакомиться со справочным руководством (***man***).
4. В своем домашнем каталоге создать иерархическую 3-х уровневую структуру из 6-ти каталогов (например, каталог А с подкаталогами А1 и А2; в подкаталоге А1 создать подкаталоги А11 и А21; в подкаталоге А11 создать текстовые файлы file1.txt и file2.txt и т.п.).
5. Осуществить просмотр содержимого созданных файлов (***cat***, ***more***).
6. Создать файлы различных типов (***cat***, ***mkdir***, ***mknod***, ***ln***).
7. Выполнить копирование, перемещение и удаление файлов и каталогов (***cp***, ***mv***, ***rm***, ***rmdir***).

8. Освоить команды: ***mkdir, chdir, ls, rm, pwd, passwd*** в различных режимах (в частности, ***ls*** в «коротком» (“***ls***”) и «длинном» (“***ls -al***”) вариантах).
9. Изучить способ определения типов файлов в UNIX и ознакомиться с командой ***file***. Изучить специальные типы файлов (файлы устройств).
10. Изучить средства создания ссылок в файловой системе (команда ***ln***).
11. Выполнить просмотр информации о файлах, изменение прав доступа и владельца файлов (***ls***, ***chmod***, ***chown***, ***chgrp***, ***umask***).
12. Назначить одному из файлов права. Убрать право “**x**”, назначить “**w**”.
13. Выполнить поиск файлов по различным критериям (***find)***.
14. Отработать совместное использование каталогов: одному из пользователей зайти в домашний каталог другого, просмотреть созданную им структуру. Отредактировать файлы.
15. Удалить созданную структуру.
16. Просмотреть список процессов и информацию о процессах, установить поправки приоритета (***ps***, ***nice***).
17. Выполнить посылку сигналов процессам (***kill***).
18. Изучить средства переадресации ввода-вывода и конвейеризации команд интерпретатора. Записать в файл с именем текущей даты информацию о процессах, запущенных от имени пользователя.
19. Получить справку по командам ***ls, chmod***.
20. Изучить основные стандартные интерпретаторы (**bash**, **csh**, **ksh**) и главные различия между ними.
21. С помощью команды Windows ***ipconfig*** (используется из командной строки windows) узнать IP-адреса локальных машин. Проверить, используя команду ***ping***, проходят ли пакеты между ними. С помощью ***traceroute*** (в версии Windows команда называется ***tracert,*** используется таким же образом, как и в UNIX) определить, каким образом – с участием сервера или без него – ходят пакеты между машинами.
22. Обменяться сообщениями электронной почты, используя почтовую программу ***mail.***
23. Завершить сеанс работы.

Требования

Отчет по работе оформляется по каждому из пунктов задания и должен отражать проделанную работу и полученный результат.

Отчет должен включать:

- дерево каталогов,

- описание сеанса по его созданию,
- пример выполнения каждого из пунктов задания – введенную команду и полученный результат.

Для защиты выполненной работы необходимо предоставить преподавателю оформленный отчет и продемонстрировать навыки владения командами в среде Unix.

Методические указания

Первая часть работы в операционной среде Unix в общем случае может быть выполнена как в автономном режиме (когда система Unix установлена на автономном компьютере), так и на клиентской машине в сетевом режиме.

При выполнении работы в компьютерном классе кафедры операционная система Unix установлена на одной – серверной – машине; остальные машины являются клиентскими в рамках локальной сети. Поэтому для выполнения задания в дисплейном классе прежде всего необходимо установление связи с сервером: обеспечение удаленного доступа посредством протоколов **telnet/ssh**.

Заметим, что в операционных системах Windows Vista и более новых версий, по умолчанию отсутствует установка клиента Telnet. Также для всех версий ОС Windows (до Windows 10 10/18) отсутствует встроенный клиент для работы по протоколу **ssh**.

Студентам рекомендуется использовать свободно распространяемую программу **Putty** [10], а для переноса данных между сервером Unix и компьютером в классе – программу **Winscp** [11]. Для выполнения лабораторных работ достаточно использовать версии программы **Putty 0.60** и выше, а также **Winscp** версии 4.0 и выше. Описание работы с данным программным обеспечением приведено в Приложении 3.

В случае использования клиента системы UNIX, все необходимые команды уже имеются в стандартной инсталляции.

Контрольные вопросы

1. Перечислите основные возможности операционной системы Unix.
2. В чем заключаются особенности организации работы с электронной почтой в системе Unix?
3. Какие группы прав и права существуют в Unix?
4. Каковы особенности модели доступа в системе Unix?
5. Что такое символическая ссылка на файл?
6. Чем отличаются жесткие и символические ссылки в файловой системе Unix?

7. Как определить какие права назначены файлу по умолчанию?
8. В чем состоит принципиальное отличие определения типов файлов в ОС Windows и Unix?
9. Как переадресовать стандартный ввод-вывод в текстовой файл?
10. Чем являются и зачем используются специальные файлы **/dev/null, /dev/tty, dev/mouse**?

Лабораторная работа №6
Операционная система UNIX. Часть 2

Программирование на языке командного интерпретатора

Цель работы

Освоение языка командного интерпретатора Unix стандарта **bash** ("**B**ourne **a**nother **sh**ell"). Разработка программ на языке командного интерпретатора.

Подготовка к работе

Изучить необходимый материал по интернет-курсу «Основы операционной системы Unix» [7].

Лабораторное задание

Разработать программу, реализующую алгоритм (разработанный при выполнении лабораторной работы №1) перевода чисел из одной системы счисления в другую на языке командного интерпретатора Unix (стандарт **bash**).

Требования

Реализация должна позволять обработку исходных данных, полученных как в интерактивном режиме, так и при задании в командной строке.

Например, для перевода из 16-й системы счисления в 8-ую систему в командной строке необходимо записать:

$ ./perevod F.8 -6.4

<output>

Отчет по работе должен включать:

- постановку задачи (в соответствии с вариантом лабораторной работы №1);
- описание алгоритма;
- кодирование на языке командного интерпретатора *bash*;
- результаты тестирования (примеры выполнения расчетов).

Для защиты выполненной работы необходимо предоставить преподавателю оформленный отчет и продемонстрировать работоспособность программы.

Методические указания

Общие методические указания совпадают с указаниями для лабораторной работы 5.

Контрольные вопросы

1. Определите понятия: простая команда, конвейер и список.
2. Какие метасимволы используются для генерации имен файлов и экранирования?
3. Как получить в программе доступ к имени файла-аргумента?
4. Какая переменная называется «переменная среды»?
5. Приведите основные команды интерпретатора bash**.**
6. Как разработать функции в командном интерпретаторе bash? Как обеспечить вызов функций?
7. Каковы основные конструкции встроенного языка программирования интерпретатора bash?
8. С помощью каких команд выполняются операции ввода-вывода в интерпретаторе bash?
9. Перечислите команды, поддерживающие выполнение арифметических операций в bash
10. Перечислите команды, поддерживающие выполнение логических операций в bash

Лабораторная работа №7
Представление текстовой информации в компьютере
Коды ANSI и UNICODE

Цель работы

Освоение основных стандартов представления информации в компьютере, разработка алгоритмов перекодирования текста из одной системы кодирования в другую.

Подготовка к работе

Самостоятельно изучить стандарты представления текстовой информации в компьютере по интернет-курсу «Представление текстовой информации в ЭВМ» [8].

Лабораторное задание

1. Запишите в двоичном и шестнадцатеричном коде английскую букву A.
2. Запишите в двоичном и шестнадцатеричном коде в альтернативной кодировке русскую букву а.
3. Запишите в двоичном и шестнадцатеричном коде в кодировке Windows-1251 русскую букву Я.
4. Запишите символ, который в кодировке Windows-1251 имеет двоичный код 110110.
5. В ОС Windows в командном интерпретаторе (***cmd***) изучить команды смены кодовых страниц. Записать текст в кодировке CP866 в отдельный файл. Убедиться, что кодировка текста CP866.
6. В ОС Windows в командном интерпретаторе (***cmd***) изучить команды смены кодовых страниц. Записать текст в кодировке Unicode в отдельный файл. Убедиться, что кодировка текста Unicode.
7. В ОС Unix в командном интерпретаторе (***bash***) изучить команды смены кодовых страниц
8. В ОС UNIX изучить команду ***iconv*** для перекодирования текста. Решить задачу: перекодировать содержимое всех текстовых файлов (*.txt) в заданном каталоге.
9. Следующее задание выполняется по вариантам, приведенным в таблице. Разработайте алгоритм перевода строки символов из «Откуда» в «Куда» с соблюдением указанного «Порядка»:

Таблица

№ Варианта	Откуда	Куда	Порядок
1	ANSI	UTF-8 без BOM	BIG_ENDIAN
2	ANSI	UTF-8 без BOM	LITTLE_ENDIAN
3	ANSI	Win-1251	BIG_ENDIAN
4	ANSI	KOI-8U	BIG_ENDIAN
5	ANSI	KOI-8U	LITTLE_ENDIAN
6	ANSI	UTF-16	LITTLE_ENDIAN
7	ANSI	KOI-8R	LITTLE_ENDIAN
8	ANSI	ISO-8559-5	LITTLE_ENDIAN
9	ANSI	Win-1251	LITTLE_ENDIAN
10	ANSI	KOI-8U	LITTLE_ENDIAN
11	UTF-8	UTF-16	BIG_ENDIAN
12	UTF-8	KOI-8R	BIG_ENDIAN
13	UTF-8	ISO-8559-	BIG_ENDIAN
14	UTF-8	Win-1251	BIG_ENDIAN
15	UTF-8	KOI-8U	BIG_ENDIAN
16	UTF-8	UTF-16	LITTLE_ENDIAN
17	UTF-8	KOI-8R	LITTLE_ENDIAN

18	UTF-8	ISO-8559-	LITTLE_ENDIAN
19	UTF-8	Win-1251	LITTLE_ENDIAN
20	UTF-8	KOI-8U	LITTLE_ENDIAN
21	UTF-16	UTF-8	BIG_ENDIAN
22	UTF-16	KOI-8R	BIG_ENDIAN
23	UTF-16	ISO-8559-	BIG_ENDIAN
24	UTF-16	Win-1251	BIG_ENDIAN
25	UTF-16	KOI-8U	BIG_ENDIAN
26	UTF-16	UTF-8	LITTLE_ENDIAN
27	UTF-16	KOI-8R	LITTLE_ENDIAN
28	UTF-16	ISO-8559-	LITTLE_ENDIAN
29	UTF-16	Win-1251	LITTLE_ENDIAN
30	UTF-16	KOI-8U	LITTLE_ENDIAN
31	Win-1251	UTF-8	BIG_ENDIAN
32	Win-1251	KOI-8R	BIG_ENDIAN
33	Win-1251	ISO-8559-	BIG_ENDIAN
34	Win-1251	UTF-16	BIG_ENDIAN
35	Win-1251	KOI-8U	BIG_ENDIAN
36	Win-1251	UTF-8	LITTLE_ENDIAN
37	Win-1251	KOI-8R	LITTLE_ENDIAN
38	Win-1251	ISO-8559-	LITTLE_ENDIAN
39	Win-1251	UTF-16	LITTLE_ENDIAN
40	Win-1251	KOI-8U	LITTLE_ENDIAN

Требования

В качестве языка программирования можно использовать языки Паскаль или Си/Си++, C#.

Программа должна производить перевод строки символов из одной кодировки в другую.

Программу следует оттестировать на простейших примерах. Оформить полную спецификацию задачи.

Отчет по работе оформляется по каждому из пунктов задания и должен отражать проделанную работу и полученный результат.

Для защиты выполненной работы необходимо предоставить преподавателю оформленный отчет и продемонстрировать знания по различным кодировкам.

Методические указания

Создать программный продукт в среде программирования, а также соответствующий выполняемый файл.

Проверку корректности перекодирования производить в программе - редакторе текста Notepad++ [9].

Контрольные вопросы

1. Что такое кодировка символов?
2. Чем отличаются порядок байтов LITTLE_ENDIAN и BIG_ENDIAN?
3. Поясните, чем отличаются однобайтовые и многобайтовые кодировки?
4. Перечислите особенности стандарта Unicode?
5. Как перекодировать знак из UTF 8 в Win-1251?
6. Сколько байт требуется для хранения 1 символа в кодировке UTF-8?
7. Сколько байт требуется для хранения 1 символа в кодировке UTF-8 без BOM?
8. Сколько байт требуется для хранения 1 символа в UTF-16?
9. Перечислите средства редактора Notepad++ для преобразования символов в другие кодировки
10. Укажите средства редактора Notepad++ для сохранения информации в других кодировках

Приложение 1. Оформление внешних спецификаций

1. **Задача**

<чётко сформулированное условие задачи>

2. **Состав данных**

тип	имя	смысл	структура	диапазон	точность	формат
		Исходные данные Результаты Промежуточные данные				

3. **Форма ввода**

<расположение данных на экране монитора ПЭВМ или в файле>

4. **Выходная форма**

<заголовок программы>
<исходные данные>
[промежуточные данные]
<результаты>
<реакция на недопустимые ситуации>

5. **Метод**

<словесное описание метода или его название, если метод общеизвестен>

6. **Недопустимые ситуации** (аномалии)

<ситуация, при которой решение - <реакция на неё> прерывается>

7. **Тесты**

№	<исходные данные>	<ожидаемые результаты>

8. **Описание алгоритма** (на языке блок-схем, на псевдокоде и др.)

9. **Программа**

Приложение 2. Список логических функций для тестирования результатов лабораторной работы №3

$F_1(x,y,z) = \{[(\bar{x} \equiv z) \vee (x \oplus y)] \,\&(x \oplus \overline{y})\} \to z$

$F_2(x,y,z) = (x \to y) \to \big((x \to z) \to (x \to y \,\&z)\big)$

$F_3(x,y,z) = \overline{\{[(x \oplus \overline{y}) \vee (\overline{x} \to z)] \,\&(y/z)\}}$

$F_4(x,y,z) = \Big\{(x/\overline{y}) \vee \big(\overline{x \oplus y}\big) \to \big(\overline{\overline{z} \,\& x}\big)\Big\} \equiv z$

$F_5(x,y,z) = \{[(\overline{x} \to \overline{z}) \oplus (y \oplus z)]\&(\overline{y} \equiv z)\}/x$

$F_6(x,y,z) = \{[(x \oplus \overline{z}) \,\&(x \to \overline{y})] \vee (\overline{x} \to y)\}/\overline{z}$

$F_7(x,y,z) = [\{(x \to \overline{y}) \,\&(x/z)\} \to \overline{x}] \oplus \big(\overline{x \vee y}\big)$

$F_8(x,y,z) = [(x \vee y \,\&\overline{z}) \oplus (y \to z \to \overline{x})] \to (x/\overline{z})$

	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
$F_1(x,y,z)$	1	1	1	1	1	1	0	1
$F_2(x,y,z)$	1	1	1	1	1	1	1	1
$F_3(x,y,z)$	0	0	1	1	0	0	0	1
$F_4(x,y,z)$	0	1	0	1	0	1	1	1
$F_5(x,y,z)$	1	1	1	1	1	1	1	1
$F_6(x,y,z)$	0	1	0	1	0	1	0	1
$F_7(x,y,z)$	0	0	1	1	0	1	1	1
$F_8(x,y,z)$	1	1	1	1	0	1	1	1

Приложение 3. Работа с клиентами TELNET, SSH

3.1. Встроенный клиент **TELNET**

Протокол прикладного уровня **TELNET** (от англ. **TE**rmina**L** **NET**work) — сетевой протокол для реализации текстового интерфейса по сети. Название **telnet** получили также клиентские программы реализации данного протокола, практически для всех существующих операционных систем. Протокол **Telnet** – один из старейших сетевых протоколов, разрабатывавшихся как средство связи между удаленными терминалами в тестовом режиме. Поэтому в нем не предусмотрено шифрование данных и использование современных средств проверки подлинности. Протокол уязвим для множества сетевых атак, и не может использоваться в качестве средства управления сетевыми операционными системами.

В современных ОС семейства Windows, утилита **telnet.exe** по умолчанию, не устанавливается. Для ее установки нужно перейти в **Панель управления - Программы и Компоненты – Включение или отключение компонентов Windows** и установить галочку для **Клиент Telnet**. Для желающих работать в режиме командной строки, можно в командной строке, запущенной от имени администратора, выполнить команду: **pkgmgr /iu:"TelnetClient"**

Формат командной строки:

telnet [-a][-e Символ][-f Файл][-l Имя][-t Тип][Узел [Порт]]

где **Узел** - имя узла или **IP**-адрес удаленного компьютера, к которому выполняется подключение; **Порт** - номер порта или имя службы. Если номер не задан, то по умолчанию используется стандартный порт **Telnet 23/TCP.**

Запустить окно командной строки (cmd) и набрать в командной строке команду:

telnet <ip addr> <port>

Для аварийного завершения работы (в случае, если сеанс завис) служит сочетание клавиш «**Ctrl+]**» - символ переключения режима.

В настоящее время, для удалённого доступа к системе применяется сетевой протокол **SSH** (**S**ecure **SH**ell), при создании которого упор делался именно на вопросы безопасности. Относительная безопасность сессий **Telnet** осуществляется только в полностью контролируемой сетевой среде или с применением защиты на сетевом уровне (различные реализации VPN - виртуальных частных сетей).

Протокол **ssh** по умолчанию использует **tcp** порт **22**.

Формат командной строки (**для UNIX-систем**):

ssh [-l имя_регистрации] [имя_машины | пользователь@имя_машины] [команда]

ssh [-afgknqstvxACNPTX1246] [-c cipher_spec] [-e escape_char] [-I файл_идентификации] [-l имя_регистрации] [-m mac_spec] [-o параметр] [-p порт] [-L порт:машина:порт_машины] [-R порт:машина:порт_машины] [имя_машины | пользователь@имя_машины] [команда]

где **имя_регистрации (пользователь)** - имя учётной записи пользователя на удалённой машине;

имя_машины – имя удалённого компьютера;

команда – команда, которую необходимо выполнить на удалённом компьютере.

Из остальных многочисленных параметров отметим лишь самые важные:

- 1 использовать протокол SSH версии 1

- 2 использовать протокол SSH версии 2

- p порт - номер порта. Если номер не задан, то по умолчанию используется стандартный порт **ssh 22/TCP.**

Подробное описание ключей команды **ssh** имеется в документации.

Заметим, что для всех версий ОС Windows (вплоть до версии Windows 10 10/17) отсутствует встроенный клиент для работы по протоколу **ssh.** В Windows 10 10/17 компания **Microsoft** наконец-то реализовала встроенный клиент **ssh** в рамках операционной системы. Тем не менее, поскольку используется клиент **OpenSSH**, нет графического интерфейса. Поэтому существующая уже много лет программа **Putty** является наиболее удобным и простым приложением, реализующим ssh-клиент в ОС Windows.

3.2. Программа **Putty**

Putty – программа для системных администраторов и пользователей, работающих с локальной сетью, позволяет передавать специальные команды по популярным сетевым протоколам. Программа является одним из популярных инструментов под OS Windows. Основное назначение – **передача команд подключенным устройствам** (хостам) по протоколам **SSH**, **Telnet** и **Rlogin**, а также настройка устройств с помощью COM-портов. Актуальную версию программы PuTTY можно скачать бесплатно из сети интернет.

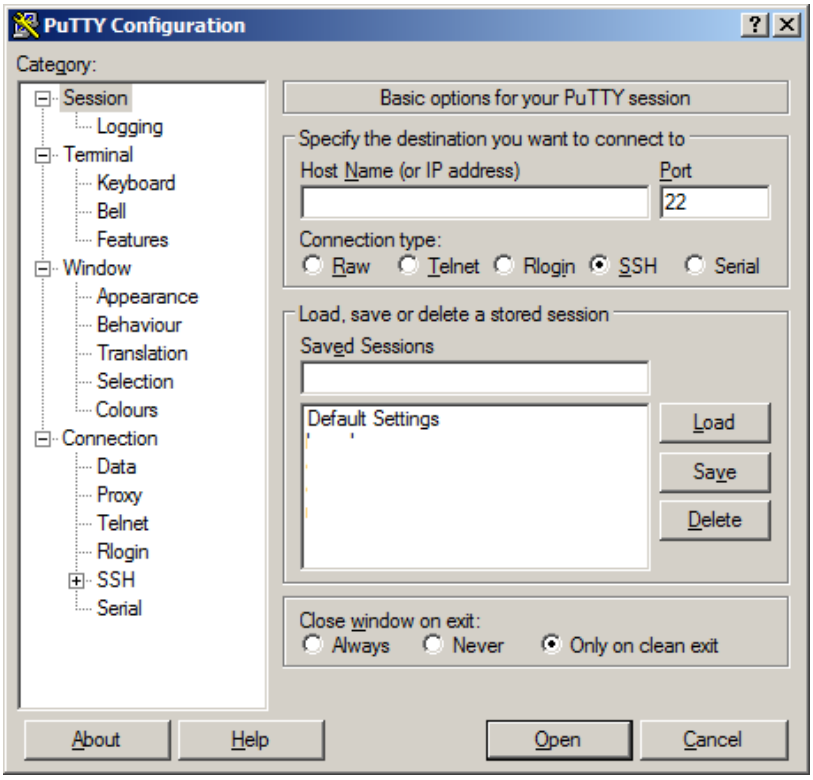


Рис. 1. Окно настройки программы PuTTY

После запуска программы, в окне настройки **PuTTy Configuration**, представленном на рис.1 в поле **Host Name** (or **IP address**) следует вводить **ip** адрес сервера UNIX.

Port – это поле, в котором указывается порт для подключения устройства. После этого для удобства можно сохранить все введенные данные, чтобы не прописывать их каждый раз при открытии программы. Для этого в поле «**Saved sessions**» необходимо указать название выполняемого подключения, например, work 1, work 2 и т.д. и нажать кнопку «**Save»**.

В дальнейшем, активируя кнопку «**Open**», на рабочем столе возможно открыть окно, где пользователь может ввести данные для дальнейшей авторизации. Во время первого открытия программы **PuTTY** и подключения к серверу по протоколу **SSH** пользователь должен согласиться с записью ключа для удаленного сервера.

В меню слева (см. рис. 1) доступны настройки программы **PuTTY**. В частности, настройки взаимодействия по протоколу **SSH** находятся во вкладке **SSH**.

Пример работы приведён на рисунках 2-4. В качестве **IP**-адреса сервера используется адрес – **10.5.2.25.**

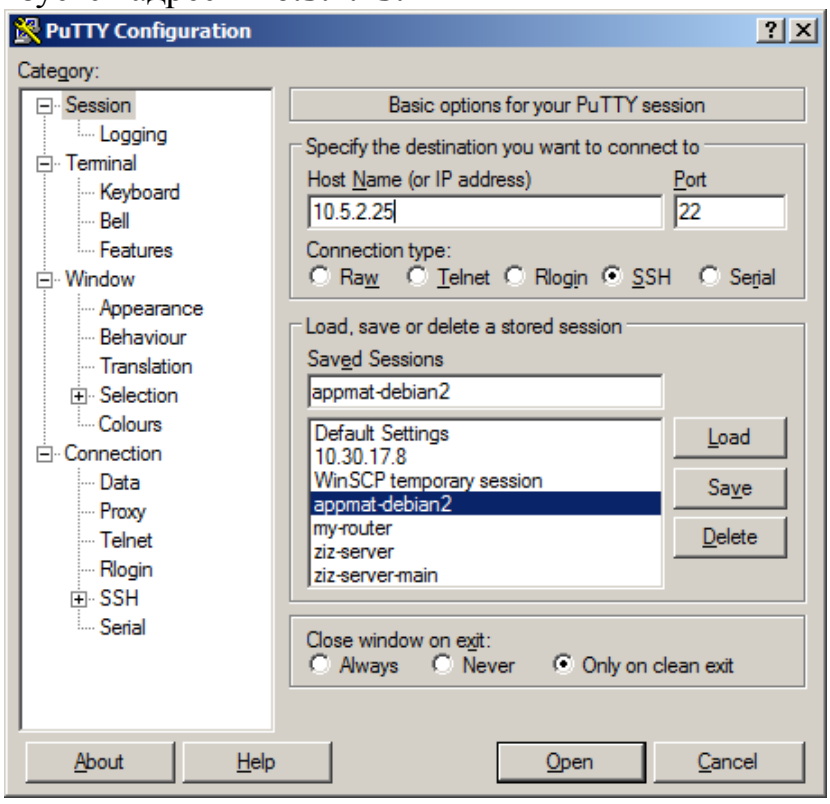


Рис. 2. Выбор сервера для запуска сеанса

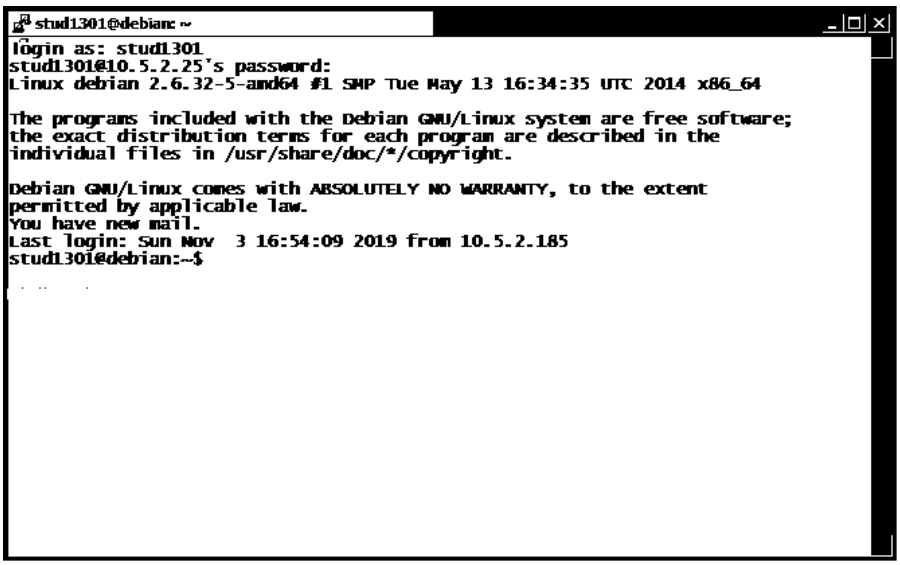


Рис. 3. Сеанс работы пользователя

3.3. Программа **Winscp**

WinSCP — свободный графический клиент с открытым исходным кодом протоколов **SFTP** и **SCP**, предназначенный для ОС Windows. Распространяется по лицензии **GNU GPL**. Обеспечивает защищённое копирование файлов между компьютером и серверами, поддерживающими эти протоколы. Также позволяет работать по протоколу **FTP,** имеет средства интеграции с программой **Putty,** имеет локализованные версии на большом числе языков, в том числе и русском. Основной интерфейс программы приведен на рис. 4.

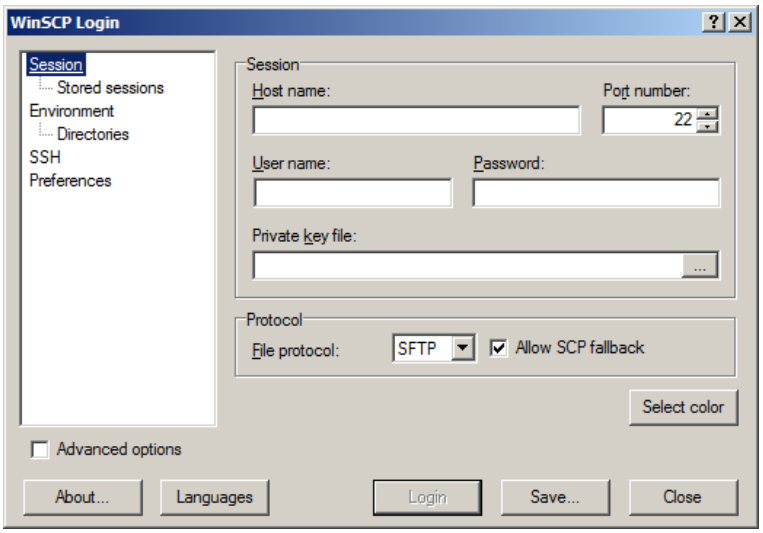


Рис. 4. Интерфейс программы **WinSCP**

В поле **Host Name** (or **IP address**) следует вводить **ip** адрес сервера UNIX. Для удобства можно сохранить сеанс, введя имя пользователя и нажав кнопку «**Save**». ***Замечание:*** *ни в коем случае не следует сохранять пароль пользователя!*

Список рекомендуемой литературы

Основная литература:

1. Орлов С.А., Цилькер Б.Я. Организация ЭВМ и систем: Учебник для вузов. 3-е изд. Стандарт третьего поколения. – СПб.: Питер, 2014. – 688 с.
2. Пятибратов А.П., Гудыно Л.П., Кириченко А.А. Вычислительные системы, сети и телекоммуникации: под ред. А.П. Пятибратова. – 4-е изд., перераб. и доп. – М., «Финансы и статистика», 2014. – 735 с.
3. Э. Таненбаум, Т. Остин. Архитектура компьютера – Санкт-Петербург, Питер, 2017 г., - 816 с.

Дополнительная литература:

4. Э. Таненбаум, Х. Бос. Современные операционные системы -4-е изд. Санкт-Петербург, Питер, 2015 г. - 1120 с.
5. В. Юров. Язык Ассемблер 2-е изд. - Санкт-Петербург, Питер, 2010 г. - 638 с.
6. А. Робачевский, С. Немнюгин, О. Стесик. Операционная система UNIX 2-е изд. - Санкт-Петербург, BHV-Петербург, 2010 г. - 656 с.

Интернет-ресурсы:

7. Интернет-курс «Основы операционной системы Unix» http://www.opennet.ru/docs/RUS/unix_basic
8. Интернет-курс «Представление текстовой информации в ЭВМ» http://samag.ru/archive/article/2653
9. Интернет-ресурс: домашняя страница пакета Notepad++ https://notepad-plus-plus.org/downloads/
10. Интернет-ресурс: домашняя страница пакета Putty http://www.chiark.greenend.org.uk/~sgtatham/putty/
11. Интернет-ресурс: домашняя страница пакета Winscp http://www/winscp.net

Оглавление