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)

назад

КУРСОВАЯ РАБОТА
по дисциплине "Системное программное обеспечение"
Вариант №14

Тема: "Разработка командного интерпретатора, распознающего внутренние и внешние команды"

Выполнила студентка группы ЭВМ 3-1
Апалькова Елена Александровна
(Ф.И.О.)

Руководитель:
доцент, к. т. н., Романчева Н.И.
(звание, степень, Ф.И.О.)

МОСКВА - 2004

Аннотация

В данной курсовой работе разработана программа, представляющая собой командный интерпретатор, распознающий внутренние и внешние команды.

В пояснительной записке проведён анализ программы, разработана её структура, алгоритм решения задачи и алгоритм программы, приведены требования для работы программы и действия, необходимые для её запуска и функционирования. Текст программы написан на языке С.

Содержание

1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Назначение программы

Разработать командный интерпретатор, распознающий внутренние и внешние команды.

При разработке программы необходимо учесть некорректные действия пользователя, которые могут привести к сбою программы.

1.2 Общие технические требования

- Процессор x386 и выше
- Наличие ОС UNIX
- Свободное место на диске не менее 16 Кб
- Операционная система семейства Unix
- Среда программирования ОС Unix
- Язык программирования С
- Компилятор языка С

2 АНАЛИЗ РАЗЛИЧНЫХ КОМАНДНЫХ ИНТЕРПРЕТАТОРОВ

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.

Командный интерпретатор занимает важное место в операционной системе UNIX, прежде всего, благодаря следующим обстоятельствам:

1) Первая программа, с которой по существу начинается работа пользователя, - shell. 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.

2) Командный интерпретатор является удобным средством программирования. Синтаксис языка различных команд интерпретаторов несколько отличается, в качестве базового мы рассмотрим командный интерпретатор Bourne. С помощью shell вы можете создавать сложные программы, конструируя их, как из кирпичиков, из существующих утилит UNIX. Программы на языке shell часто называются скриптами или сценариями. Интерпретатор считывает строки из файла-скрипта и выполняет их, как если бы они были введены пользователем в командной строке.

3) При входе пользователя в систему его инициализационный скрипт, выполняющий несколько функций: установку пути поиска программ, инициализацию терминала, определение расположения почтового ящика. Помимо этого может быть выполнен целый ряд полезных действий, - например, установка приглашения. Скорее всего, придётся «покопаться» в этом скрипте, по крайней мере, чтобы добавить необходимые пути поиска. Инициализационный скрипт находится в домашнем каталоге пользователя.

Для разных командных интерпретаторов используются различные скрипты инициализации:

Командный интерпретатор	**Скрипт инициализации**
Bourne shell (sh)	.profile
C shell (csh)	.login, .cshrc
Korn shell (ksh)	.profile, .kshrc
Bourne-Again shell (bash)	.profile, .bashrc

Скрипты .profile и .login выполняются при первом входе в систему. Скрипты .cshrc, .kshrc и .bashrc выполняются при каждом запуске интерпретатора.

4) Основная инициализация операционной системы происходит в результате выполнения скриптов shell. Если понадобится модифицировать процесс инициализации (например, добавить новый системный сервис), то придётся заглянуть в эти скрипты.

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.

Разработанный в курсовой работе командный интерпретатор предназначен для организации диалога пользователя с системой. Пользователь, введя команду, может рассчитывать на выполнение этой команды. Если команда является внутренней, то командный интерпретатор реализует её посредством стандартных библиотечных функций, системных вызовов или внутренних алгоритмов. Если команда является внешней, т.е. реализованной в виде внешней утилиты, то он пытается запустить эту утилиту на выполнение, полностью возлагая ответственность за наличие данной утилиты на систему. В случае невозможности или некорректности выполнения внешней или внутренней команды, выдаётся сообщение об этом с указанием ошибки. Если в команде указаны параметры, они учитываются при выполнении команды. Допускается указание не более 10 параметров. В качестве строки приглашения выводится *« -<-<-@ »*, что по внешнему виду напоминает розочку.

Разработанный командный интерпретатор реализует выполнение 6 внутренних команд: *elena, info, f, echo, cd, exit.*

3 РАЗРАБОТКА СТРУКТУРЫ ПРОГРАММЫ

В соответствии с алгоритмом, приведенном на **рисунке 2**, разработана программа, которая реализована на языке программирования С.

Структура программы представлена на **рисунке 1**.

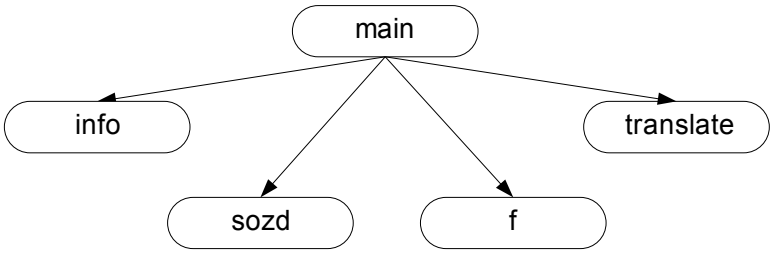

Рисунок 1 – Структура программы

В **таблице 1** приведены основные функции, используемые в данной программе.

Таблица 1 – Функции программы

Название функции	Тип возвращаемого результата	Выполняемые действия
main	void	вывод на экран приглашения и запрос команды
info	void	вывод информации о машине/системе
sozd	void	выполнение внешней команды
f	void	весь ввод с консоли направляет в файл
translate	int	разбор строки с командой и аргументами и её анализ

4 РАЗРАБОТКА АЛГОРИТМОВ ПРОГРАММЫ

4.1 Описание основного алгоритма программы

Функция main является главной функцией программы. 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 (константы описаны в Таблице 2). Потом с помощью оператора *switch* анализируется возвращенная константа и выполняются соответствующие действия.

Основной алгоритм программы представлен на **рисунке 2**.

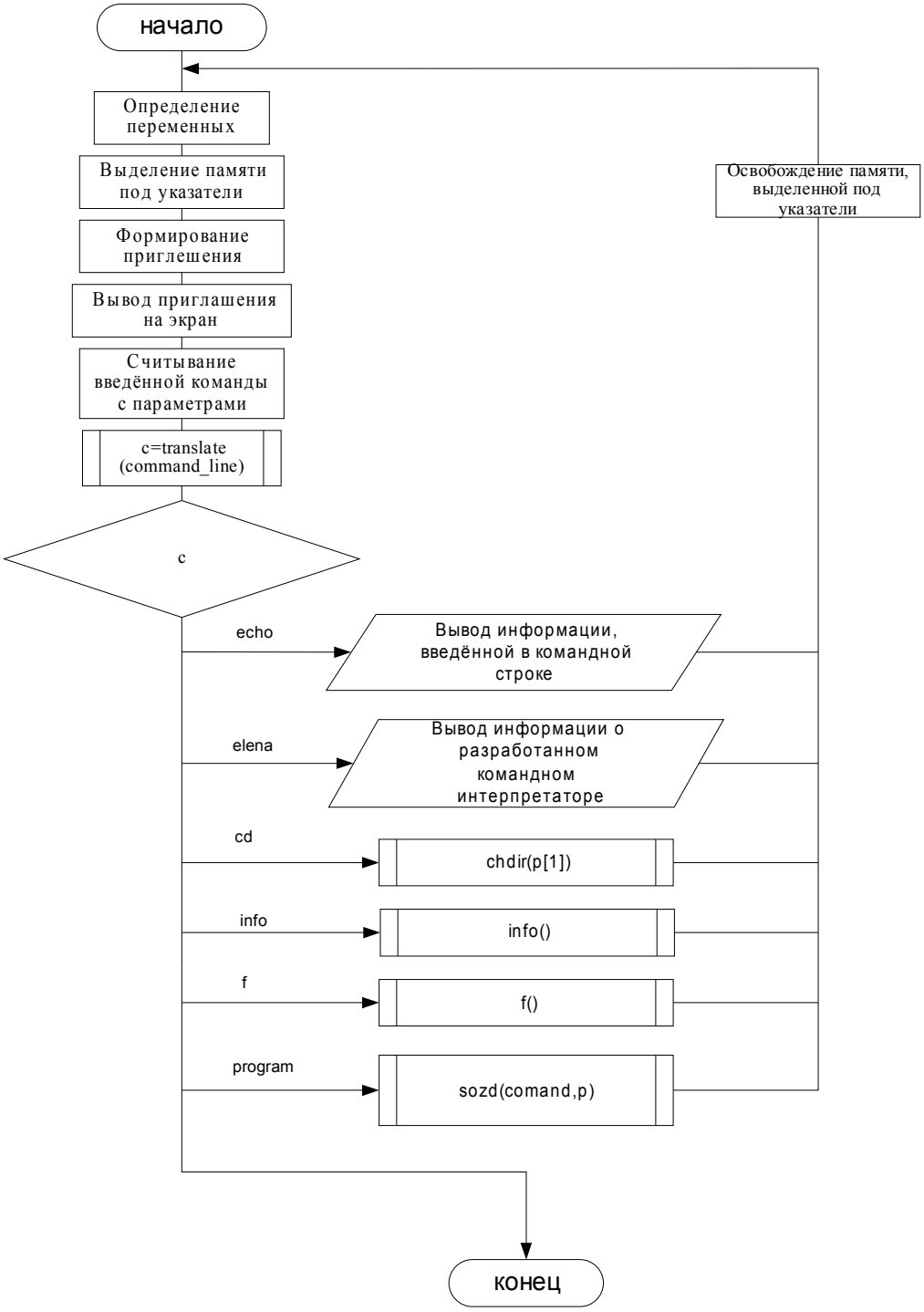

Рисунок 2 – Основной алгоритм программы

4.2 Описание функции translate

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 3.**

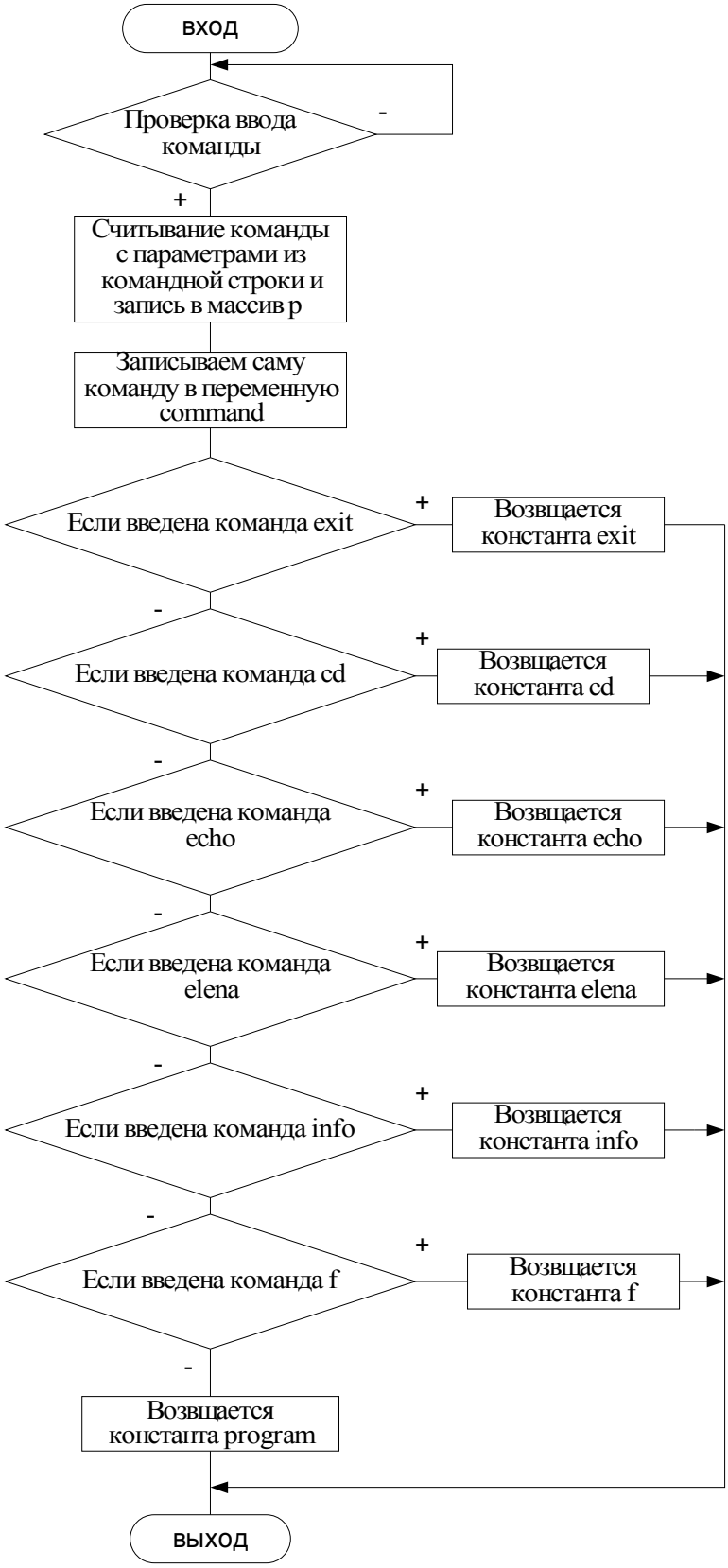

Рисунок 3 – Алгоритм функции translate

4.3 Описание функции sozd

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.**

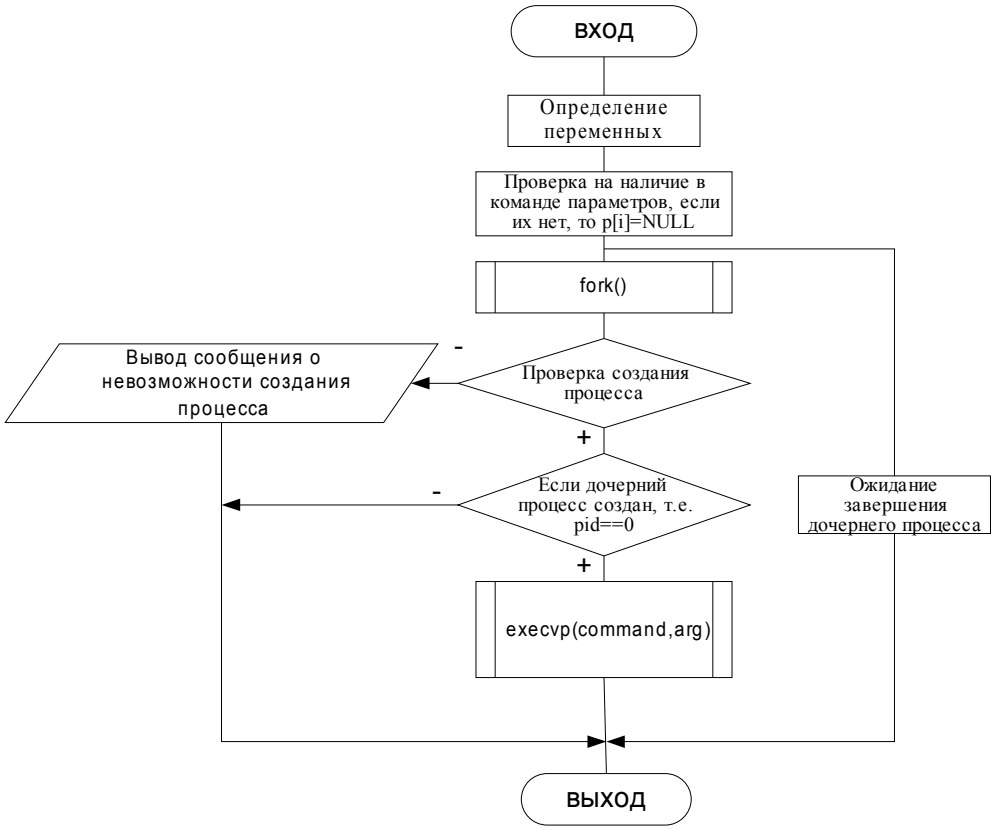

Рисунок 4 – Алгоритм функции sozd

4.4 Описание функции info

Info– функция, выводящая на экран информацию о системе: имени ОС, релиз-типе ядра, типе машины процессора, модели процессора, частоте процессора, размере КЭШа. Информация берётся из файла /proc/cpuinfo.

Алгоритм функции info представлен на **рисунке 5.**

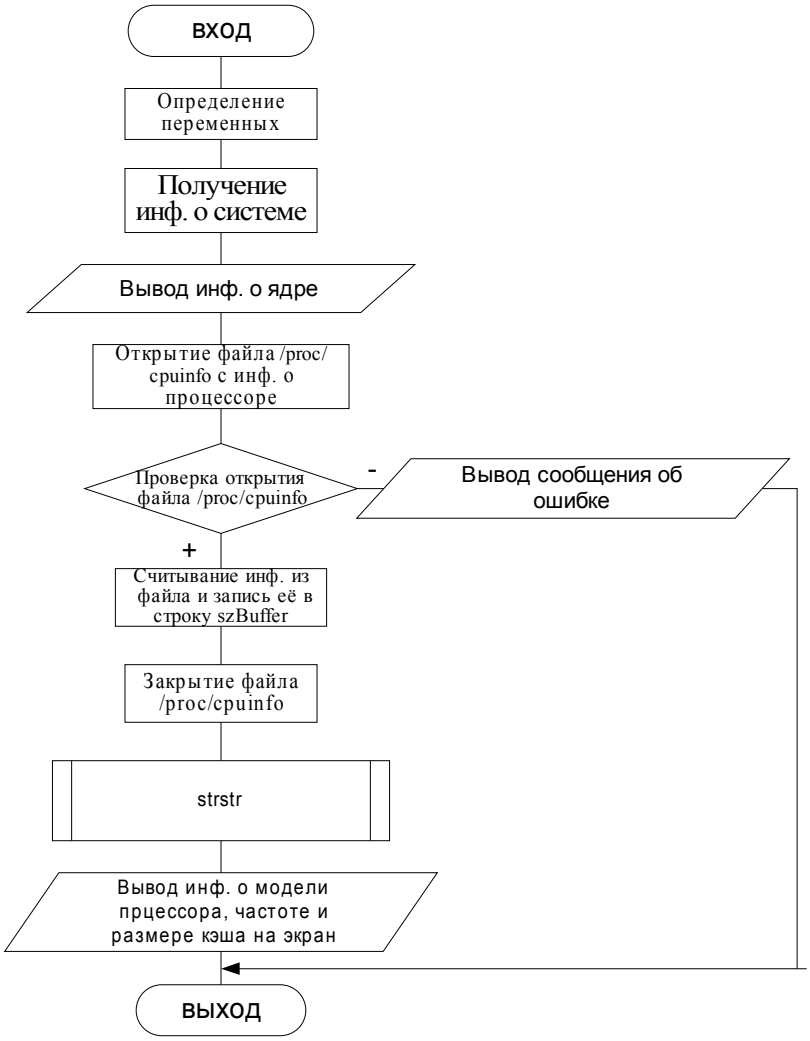

Рисунок 5 – Алгоритм функции info

4.5 Описание функции f

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 создатся с именем 1.txt в текущей директории.

Алгоритм функции F представлен на **рисунке 6.**

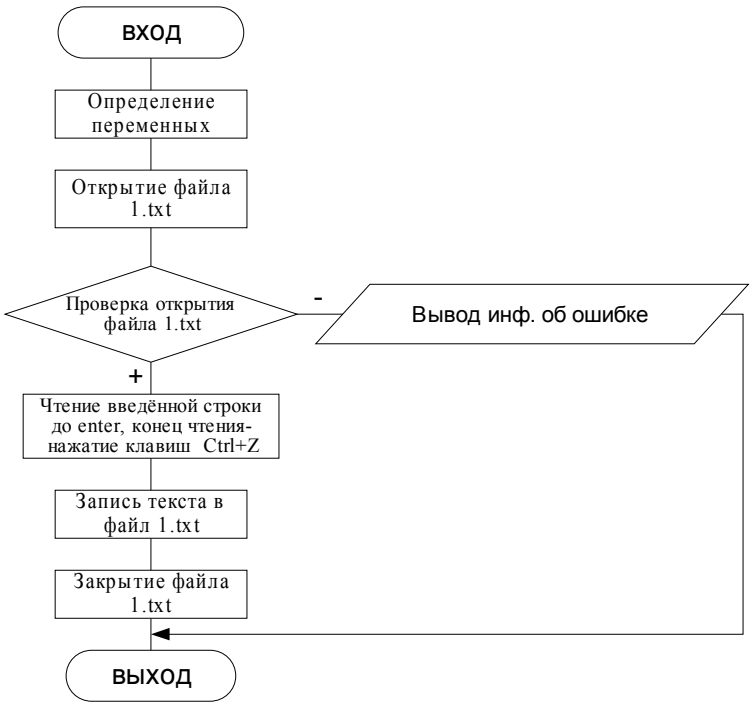

Рисунок 6 – Алгоритм функции F

5 РАЗРАБОТКА ПРОГРАММЫ

В программе используются глобальные переменные, представленные в **таблице 2**.

Таблица 2 - Глобальные переменные

Наименование	Тип	Назначение переменной
command_line[50]	char	Массив для ввода команды и аргументов
command[20]	char	Массив для команды
p[10]	char*	Массив указателей на аргументы команды
comand_size	int	Длина команды
buffer[100]	char	Буфер для строки с приглашением
cwd[50]	char	Буфер для названия текущей директории
c	int	Идентификатор введённой команды

В программе используются константы, обозначающие внутренние команды интерпретатора, представленные в **таблице 3**.

Таблица 3 - Константы

Наименование	Назначение константы
ELENA	Команда вывода информации о проекте
INFO	Команда вывода информации о системе
F	Команда записи текста в файл
ECHO	Команда вывода на экран
CD	Команда смены директории
PROGRAM	Запуск внешней программы
NO_COMMAND	Отсутствие команды
EXIT	Команда выхода

Локальные переменные главной функции main, используемые в программе, представлены в **таблице 4.**

Таблица 4 – Локальные переменные функции main

Наименование	Тип	Назначение переменной
i	int	Счётчик цикла

Локальные переменные в функции translate не используются.

Переменные, используемые в функции sozd представлены в **таблице 5.**

Таблица 5 – Локальные переменные функции sozd

Наименование	Тип	Назначение переменной
i	int	Счётчик цикла
pid	int	Идентификатор дочернего процесса
status	int	Статус завершения процесса

Переменные, используемые в функции info представлены в **таблице 6**.

Таблица 6– Локальные переменные функции info

Наименование	Тип	Назначение переменной
szBuffer[1024]	char	Буфер для файла для считывания инф.
fd	int	Дескриптор файла
i	int	Кол-во реально прочитанных байт
p	char*	Указатель на строку
u	utsname	Структура, хранящая инф. о системе
cache_size	int	Размер кэша
cpu_speed	float	Частота процессора
model_name[50]	char	Модель процессора

Переменные, используемые в функции f представлены в **таблице 7.**

Таблица 7 – Локальные переменные функции f

Наименование	Тип	Назначение переменной
szBuffer[1024]	char	Счётчик цикла
fd	int	Идентификатор процесса
i	int	Кол-во реально прочитанных байт

Программа, реализована на языке С в среде программирования ОС Unix. Текст программы приведен в приложении Б.

ЗАКЛЮЧЕНИЕ

В процессе решения поставленной задачи разработана программа, выполняющая работу командного интерпретатора.

В пояснительной записке проведён анализ программ подобного типа, разработаны структура программы, алгоритм решения задачи и алгоритмы программы, приведены требования для работы программы и действия, необходимые для её запуска и функционирования. Текст программы написан на языке С.

СПИСОК ЛИТЕРАТУРЫ

1) Пособие по оформлению курсовых и дипломных работ для студентов специальности 2001 дневного обучения/под ред. Соломенцева В.В.. - М: МГТУ ГА, 2002.
2) Робачевский А. М. Операционная система Unix. - СПб: БХВ-Петербург, 2001.
3) Стахнов А. А. Linux. – СПб.: БХВ-Петербург, 2003.

Приложение А

Спецификация

Обозначения	Наименование	Примечание
	Документация	
КР 021003 12	Разработка	
	командного	
	интерпретатора,	
	распознающего	
	внутренние и внешние	
	команды.	
	Текст программы.	
КР 021003 34	Разработка	
	командного	
	интерпретатора,	
	распознающего	
	внутренние и внешние	
	команды.	
	Руководство оператора.	

Приложение Б

УТВЕРЖДЕН
КР 021003 12

РАЗРАБОТКА КОМАНДНОГО ИНТЕРПРЕТАТОРА, РАСПОЗНАЮЩЕГО ВНУТРЕННИЕ И ВНЕШНИЕ КОМАНДЫ

ТЕКСТ ПРОГРАММЫ

КР 021003 12

Листов 4

2004

```
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>
#include <sys/utsname.h> //для вывода системной информации

#define EXIT            0     //команда выхода
#define CD              1     //команда смены директории
#define ECHO            2     //вывод на экран
#define PROGRAM         1000  //внешняя программа
#define ELENA           1985  //о программе
#define INFO            1024  //информация о машине и системе
#define F               2004  //вывод в файл с консоли (команда cat>1.txt)
#define NO_COMMAND      2005  //нет команды

char command_line[50];       //массив для введенной команды с аргументами
char command[20];            //сама команда
char* p[10];     //массив указателей на аргументы команды (p[0]-сама команда)
int command_size=0;
char buffer[100];            //буфер под строку с приглашением
char cwd[50];                //буфер под название текущей директории
int c=0;                     //идентификатор введенной команды

//функция аналогична cat>1.txt
//весь ввод с консоли направляется в файл
//для вызова - команда "f"
void f()
{
    char szBuffer[1024];
    int fd,i;

    fd = open("1.txt",O_WRONLY | O_CREAT,S_IWUSR|S_IRUSR);
    if (fd==-1)
      {
          printf("Не могу открыть файл\n");
          return;
      }

    while(1)//бесконечное чтение
      {
          i = read(0,szBuffer,1023); //чтение до Enter
          szBuffer[i]='\0';
          write(fd,szBuffer,i);     //запись в файл
      }//из цикла выходим только если ввели Ctrl+Z

    close(fd);//закрываем файл

}

 //функция - вывод информации о машине/системе
void info()
{
    int fd,i;           //дескриптор файла, кол-во реально прочитанных байт
    char szBuffer[1024];//буфер под файл
    char *p;
    struct utsname u;  //структура, куда заносятся данные о машине/системе

    int cache_size;     //сюда заносим уже выбранные данные о размете КЭШа,
    float cpu_speed;    //частоте процессора,
    char model_name[50];//модели процессора.
```

```
    uname(&u);//получаем информацию о системе/машине

    printf("\n%s. Kernel - %s on %s\n\n",u.sysname,u.release,u.machine);

    //открываем файл с инф. о процессоре
    fd = open("/proc/cpuinfo",O_RDONLY,S_IRUSR);
    if (fd == -1)
      {
       printf("Не могу вывести информацию\n");
       return;
      }

    i = read(fd,szBuffer,1023);
    szBuffer[i] = '\0';
    close(fd);

    p = strstr(szBuffer,"model name");

    if (p == NULL) return;

    sscanf(p,"model name : %s",model_name);
    printf("\tМодель процессора - %s\n",model_name);

    p = strstr(szBuffer,"cpu MHz");

    if (p == NULL) return;

    sscanf(p,"cpu MHz : %f",&cpu_speed);
    printf("\tЧастота процессора - %f МГц\n",cpu_speed);

    p = strstr(szBuffer,"cache size");

    if (p == NULL) return;

    sscanf(p,"cache size : %i",&cache_size);
    printf("\tРазмер кэша - %i Кб\n",cache_size);

    printf("\n");


}

//функция, выполняющая разбор командной строки

int translate(char* command_line)

{
    if (command_line[0]=='\0') //если кто-то балуется Enter'ом
       return NO_COMMAND;
    sscanf(command_line,"%s%*[ ]%s%*[ ]%s%*[ ]%s%*[ ]%s%*[ ]%s%*[ ]%s%*[ ]%s
%*[ ]%s%*[ ]%s%*[ ]",p[0],p[1],p[2],p[3],p[4],p[5],p[6],p[7],p[8],p[9]);
    strcpy(command,p[0]);

    //анализ команды
    if (!strcmp(command, "exit"))
          return EXIT;

    else if (!strcmp(command, "cd"))
          return CD;

    else if (!strcmp(command, "echo"))
          return ECHO;
```

```
    else if (!strcmp(command, "elena"))
          return ELENA;

    else if (!strcmp(command,"info"))
          return INFO;

    else if (!strcmp(command,"f"))
          return F;

    else // команда внешняя
          return PROGRAM;
}

//функция, выполняющая внешнюю команду
void sozd (char *command,char* arg[])
{
    int i;
    int pid;
    int status=0;

    for (i=0;i<10;i++)
      if (strlen(p[i])==0)
          {
           p[i]=NULL;
           break;
          }

    pid=fork();
    if (pid < 0)
      write(2,"inter: не могу создать процесс",31);

    else if (pid==0)
        {
          i=execvp(command,arg);
          //функция execvp возвращает значение только в случае ошибки
          //следовательно нижеследующий код выполнится
          //ТОЛЬКО при наличии ошибки и завершит дочерний процесс
          //в случае успешного запуска дочернего процесса
          //нижеследующий код будет замещен кодом дочернего процесса
          //и выполнятся не будет
          write(1,"inter: ",7);
          write(1,p[0],strlen(p[0]));
          write(1," Команда не найдена\n",20);
          abort();   //завершение дочернего процесса
      }
    else
       wait(&status);

}

void main()
{
  int i=0;

  while (1)
    {
        for (i=0;i<10;i++)
       p[i]=malloc(20);

        for (i=0;i<10;i++)
         memset(p[i],'\0',20);

      getcwd(cwd,49);//получаем текущую директорию
```

```
  sprintf(buffer,"[%s] -<-<-@ ",cwd);//формируем приглашение
 write(1,buffer,strlen(buffer)); // печатаем приглашение

 command_size=read(0,command_line,50);     // получаем строку
 command_line[command_size-1]='\0';
 c=translate(command_line);

 switch(c)
     {
       case (EXIT):
                        exit(0);

       case (CD):
                        i=chdir(p[1]);
                        if (i && errno==ENOENT)
                          {
                           write(1,"inter: cd: ",10);
                           write(1,p[1],strlen(p[1]));
                           write(1,": Нет такого каталога\n",23);
                          }
                        break;
       case (ECHO):
                        for (i=1;i<10;i++)
                          {
                              write(1,p[i],strlen(p[i]));
                              write(1," ",1);
                          }
                        write(1,"\n",1);
                        break;

       case (PROGRAM):
                        sozd(command,p);
                        break;
       case (ELENA):
                        write(1,"inter - command interpretator \n",31);
                        write(1,"it can makes all you want\n",26);
                        write(1,"made by Al' in 2004\n",20);
                        break;
       case (INFO):

                        info();
                        break;
       case (F):
                        f();
                        break;
       case (NO_COMMAND):
                        ;
     }
 for (i=0;i<10;i++)
     free(p[i]);
}

}
```

Приложение В

УТВЕРЖДЕН
КР 021003 34

РАЗРАБОТКА КОМАНДНОГО ИНТЕРПРЕТАТОРА, РАСПОЗНАЮЩЕГО ВНУТРЕННИЕ И ВНЕШНИЕ КОМАНДЫ

РУКОВОДСТВО ОПЕРАТОРА

КР 021003 34

Листов 3

2004

Аннотация

Руководство оператора содержит информацию о назначении данной программы, условия, необходимые для выполнения программы, подробное описание по выполнению программы, а также информацию об ошибках, которые могут возникнуть в ходе выполнения программы.

Содержание

1 НАЗНАЧЕНИЕ ПРОГРАММЫ

Данная программа выполняет работу командного интерпретатора, распознающего внутренние и внешние команды. При вводе пользователем команды, она выполняется или выводится сообщение об ошибке.

2 УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Условия, необходимые для выполнения программы, предполагают наличие следующего минимального состава аппаратурных и программных средств:

- Процессор x386 и выше
- Наличие ОС UNIX

3 ВЫПОЛНЕНИЕ ПРОГРАММЫ

3.1 Запуск программы

Для того чтобы запустить программу, необходимо следующее:
-включить компьютер и загрузить ОС Linux;
-отправить на выполнение файл *inter (./inter)

3.2 Просмотр результатов

Виды экрана представлены на рисунке 7.

```
[/mnt/win_e]-<-<-@ echo Privet
Privet
[/mnt/win_e]-<-<-@ info
Linux.Kernel – 2.6.3 – 7mdk on i686
  Модель процессора – Intel (R)
  Частота процессора -  2656.593018 МГц
  Размер КЭШа – 512 Кб
[/mnt/win_e]-<-<-@ Elena
inter – command interpretator
it can make all you want
made by Al’ in 2004
[/mnt/win_e]-<-<-@ cd /mnt/win_d
[/mnt/win_d]-<-<-@ ls
Animation   Recycled
music          System Volume
[/mnt/win_d]-<-<-@ f
Hello
[/mnt/win_e]-<-<-@ exit
```

Рисунок 7 – Вид экрана при работе интерпретатора

3.3 Сообщения об ошибках

Выводятся сообщения об ошибке при невозможности открытия файла 1.txt, вывода информации о системе из файла /proc/cpuinfo, создания дочернего процесса, нахождения команды или каталога.

4 СООБЩЕНИЯ ОПЕРАТОРУ

После запуска программы выводится приглашение командного интерпретатора ([/mnt/win_e]-<-<-@) для начала работы пользователя, который может вводить команды и, нажав клавишу «Enter», запускать их на выполнение. После выполнения одной команды опять выводится приглашение. Выход из программы осуществляется при вводе команды“exit”.