МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов, систем и сетей

Н.И. РОМАНЧЕВА

оглавление

СИСТЕМНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ.

ПОСОБИЕ

к выполнению курсовых работ

для студентов 3 курса

специальности 220100

дневного обучения

Москва- 2005

Рецензент доктор техн. наук, проф. А.А.Егорова

Романчева Н.И.

Системное программное обеспечение: Пособие к выполнению курсовых работ. - М.: МГТУ ГА, 2005.- 40 с.

Данное пособие издается в соответствии с учебным планом для студентов специальности 220100 дневного обучения.

Рассмотрены и одобрены на заседаниях кафедры 08.04.2005г. и методического совета 08.04.2005 г.

ВВЕДЕНИЕ

Курсовая работа по дисциплине "Системное программное обеспечение" выполняется студентами специальности 220100 в 7 семестре. В рамках курсовой работы должно быть разработано приложение для операционной системы UNIX, при выполнении курсовой работы используются знания, полученные студентами при изучении дисциплины "Системное программное обеспечение", а также дисциплин "Информатика", "Алгоритмические языки и программирование", "Операционные системы".

1 ЦЕЛИ И ЗАДАЧИ КУРСОВОГО ПРОЕКТИРОВАНИЯ

1.1 Целью курсового проектирования является приобретение практических навыков по разработке структуры приложения, алгоритмов и программ для их реализации с использованием языков C, C++ и PostScript для операционной системы UNIX/Linux.

1.2 Задачей курсовой работы является разработка приложения по заданным исходным данным:

- разработка командного интерпретатора;
- разработка программы-демона;
- построение грамматики для заданного языка и автомата для его распознавания;
- построение лексического анализатора;
- разработка программы обработки запроса ядром UNIX для символьного устройства;
- разработка программы обработки запроса ядром UNIX для блочного устройства;
- разработка программы вывода графиков на языке PostScript.

2 ОРГАНИЗАЦИЯ И ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

Курсовое проектирование является формой самостоятельной работы студента и выполняется по индивидуальному заданию.

Задание на курсовую работу выдается преподавателем на первой неделе 7 семестра, защита проводится на 16 и 17 неделях того же семестра. К защите представляется пояснительная записка. На защите демонстрируется выполнение программы, доклад - не более 7 минут с презентацией разработанной программной продукции с использованием средств MS PowerPoint.

В ходе выполнения курсовой работы студент консультируется с руководителем, назначенным кафедрой.

За правильность проектных решений, качество оформления работы, своевременность выполнения отдельных этапов и представления к защите отвечает студент.

2.1 Задание на курсовую работу

Задание на курсовую работу выбирается студентом по номеру группы и порядковому номеру студента в журнале.

2.2 Объем и содержание курсовой работы

Работа состоит из расчетно-пояснительной записки (РПЗ) и программы, представленной на гибком диске.

2.2.1 Техническое задание включает общие и специальные требования к программе.

2.2.2 Объем пояснительной записки составляет 30 - 40 машинописных страниц (формат А4). РПЗ должна быть написана четко и кратко, содержать пояснение к разработанному приложению, обоснование принятых решений. РПЗ должна включать следующие разделы:

1) Титульный лист (приложение А).

2) Бланк задания, подписанный преподавателем и студентом (приложение Б).

3) Содержание.

4) Перечень условных обозначений и сокращений в алфавитном порядке в виде списка, в котором слева приводится сокращение, справа - его расшифровка.

5) Основная часть РПЗ:

- Краткие теоретические сведения: анализ существующих программ подобного класса; особенности создания приложений для ОС UNIX/ Linux;
- Разработка структуры приложения;
- Разработка алгоритма решения задачи;
- Разработка программы;
- Проектирование интерфейса приложения (если это предусмотрено заданием на КР);
- Заключение;
- Список использованных источников;
- Приложения:
    - спецификация программного обеспечения (приложение В),
    - текст программы (приложение Г),
    - руководство пользователя (системного программиста) (приложение Д).

2.3 Последовательность выполнения работы

Курсовая работа разрабатывается в последовательности, соответствующей содержанию РПЗ (п.2.2.2).

Расчетно-пояснительная записка и графический материал оформляются в соответствии с требованиями ЕСКД и ЕСПД (Единая система конструкторской документации, Единая система программной документации).

Подготовленная и оформленная работа, прошедшая экспертизу на выполнение требований ЕСКД и ЕСПД представляется преподавателю не позднее, чем за неделю до защиты.

Защита работы происходит на 16 или 17 неделе семестра.

3 ВАРИАНТЫ ЗАДАНИЙ

Вариант 1

Написать и отладить программу на языке С, использующую системные вызовы для управления файлами в среде UNIX (тип файла - обычный, каталог).

Вариант 2

Написать и отладить программу на языке С, использующую системные вызовы для управления файлами в среде UNIX (тип файла - специальный, FIFO).

Вариант 3

Написать интерпретатор для языка высокого уровня. Проект должен включать:

- трансляцию исходной программы в промежуточное представление (в виде четверок иди кода стековой машины);
- интерпретацию промежуточного представления.

Порядок создания модулей:

1) разработка механизма таблицы символов;
2) создание интерпретатора для четверок;
3) создание лексического анализатора;
4) разработка семантических действий;
5) разработка синтаксического анализатора;
6) создание подпрограмм обработки ошибок;
7) вычисления.

Вариант 4

Разработать на языке PostScript программу вывода графика функции f(), которая строит n-периода, соединяя точки, расположенные с интервалом m^o , отрезками прямых. Размер поля рисования 10x4 см, график расположен в центре листа и обведен рамкой:

а) f =sin(2a+x), n=4, $m=5^o$

б) f =cos (x), n=3, $m=10^o$

в) f =sin(x)/2, n=4, $m=8^o$

Вариант 5

Написать процедуру на языке PostScript, которая вдоль ранее созданного графического пути строит штрих - пунктирную линию, состоящую из:

1) прямоугольных штрихов со срезанными краями и кружков между ними;

2) двукратное проведение штриховой линии с разными параметрами.

Вариант 6

Разработать на языке PostScript программу, позволяющую строить различные графические изображения с использованием функций cos() и sin(), не используя при этом встроенные функции языка. Разработка данной программы предполагает вывод изображений функций cos () и sin() одновременно в различных проекциях и пространствах (двумерном, трехмерном), различным цветом, со смещенной системой координат на 90^o. Изображения строятся по точкам, расположенным друг от друга с интервалом 5^o, 10^o, 36^o, 126^o, 181^o, 359^o, Для работы с цветом использовать процедуру изменения цвета изображений (по RGB- матрице).

Вариант 7

Разработать неинтерактивную программу-демон. Создаваемая программа должна:

1) отслеживать запуск программ в определенные моменты времени;
2) обеспечивать доступ к сервисам системы из сети;
3) обеспечивать получение и отправление почты.

При разработке программы необходимо учитывать:

- демон не должен реагировать на сигналы управления заданиями, посылаемые ему при попытке операции ввода-вывода с управляющим его терминалом. Начиная с некоторого момента, демон снимает ассоциацию с управляющим терминалом, но на начальном этапе запуска ему может потребоваться вывести то или иное сообщение на экран;

- необходимо закрыть все открытые файлы (файловые дескрипторы), особенно стандартные потоки ввода-вывода. Предполагается, что демон остается работать и после того, как пользователь покинул систему UNIX;

- необходимо снять его ассоциацию с группы процессов и управляющих терминалов, что позволит демону избавится от сигналов, генерируемых терминалом (SIFINT или SIGHUB), например, при нажатии определенных клавиш или выходе пользователя из системы;

- сообщения о работе демона следует направлять в специальный журнал с помощью функции syslog;

- необходимо изменить текущий каталог в корневой для возможного размонтирования примонтированной файловой системы.

Также следует:

- игнорировать сигналы, связанные с вводом-выводом на терминал фонового процесса;

- организовать собственную группу и сеанс, не имеющие управляющего терминала, выполнить порождение дочернего процесса для проверки, не является ли процесс лидером (предыстория запуска данной программы неизвестны);

-закрыть все возможные файловые дескрипторы;

- сменить текущий каталог на корневой;

- установит опции ведения журнала – каждая запись предваряется идентификатором PID демона. При невозможности записи в журнал сообщение выводить на консоль, источник сообщений определить как "системный демон", либо перенаправляется в соответствии со списком пользователей данной или удаленной системы.

Вариант 8

Разработать транслятор инфиксных выражений в постфиксную форму. Выражения разделены между собой точками с запятой и состоят из чисел, идентификаторов и операторов +, -, *, /, div, mod.

Вариант 9

Написать на языке PostScript файл для печати рекурсивных геометрических объектов:

а) ковра Серпинского;

б) кривых Гильберта;

в) кривых Коха;

г) кривых Пеано.

Вариант 10

Разработать на языке PostScript программу построения и вывода на печать графика заданной функции:

а) f= tg(1+2a)/(a+b);

б) f=exp(3a+2);

в) f=log(1+a/4)/3

Вариант 11

Разработать командный интерпретатор (КИ), распознающий внутренние и внешние команды. При создании КИ необходимо учитывать некорректные действия пользователя, которые могут привести к сбою программы. В качестве команд использовать как стандартные, так и созданные пользователем (например, команда test как результат выполнения нескольких команд: ls –l | grep). Общие технические требования: ОС - UNIX, язык программирования С, компилятор языка С.

Вариант 12

Разработать приложение под UNIX для ведения журнала событий:

а) спецификация интерфейса удаленных процедур log.x;

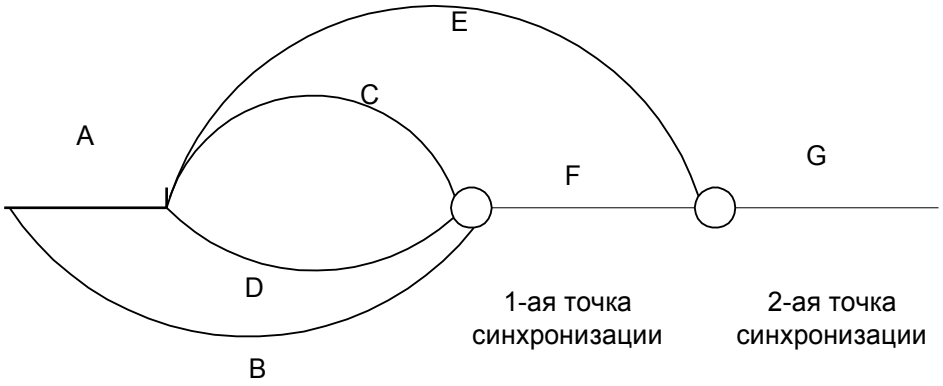
б) текст удаленной процедуры log.x;

в) текст головной программы клиента.

Вариант 13

Разработать программу создания параллельно взаимодействующих вычислительных процессов. Процесс А запускает задачи D, C, E. Время завершения задач D, C, E приблизительно одинаковое. Поток F запускается тем, кто завершается первым, но только после завершения 2-х остальных,

приходящие в точку синхронизации. Задача G запускается последним, закончившим работу потоком E или F. Схема взаимодействия отдельных потоков приведена на рисунке.



Вариант 14

Разработать алгоритм и написать программу для разбора и вычисления уравнения любой сложности. Использовать стековый алгоритм разбора с выводом стека на экран. При разработке интерфейса предусмотреть: поле для ввода выражения; окно, в котором отображается ход выполнения программы: окно для ввода переменных и поле для вывода результата вычислений.

Вариант 15

Разработать алгоритм и программу обработки запроса ядром UNIX для символьного устройства.

Вариант 16

Разработать алгоритм и программу обработки запроса ядром UNIX для блочного устройства.

4 ИНФОРМАЦИЯ ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

4.1 Демоны

Важную роль в работе операционной системы играют системные демоны. Демоны — это неинтерактивные процессы, которые запускаются обычным образом - путем загрузки в память соответствующих им программ (исполняемых файлов), и выполняются в фоновом режиме. Обычно демоны запускаются при инициализации системы и обеспечивают работу различных подсистем UNIX: системы терминального доступа, системы печати, системы сетевого доступа и сетевых услуг и т. п. Демоны не связаны ни с одним пользовательским сеансом работы и не могут непосредственно управляться пользователем. Большую часть времени демоны ожидают пока тот или иной процесс запросит определенную услугу, например, доступ к файловому архиву или печать документа.

Возможность терминального входа пользователей в систему, доступ по сети, использование системы печати и электронной почты, — все это обеспечивается соответствующими демонами. Некоторые демоны работают постоянно, пример такого демона — процесс init(), являющийся прародителем всех прикладных процессов в системе. Другими примерами являются cron(), позволяющий запускать программы в определенные моменты времени, inetd(), обеспечивающий доступ к сервисам системы из сети, и sendmail(), обеспечивающий получение и отправку электронной почты.

На рисунке 4.1 приведен основной алгоритм неинтерактивной программы. В ней вызываются функции, необходимые для выполнения программы, отслеживающей запущенные процессы. Данная функция организовывает собственную группу и сеанс, не имеющие управляющего терминала. Так как лидером группы и сеанса может стать процесс, если он еще не является лидером, а предыстория запуска данной программы неизвестна, необходима гарантия, что процесс не является лидером. Для этого порождаем дочерний процесс. Так как его PID уникален, то ни

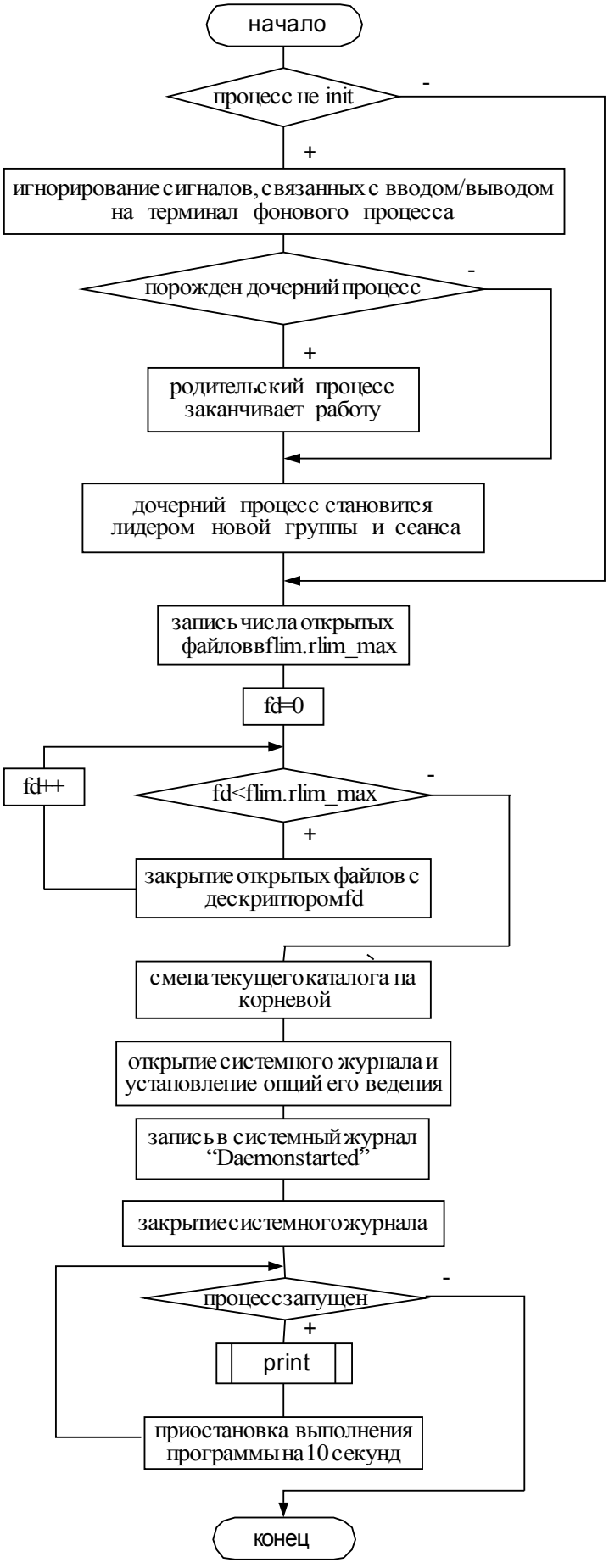


Рисунок 4.1 – Основной алгоритм программы

группы, ни сеанса с таким идентификатором не существует, а значит, нет и лидера. При этом родительский процесс немедленно завершает выполнение, поскольку он уже не нужен. Существует еще одна причина необходимости порождения дочернего процесса. Если демон был запущен из командной строки командного интерпретатора shell не в фоновом режиме, последний будет ожидать завершения выполнения демона, и таким образом, терминал будет заблокирован. Порождая процесс и завершая выполнение родителя, имитируется для командного интерпретатора завершение работы демона, после чего shell выведет свое приглашение. Далее закрываются открытые файлы, и выполняется запись в системном журнале. Для этого сначала устанавливается опции ведения журнала: каждая запись будет предваряться идентификатором PID демона. При невозможности записи в журнал сообщения будут выводиться на консоль. Каждые десять секунд вызывается функция, реализующая запись запущенных процессов в файл, например, sysmag.txt.

4.2 Язык PostScript

Язык PostScript не является языком программирования, он используется для описания страниц, подготовленных для печати на принтерах. Эти страницы могут включать не только текст, но и графические объекты, причем в самых разнообразных комбинациях. Вместе с тем PostScript обладает вычислительными возможностями, работает со строками текста, поддерживает ввод/вывод. Все графические объекты в языке PostScript, а так же шрифты являются векторными. В векторной графике положение и размеры объекта не зависят от размера точки изображения на конкретном устройстве печати и размера области изображения. Векторные изображения легко масштабируются, сдвигаются и поворачиваются. Изображения переводятся в растровый формат устройства вывода непосредственно перед печатью, что обеспечивают высокое качество печати. Координаты точек на странице задаются вещественными числами в полиграфических единицах длины – пунктах.

4.2.1 Элементы языка PostScript

PostScript-файл состоит из слов, которые отделяются друг от друга пробелами, символами табуляции, специальным символом конца строки и некоторыми другими специальными символами.

В словах кроме обычных, алфавитно-цифровых символов можно использовать специальные символы. Специальными символами являются скобки трех видов (,)> [,], { и }, символы <, >, / и X.

Строка может содержать любое количество слов, и любое слово может быть первым в строке. Исходный текст PostScript-файла может располагаться (форматироваться) так, чтобы подчеркнуть его логическую структуру, сделать удобочитаемым либо уменьшить размер файла. В PostScript используется обратная (польская) запись, при которой сначала указываются операнды, а затем оператор, выполняющий с ними действия.

Использование польской записи объясняется тем, что все операции в PostScript выполняются над операндами в стеке. Стек представляет собой специальную временную область памяти, организованную по принципу «последний вошел — первый вышел». Команда выбирает последнее значение, удаляя его из стека. После этого текущим становится значение, бывшее ранее предпоследним. Таким образом, находящиеся в стеке значения извлекаются из него в обратной последовательности. Интерпретатор PostScript последовательно считывает файл, интерпретируя очередное слово. Если слово представляет собой команду, она немедленно выполняется, а если это некоторый объект данных, он заносится в стек.

4.2.2 Типы данных

Основные типы данных языка PostScript;

- целое число со знаком;
- вещественное число;
- логическое значение;
- строка символов;
- процедура;
- массив произвольных объектов, в том числе и разнотипных;

- словарь.

Целые константы задаются в десятичном формате или в формате без знака с указанием основания системы счисления. Шестнадцатеричные константы записываются в угловых скобках.

Вещественные константы содержат десятичную точку или указатель десятичного порядка, без указателя порядка и с указателем порядка.

Логические константы принимают только два значения, true и false («истина» и «ложь»).

Строка записывается в круглых скобках. Скобки играют роль ограничителей. Строка может содержать пробелы, специальный символ конца строки, любые другие символы, а также экранированные последовательности, начинающиеся символом \.

Процедура — это группа команд, которая решает какую-то частную задачу или используется достаточно часто. В PostScript у процедуры нет ни имени, ни параметров. Процедура имеет тело и ссылку на него, этим она похожа на объекты других сложных типов. Текст процедуры содержится в фигурных скобках. При интерпретации процедуры ее операторы записываются в тело процедуры, а ссылки на нее — в стек операндов. В дальнейшем мы будем говорить, что в стек записывается процедура. Процедуру, записанную в стек, можно исполнить или записать в переменную, которая в таком случае будет указывать на тело процедуры, играя роль ее имени.

При исполнении процедуры в стек записывается значение, а затем выполняется умножение двух последних чисел из стека. Результат заносится в стек. Процедура записывает в стек операндов одно число, а использует для вычислений два. Первое число должно перед исполнением процедуры находиться в стеке операндов, оно играет роль аргумента. После выполнения процедуры в стеке операндов остается вещественное значение, оно и возвращается. Процедура может записать в стек несколько значений разных типов.

Во время исполнения интерпретатор PostScript помещает тело процедуры в стек исполнения, а исполнив, удаляет его из этого стека. Если из одной процедуры вызывается другая, первая приостанавливается, а тело второй заносится в стек исполнения и она становится текущей. После ее окончания и удаления из стека текущей становится вызывавшая процедура, которая продолжает работу с того места, где она была приостановлена. Это обстоятельство, а также метод хранения данных в стеке операндов позволяет использовать рекурсию, когда процедура вызывает на исполнение саму себя.

Массивом в программировании называют совокупность однотипных данных, которой присвоено общее имя. Массив в PostScript может состоять и из разнотипных объектов. Значения элементов массива записываются в квадратных скобках:

```
[ 1 (String) 1.0 ]
```

Символы квадратных скобок являются командами. Команда [записывает в стек операндов специальную метку, а команда] создает массив, содержащий элементы этого стека от текущего значения до ближайшей метки. Элемент, находящийся в стеке сразу после метки, имеет нулевой индекс. Между командами [и] можно исполнять любые команды со значениями, записанными в этот стек.

Словарь является сложным типом данных. Рассмотрим только применение словаря для хранения шрифта и изменения его кодировки. Каждый словарь имеет определенную емкость — максимальное число словарных записей, которое определяется при его создании. Словарная запись состоит из имени и значения. Значением может быть объект любого типа.

Команды PostScript хранятся в специальных словарях, находящихся в стеке словарей с самого начала, которые не могут быть удалены. Это словарь ошибок, системный и пользовательский словари. Данные в системном словаре и словаре ошибок нельзя изменить, эти словари закрыты для записи. Пользовательский словарь изначально является текущим. Можно создавать новые словари, помещать их в стек словарей и удалять из него. Текущим является последний словарь, записанный в этот стек.

4.2.3 Структура PostScript-файла

PostScript-файл состоит из трех частей: пролога, тела и эпилога. В прологе содержатся описания подпрограмм и данные, необходимые для печати документа. В теле PostScript-файла содержатся только команды, формирующие страницы документа. Эпилог не содержит никаких команд.

Также важную роль при написании программы имеют комментарии, так как по комментарию, расположенному в первой строке файла, ОС UNIX и прикладные программы определяют его тип.

Структурные комментарии используются для выделения элементов логической структуры PostScript-файла и интерпретируются программами просмотра/преобразования.

Структурные комментарии делятся на три группы:

- комментарии в заголовке программы (перед прологом);
- комментарии в теле программы;
- завершающие комментарии (в эпилоге).

Для принтера, печатающего файл, комментарии не играют никакой роли, подобно тому, как комментарии в обычном языке программирования не обрабатываются компилятором. Комментарий начинается символом % и продолжается до конца строки.

Для того чтобы подсистема печати или прикладная программа смогли правильно определить тип файла, его первая строка должна начинаться комментарием:

%!

Данный комментарий является обязательным. После символов %! в той же строке может следовать текст PS-Adobe-N.M. Цифры N.M (например, 1.0) определяют версию соглашения о структурных комментариях (это специальный документ, содержащий правила оформления структурных комментариев).

Структурные комментарии начинаются символами %%. Если программа содержит такие комментарии, она соответствует договоренностям о структуре PostScript-файлов. После %% без пробела следуют

зарезервированные (ключевые) слова. Регистр букв в них имеет значение. При необходимости с ключевым словом может быть указано значение, которое отделяется от ключевого слова двоеточием и пробелом.

Комментарий %%DocumentFonts позволяет программам обработки загружать в принтер необходимые шрифты, если они еще не содержатся в его памяти. В комментарии %%PageFonts перечисляются только те шрифты, которые используются при печати данной страницы.

Комментарий %%BoundingBox определяет положение изображения на странице. Четыре целых числа — это координаты левого нижнего и правого верхнего углов прямоугольника, содержащего изображение. В многостраничном документе прямоугольник должен охватывать все изображения. Этот комментарий важен для программ, которые включают изображение из одного файла в другой файл. Если точно определить размер изображения невозможно, следует указать заведомо большие значения, например соответствующие размеру листа бумаги формата A4:

%%BoundingBox: 0 0 595 842

Комментарием %%EndProlog завершается раздел описаний PostScript-файла и начинается тело документа.

Комментарий %%Page открывает ту часть файла, которая описывает печать очередной страницы. У него два аргумента — метка страницы и ее номер. Метка представляет собой последовательность любых символов, кроме пробела, и на печать не выводится. Номер страницы задается числом, начиная с 1.

Комментарий %%PageFonts, если он имеется, должен идти сразу после % %Page. Очередной комментарий %%Page обозначает конец предыдущей страницы и начало следующей. Конец последней страницы отмечается комментарием %%Trailer.

В языке PostScript нет четкого деления файла с описанием документа на раздел описания и раздел операторов (тело программы), как, например, в языках программирования Pascal или Фортран. Определения переменных в PostScript могут чередоваться с командами формирования изображения.

Текст в PostScript тоже является изображением. Важно лишь, что правильно составленная программа должна иметь правильную структуру. Определения общих для всех страниц документа переменных и процедур следует выносить в пролог.

На рисунке 4.2 представлена структура программы, написанная на языке описания страниц – PostScript.

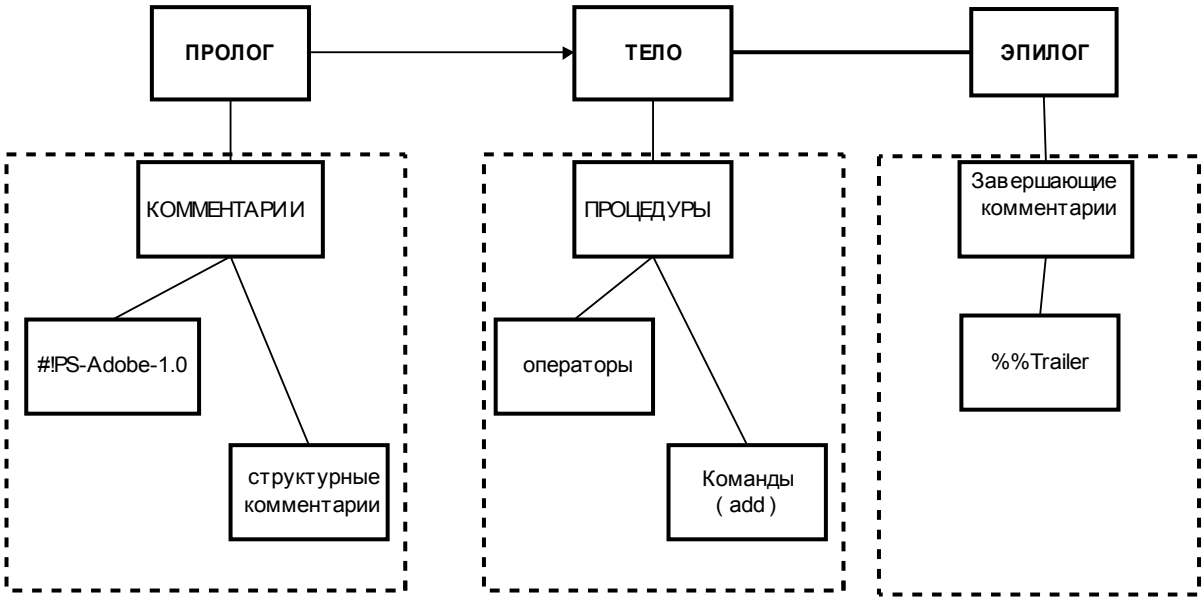


Рисунок 4.2 - Структура программы

4.2.4 Обзор универсальных команд

Для описания команд используется следующая нотация:

аргумент1 аргумент2, ... команда ® результат1 результат2, ... % комментарий

Имя команды выделяется специальным шрифтом. Слева от имени описывается состояние стека до исполнения команды, а справа от стрелки — после ее исполнения. Минус вместо аргумента или результата обозначает отсутствие значения. Данные обозначаются словами, производными от названий типов или математических обозначений, характеризующих операцию. Такая запись приближена к записи текста программы. Например, описание операции сложения выглядит так:

p q add -> p+q

Запись команды в тексте PostScript-файла может быть такой:

4.0 5.0 add

или

17 10 add

Целые числа обозначаются буквами i, j, n, вещественные числа — x, y, z, логические (булевы) значения — буквой Ь, координаты — x, y. Буквами p, q, r, s обозначаются любые числа, как целые, так и вещественные, если их тип не имеет значения. Строки, массивы и процедуры указываются вместе со скобками, в которых они определяются.

PostScript имеет команды общего назначения для работы с данными (присваивание, работа со стеком), операторы вызова подпрограмм, ветвления и циклы. Операторов безусловного перехода и меток нет. Имеются команды ввода/вывода, но при печати на принтере они теряют смысл, так как PostScript-файл передается в принтер и обрабатывается в нем автономно, без связи с компьютером.

При интерпретации команды интерпретатор PostScript ищет ее имя в стеке словарей. Если имя найдено, соответствующее значение либо заносится в стек операндов (если это не процедура), либо исполняется (если это процедура). Если имя не найдено ни в одном из словарей, возникает ошибка и интерпретация PostScript-файла завершается. Имена стандартных команд представляют собой имена процедур, записанные в системном словаре. При необходимости эти имена можно переопределить. Команда load выполняет поиск указанного имени, но в любом случае заносит найденное значение в стек. Эту команду можно применять для того, чтобы использовать описанную ранее процедуру в качестве тела цикла или условного оператора.

Условный оператор if удаляет из стека операндов процедуру и логическое значение, после чего, либо исполняет процедуру, либо нет. Процедура может использовать значения в стеке операндов, записанные до того, как туда было записано логическое значение. Условный оператор ifelse удаляет из стека операндов обе процедуры и логическое значение, после чего исполняет одну из процедур. В этом случае обе процедуры должны одинаково использовать старые значения в стеке и оставлять там одинаковое количество новых значений.

Оператор цикла for удаляет из стека операндов все свои 4 аргумента и устанавливает значение параметра цикла равным $p_{нач}$. После этого он циклически выполняет следующие действия: проверяет, не вышло ли текущее значение параметра за допустимую границу $s_{кон}$, помещает значение параметра цикла в стек операндов, выполняет процедуру, увеличивает значение параметра на величину $q_{шаг}$. Шаг может быть как положительным, так и отрицательным. Если начальное значение больше конечного при положительном шаге или меньше конечного при отрицательном шаге, процедура не выполняется. Процедура может использовать предыдущие значения в стеке, а также значение параметра цикла. После окончания цикла стек не восстанавливает автоматически свое прежнее состояние, что может привести к его переполнению, поэтому, в частности, необходимо удалять из стека операндов очередное значение параметра цикла.

В таблице 1 приведен обзор некоторых универсальных команд.

Таблица1

Описание команды	Примечание
p q add->p+q	-
p q sub->p-q	-
p q mul->p*q	-
p q div->p/q	-
i j idiv->[i/j]	Целая часть от деления. Аргументы только целого типа.
i j mod->imodj	Остаток от деления. Аргументы только целого типа.
p neg->-p	Изменение знака
p abs->|p|	Модуль числа
p cvi->i	Преобразование к целому типу. У вещественного числа отбрасывается дробная часть.
p cvr->x	Преобразование к вещественному типу.
p round ->q	Округление числа. Тип результата совпадает стипом аргумента.
(строка) cvi-> i	Преобразование символьного представления числа в целое
(строка) cvr->x	Преобразование символьного представления числа в вещественное
Функции	
α^{0} sin → $\sin(\alpha^{0})$	Угол задается в градусах
p sqrt → $\sqrt{p}$	Квадратный корень
p ln → $\ln(p)$	Натуральный логарифм

1	2
p log→ lg(p)	Десятичный логарифм
-rand->i	Генератор псевдослучайных чисел в диапазоне от 0 до 231-1
Отношения	
p q eq->p=q	В стек помещается логическое значение true или false
p q ne-> p≠ q	
p q le->p≤ q	
p q lt->p<q	
p q gt->p>q	
Логические операции	
b1 b2 and->b1∧ b2	Логическое И
b1 b2 or->b1∨ b2	Логическое ИЛИ
b1 b2 xor->b1=b2	Исключающее ИЛИ
b not->$\bar{b}$	Логическое отрицание

4.2.5 Графический контекст

В операторах вывода графики используются неявные параметры, влияющие на их выполнение. Совокупность этих параметров называется графическим контекстом. Графический контекст составляют положение, ориентация и масштаб системы координат, толщина и стиль рисования линии, область рисования, текущий путь построения линии и некоторые другие. В PostScript имеются команды для работы с графическим контекстом.

Среди них команды, которые заносят в специальный стек графических контекстов текущий графический контекст и могут восстановить ранее записанное состояние. Это позволяет изменять параметры вывода одной части рисунка и возвращаться к предыдущему состоянию перед выводом другой его части. Команда gsave выполняет запись в стек графических контекстов, а команда grestore считывает из этого стека графический контекст, делая его текущим. Таким образом, все изменения параметров рисования, произведенные с момента сохранения контекста, перестают действовать на последующие команды вывода графических объектов. Команда grestoreall восстанавливает самое первое состояние, записанное в стек командой gsave.

Более «мощные» команды save и restore кроме сохранения графического контекста сохраняют еще и область памяти виртуальной ЭВМ, содержащую все переменные, и восстанавливают как графический контекст, так и значения переменных. Эти команды можно использовать для предотвращения побочных эффектов при печати страниц, если среди команд печати встречаются команды изменения каких-либо объектов. Команда grestoreall восстанавливает последний графический контекст, сохраненный командой save, или, если таких команд не было, самый первый (верхний в стеке), сохраненный командой gsave. Образ памяти, создаваемый командой save, рекомендуется хранить в стеке операндов и использовать команду restore, предварительно убрав из стека все записанные туда позже операнды.

Команды save/restore рекомендуется использовать в начале и в конце описания каждой страницы.

Такой стиль программирования позволит в начале каждой страницы получать тот графический контекст и переменные, которые были определены в прологе программы. Использование в начале/конце описания страниц команд gsave/grestore позволяет сохранять графический контекст, но не освобождает память от определенных, но уже не используемых значений. Эти команды выполняются быстрее, чем save/restore, поэтому при печати больших файлов они могут оказаться более эффективными. В таблице 2 приведены команды для работы с графическими объектами.

Таблица 2

Описание команд	Примечание
Сохранение и восстановление графического контекста	
-gsave->-	Запись текущего графического контекста в стек графических контекстов
-grestore->	Восстановление текущего графического контекста из сохраненного командой gsave
-save-> образ памяти метка	Сохранение текущего графического контекста в стеке и запись в стек операндов образа памяти
образ памяти restor->-	
Изменение системы координат	
x y translate->	Перенос начала координат
α ^{0} rotate->	Поворот координатных осей

1	2
Sx Sy scale->	Изменение масштаба осей координат
Создание графического пути	
-newpath->	Инициализация нового графического пути
-closepath->-	Замыкание участка графического пути из текущей тички в начальную точку пути
x y moveto->-	Установка текущей точки без проведения линии
-currentpoint->x y	Определение координат текущей точки
x y lineto->-	Добавление отрезка прямой линии
Использование графического пути для вывода фигур	
-stroke->-	Построение линии вдоль графического пути
-fill->	Закрашивание текущем цветом фигуры, ограниченной замкнутым графическим путем
-pathbbox->	Определение координат прямоугольника, охватывающего текущий путь
-clippath->-	Создание нового графического пути вокруг всех областей
-clip->	Ограничение текущей области рисования фигурой, определенной замкнутым графическим путем

4.2.7 Рисование и закраска фигур

Отличительным аспектом PostScript является то, что даже текст - это разновидность графики. Первой задачей будет рисование линий для создания изображения.

Основные шаги рисования и закраски фигур:

- Начать путь оператором newpath;
- Собрать путь из отрезков и кривых (не обязательно смежных);
- Нарисовать линию оператором stroke или закрасить оператором fill.

Эта последовательность действий может быть изменена для получения более сложных результатов.

Рисование прямоугольника

Нарисуем прямоугольник на расстоянии в дюйм от сторон левого нижнего угла страницы. Начнем с функции, переводящей дюймы в единицы измерения PostScript - пункты (один пункт равен 1/72 дюйма). Осуществить такое преобразование просто - достаточно умножить число дюймов на 72:

```
/inch {72 mul} def
```

Начинаем новую линию и помещаем текущую точку на расстояние в дюйм от границ:

newpath

1 inch 1 inch moveto

К этому моменту линия состоит из одной точки с координатами (72, 72). Добавим стороны с помощью оператора lineto. Этот оператор добавляет к пути отрезок, соединяющий текущую точку и точку, координаты которой находятся на стеке. Координаты конца отрезка становятся новыми координатами текущей точки. Итак, добавим три стороны квадрата:

2 inch 1 inch lineto

2 inch 2 inch lineto

1 inch 2 inch lineto

Получившуюся линию можно замкнуть кратчайшим отрезком. Это делается оператором closepath. Этот оператор особенно полезен при закраске фигур. Теперь полученную линию можно нарисовать оператором stroke. Оператор showpage закончит вывод страницы на печать:

closepath

stroke

showpage

Закраска фигур

Сначала создается путь, но вместо вызова оператора stroke вызывается оператор fill, который заполняет путь текущим цветом. Применение fill вместо stroke в приведенном примере даст закрашенный квадрат вместо контура.

Вставка текста

Вставка текста состоит из следующих основных шагов:

- Выбрать необходимый шрифт;
- Сделать текущей точку, в которую будет помещен левый нижний угол текста;
- Передать строку для печати оператору 'show'.

Оператор 'show' - это простейший опреатор для вывода строки. Его аргументом является строка, которую он выводит текущим шрифтом. Вывод происходит, начиная с текущей точки, которая становится левой нижней

точкой по отношению к тексту. После того как текст был выведен, текущей становится точка соответствующая нижнему правому краю строки.

Ниже приведен текст программы для вывода графика функции f=sin(x)\2.

```
%!PS-ADOBE-1.0
%%Title: график Функции f=sin(x)\2
%%Creator: Иванова Анна
%%Pages: 1
%%BoundingBox: 0 0 595 842
%%EndComments
 /cm { 72.0 mul 2.54 div } def %перевод см в пункты
 /x0 21.0 2 div 5.0 sum cm def %координаты нижнего
 /yo 29.7 2 div 2.0 sum cm def %левого угла графика
%%EndProlog
%%Page: 1 1
gsave
x0 y0 translate  %сдвигаем начало координат к рамке
newpath
0 0 moveto       %ввоводим рамку размером 10см x 4см
10 cm 0 rlineto
0 4 cm rlineto
-10 cm 0 rlineto
closepath
.3 setlinewidth %толщина рамки 0.3 пункта
stroke          %рисуем линию вдоль сторон рамки
newpath
0 8 1440        %заголовок цикла: от 0^0 до 8
{
  /a exch def   %локальная переменная
  a 144.0 div   %стек: это число в диапазоне 0…10см
cm     %стек: x координата лежит в диапазоне %0…10см
  a sin
```

```
  a 2 div
  1.0 add
  2.0 mul
 cm %стек: x y %координата y лежит в %интервале 0…4 см
  a 0 eg
    { moveto } %переходит в первую точку % стек:x y
    { lineto }
  ifelse
  } for          % оператор цикла
1 setlinewidth % график выводится жирной линией
stroke           % рисует линию вдоль прямой
grestore
showpage
%%Trailer
```

На рисунке 4.3 приведен результат выполнения программы.

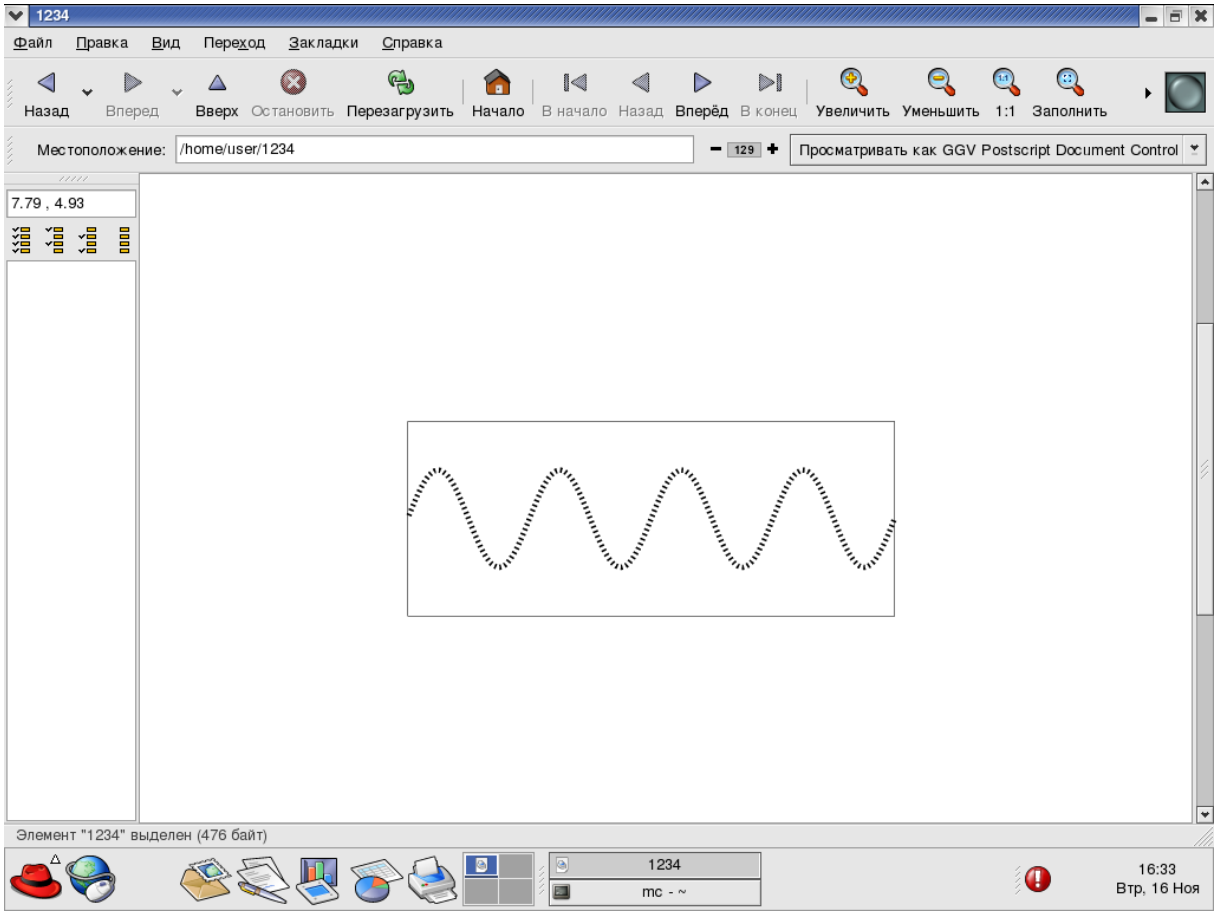


Рисунок 4.3 – Вид окна с результатами выполнения программы

4.3 Командный интерпретатор

Командный интерпретатор является одной из важнейших программ, обеспечивающих диалог пользователя с системой. Он запрашивает у пользователя команду и анализирует ее. Если команда является внутренней по отношению к командному интерпретатору, то он реализует ее своими средствами (например, команда смены директории – cd – реализуется функцией cd ()). Если же введенная команда не является внутренней, он запускает эту команду на выполнение (функция execvp()). В случае некорректной команды, выводится сообщение об ошибке.

Все современные системы UNIX поставляются по крайней мере с тремя командными интерпретаторами: Bourne shell (/bin/sh), C shell (/bin/csh) и Korn shell (/bin/ksh). Существует ещё несколько интерпретаторов, например Bourne-Again shell (bash), со сходными функциями.

В UNIX реализуется следующий сценарий работы в системе:

•При включении терминала активизируется процесс getty (1M), который является сервером терминального доступа и запускает программу login(1), которая, в свою очередь, запрашивает у пользователя имя и пароль.

•Если пользователь зарегистрирован в системе и ввёл правильный пароль, login(1) запускает программу, указанную в последнем поле записи пользователя в файле /etc/passwd. В принципе это может быть любая программа, но в нашем случае – это командный интерпретатор shell.

•Shell выполняет соответствующий командный файл инициализации, и выдаёт на терминал пользователя приглашение. С этого момента пользователь может вводить команды.

•Shell считывает ввод пользователя, производит синтаксический анализ введённой строки, подстановку шаблонов и выполняет действие, предписанное пользователем (это может быть запуск программы, выполнение внутренней функции интерпретатора) или сообщает об ошибке, если программа или функция не найдены.

•По окончании работы пользователь завершает работу с интерпретатором, вводя команду exit, и выходит из системы.

Основной алгоритм программы, реализующей функции командного интерпретатора, представлен на рисунке 4.4. В ней осуществляется вывод на экран строки с текущей директорией и приглашением командного интерпретатора, запрашивающим команду. После ввода пользователем команды, вызывается функция (translate), которая делит введенную команду на имя команды и ее аргументы, возвращая при этом константу в соответствии с именем команды. Потом с помощью оператора *switch* анализируется возвращенная константа, и выполняются соответствующие действия.

Translate – функция разбора строки команды, введенной пользователем с клавиатуры, на имя команды и ее аргументы. Функции передается параметр – строка команды, возвращаемые значения – имя команды в переменной *command,* список аргументов в массиве *p* и константа, определяющая команду.

Алгоритм функции translate представлен на рисунке 4.5.

Sozd – функция, выполняющая запуск внешней команды. Функция создает дочерний поток, в котором выполняется команда, возвращает результат выполнения команды. В случае если команда не найдена, выдается сообщение об ошибке.

Алгоритм функции sozd представлен на рисунке 4.6.

F – функция выполняет действия аналогичные команде cat>1.txt, являющейся внешней, т.е. производит запись введённой информации в файл. Весь ввод с консоли направляется в файл 1.txt. Для вызова функции следует нажать клавишу «f», для завершения записи в файл нажать сочетание клавиш «Ctrl+Z». Файл создается с именем 1.txt в текущей директории.

Алгоритм функции F представлен на рисунке 4.7.

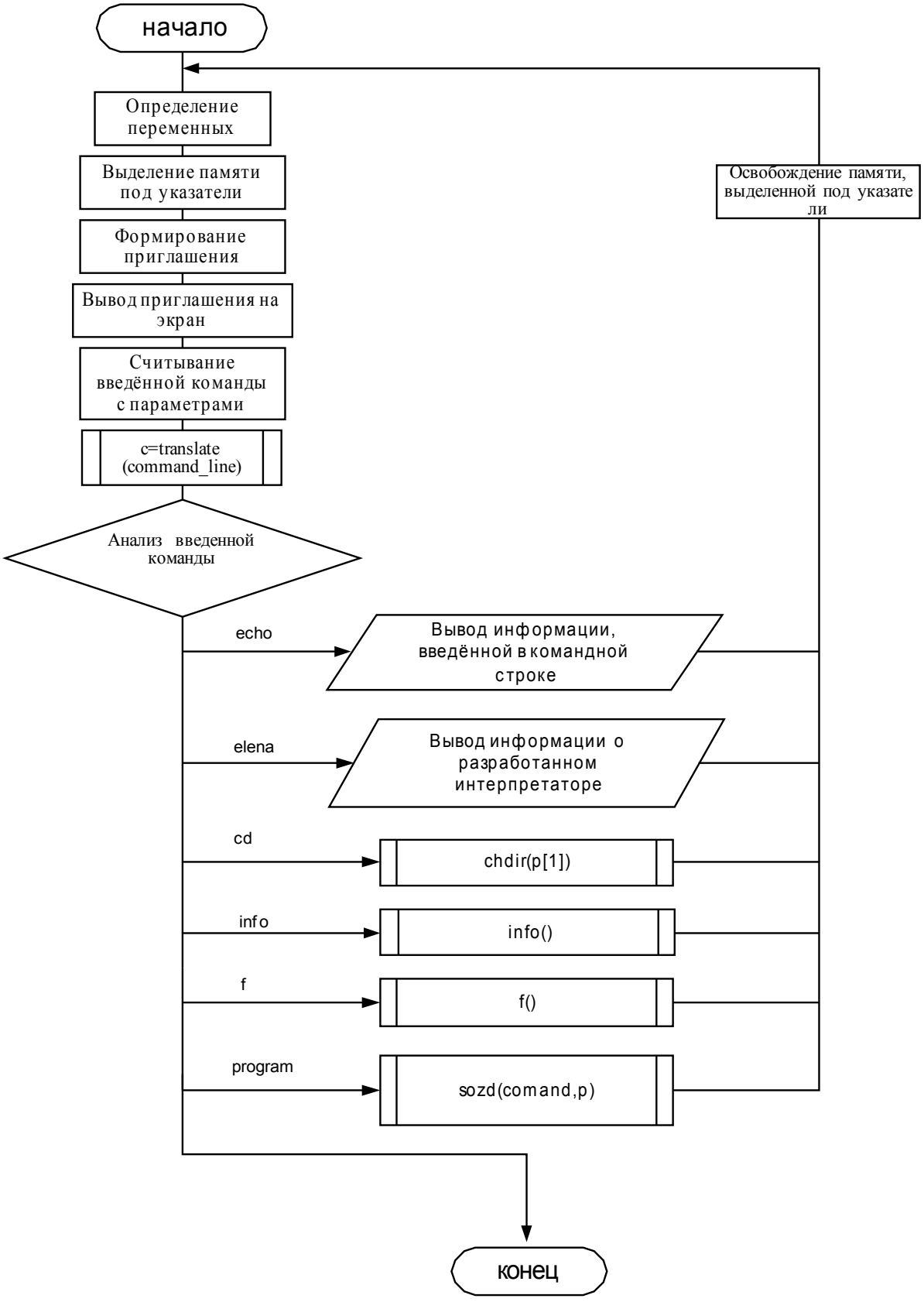


Рисунок 4.4 - Основной алгоритм программы

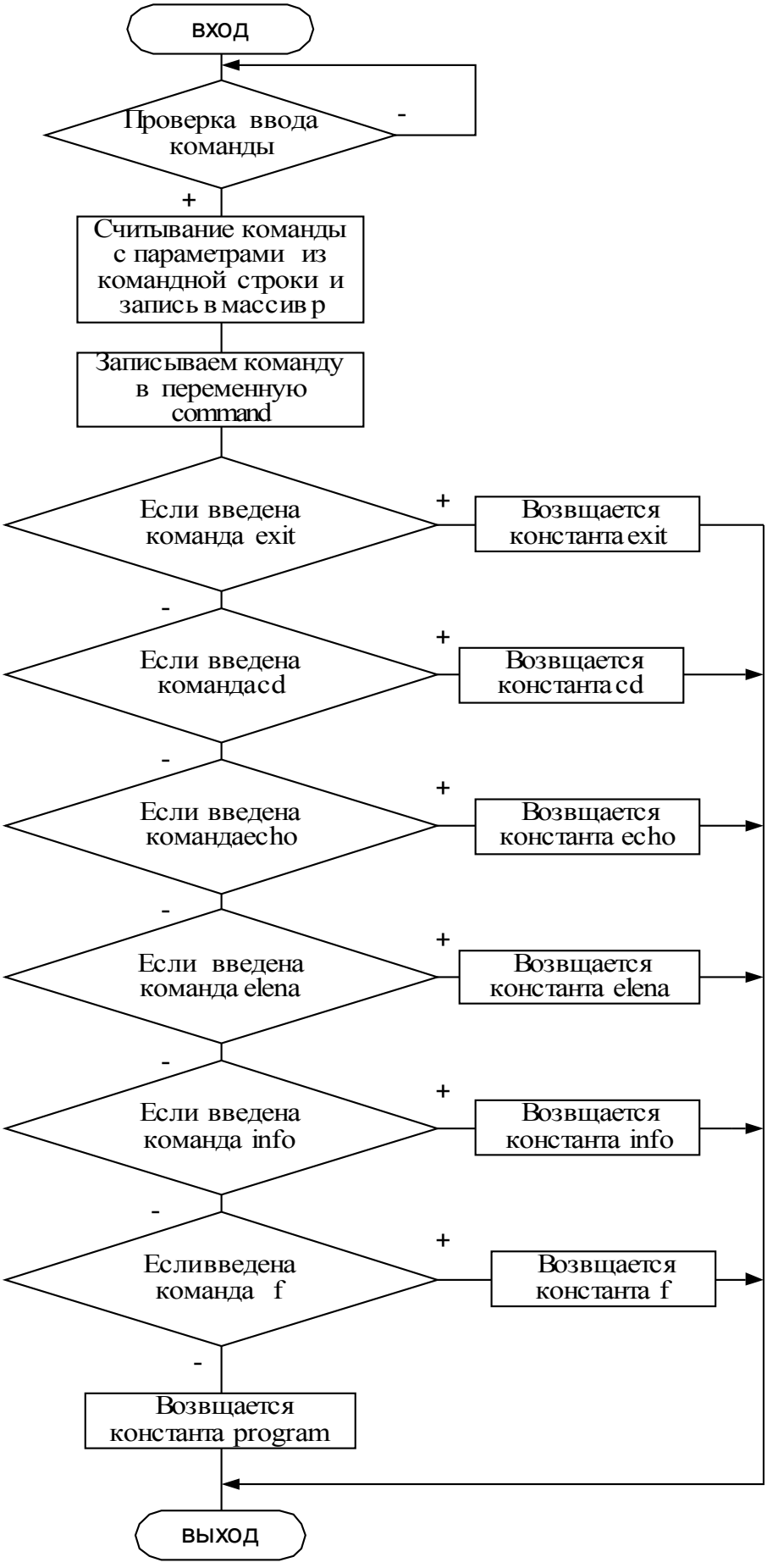


Рисунок 4.5 – Алгоритм функции translate

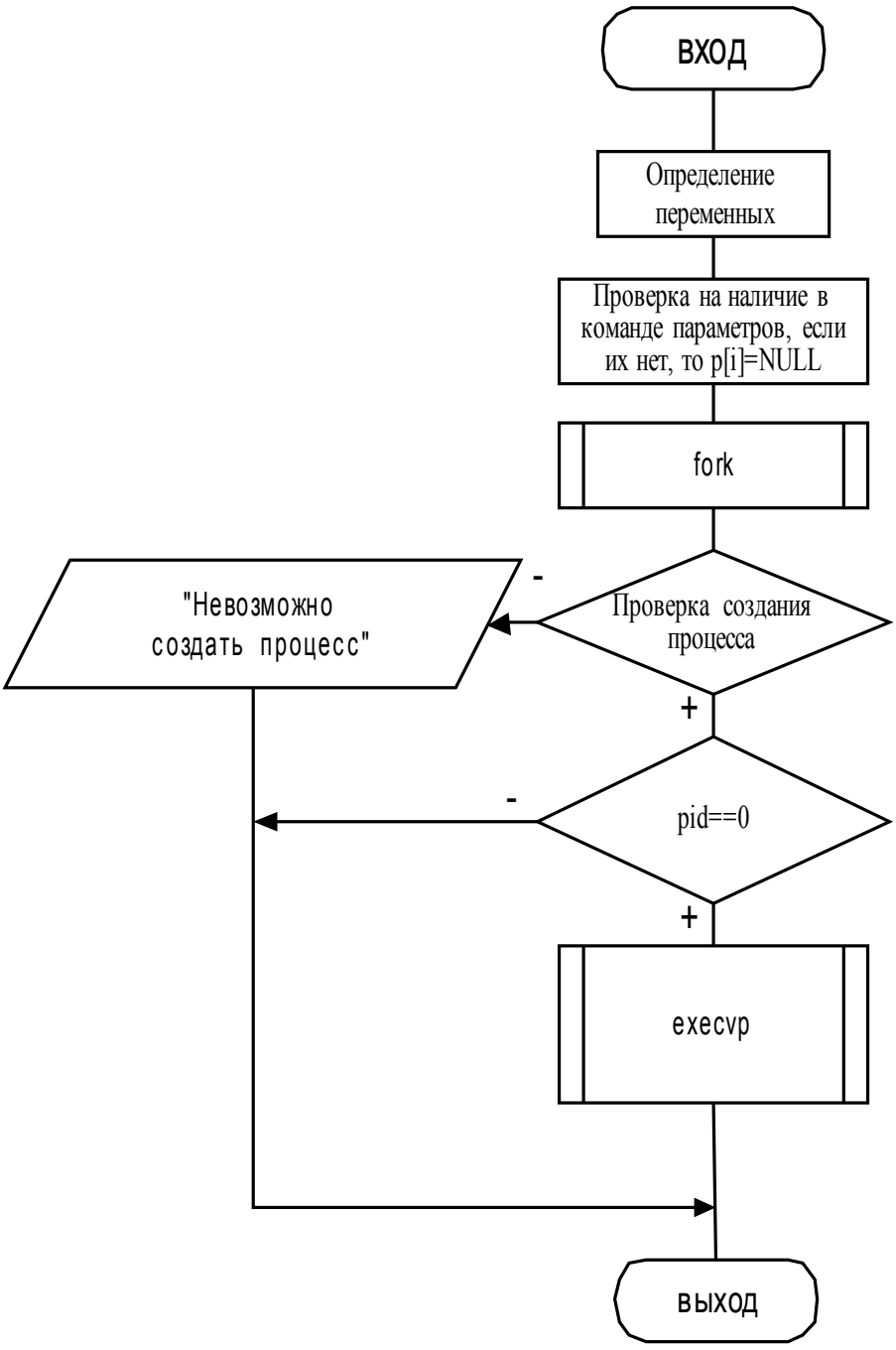


Рисунок 4.6 – Алгоритм функции sozd

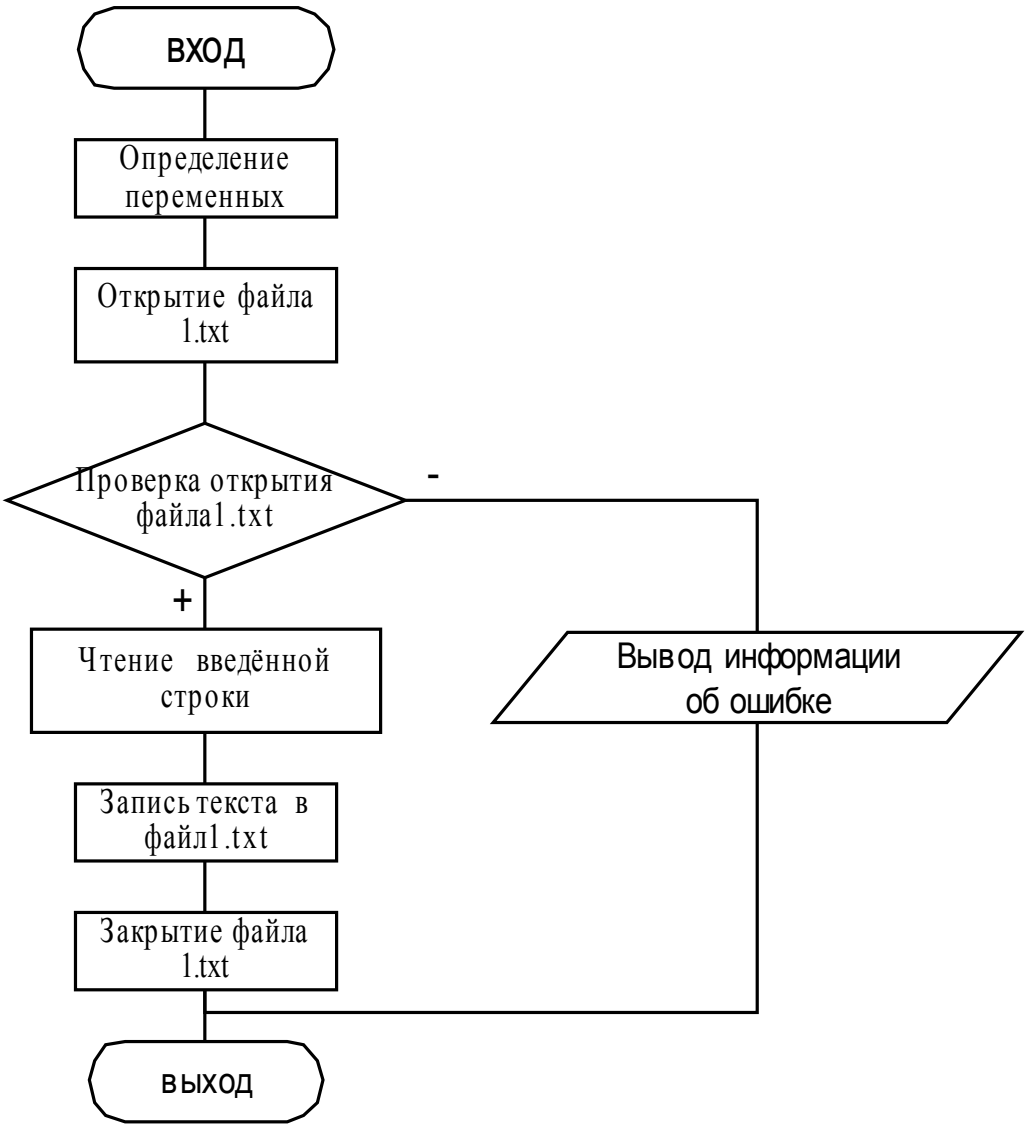


Рисунок 4.7 – Алгоритм функции F

ЛИТЕРАТУРА

1) Робачевский А. М. Операционная система Unix. - СПб: БХВ-Петербург, 2001.
2) Стахнов А. А. Linux. – СПб.: БХВ-Петербург, 2003.
3) Шоу А. Логическое проектирование операционных систем, Пер. с англ.- М: Мир, 1981.
4) Грис Д. Конструирование компиляторов для ЦВМ.-М.: Мир,1975
5) Бек Л. Введение в системное программирование.-М.: Мир, 1988
6) Немнюгин С., Чаунин М., Камолкин А. Эффективная работа: UNIX.-СПб.: Питер, 2003.
7) Пособие по оформлению курсовых и дипломных проектов и работ /Под ред. В.В. Соломенцева - М.: МГТУГА, 2002.

СОДЕРЖАНИЕ

ПРИЛОЖЕНИЕ А

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов, систем и сетей

Курсовая работа
защищена с оценкой

_____

_____
(подпись преподавателя, дата)

КУРСОВАЯ РАБОТА
по дисциплине “Системное программное обеспечение”
Вариант №14

Тема: “Разработка командного интерпретатора

Курсовая работа
допущена к защите

_____

(подпись преподавателя, дата)

Выполнила ст. группы ЭВМ 3-1
Апалькова Елена Александровна
(Ф.И.О.)
Руководитель:
доцент, к. т. н., Романчева Н.И
(звание, степень Ф.И.О.)
.

МОСКВА - 2005

ПРИЛОЖЕНИЕ Б

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов, систем и сетей

З А Д А Н И Е

на выполнение курсовой работы
по дисциплине
«Системное программное обеспечение»
вариант № 1

Исходные данные:

Разработать транслятор инфиксных выражений в постфиксную форму. Выражения разделены между собой точками с запятой и состоят из чисел, идентификаторов и операторов +, -, *, /, div, mod.

Задание выдано « »_____2005 г. (подпись преподавателя)

Задание получил _____ (подпись студента)

Приложение В
Спецификация

Обозначения	Наименование	Примечание
	Документация	
КР021045 12	Разработка программы	
	вывода графика функции	
	f=sin(x)/2. Текст	
	программы.	
КР021045 34	Разработка программы	
	вывода графика функции	
	f=sin(x)/2. Руководство	
	оператора.	

Приложение Г

УТВЕРЖДЕН
КР021045 12

Разработка программы для вывода графика функции f=sin(x)/2

Текст программы

К021045 12

Листов 12

2005

Приложение Д

УТВЕРЖДЕН
КР021045 34

Разработка программы для вывода графика функции f=sin(x)/2

Руководство оператора

К021045 34

Листов 18

2005