Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Лабораторна робота №4

з навчальної дисципліни

«Операційні системи»

Виконав:
студент групи КУ-31
Нечипоренко Д.І.

Харків – 2022

Завдання

Задание №1

Написать функцию, предназначенную для решения квадратного уравнения $a \cdot x^2 + b \cdot x + c = 0$ для произвольных значений коэффициентов a, b и c. Значения коэффициентов передать в функцию main с помощью параметров командной строки.

Задание №2

Написать программу, которой могут быть переданы три «*короткие опции*»: -h (справка по использованию программы и завершение работы), -o out_file_name (задание нестандартного имени выходного файла), -c (особый режим работы). Кроме того, программе могут быть переданы дополнительные аргументы, задающие имена входных файлов. Вывести сообщение, указывающее режим работы программы.

Задание №3

Написать программу, которой могут быть переданы указанные в предыдущем задании «*короткие опции*» и соответствующие им «*длинные опции*»: (--help, --output *out_file_name* и —compile).

Задание №4

Написать программу, которая выводит полный список переменных окружения.

Замечание: для того, чтобы в программе получить доступ ко всему списку переменных окружения вместе с их значениями можно воспользоваться переменной environ, которая в программе объявляется, как:

```
#include <stdlib.h> /* или unistd.h */

extern char **environ;
```

Данная переменная указывает на массив строк, называемый *environment* (окружение). После последней переменной окружения в данном массиве следует NULL.

Задание №5

Написать программу, которая выводит в стандартный поток вывода значение некоторой конкретной переменной окружения, указанной при вызове программы. Если имя переменной окружения не указано при вызове программы, то вывести информацию об использовании программы.

Задание №6

Написать программу, которая присваивает заданной переменной окружения указанное значение, затем проверяет только что установленную переменную и выводит полученную пару *переменная — значение* в стандартный поток вывода. Если при запуске программы необходимая информация не указана, то вывести рекомендацию по использованию программы.

Задание №7

Напишите программу, которая удаляет указанную переменную из окружения. Если имя переменной, которую нужно удалить, не указано при запуске программы, то удалить все окружение. После этого вывести текущее окружение в стандартный поток вывода.

Задание №8

Написать программу, получающую информацию о пользователе, который ее запустил.

Задание №9

Написать программу, получающую информацию о компьютере, на котором выполняется.

Код:

Task 1

Main

```
#include <stdio.h>
#include <stdlib.h>
#include "quadraticEquation.h"

int main(int argc, char *argv[]) {

    if (argc != 4) {
        printf("Error!!! Wrong number of arguments (expected 3 after call program)\n");
        return -1;
    } else {
        double a = atof(argv[1]);
        double b = atof(argv[2]);
        double c = atof(argv[3]);

        QuadraticEquationResult *quadraticEquationResult = solveQuadraticEquation(a, b, c);

        switch(quadraticEquationResult->type) {
            case -1:
                printf("This is NOT QUADRATIC EQUATION\n");
                break;
            case 0:
                printf("Discriminant less than zero\n");
                break;
            case 1:
```

```
            printf("Quadratic equation\nx1 = %g, x2 = %g\n", quadraticEquationResult->x1,
quadraticEquationResult->x2);
            break;
        default:
            printf("Wrong Solution Selector\n");
      }
   }

   return 0;
}
```

Task 2

Main

```
#include <stdio.h>
#include <getopt.h>

int main(int argc, char **argv) {
   int opt;
   char *file_name = NULL;
   int workMode = 0;

   while((opt = getopt(argc, argv, "ho:c")) != -1) {
      switch(opt) {
         case 'h':
            printf("Help string\n");
            break;
         case 'c':
            workMode = 1;
```

```
            break;
        case 'o':
            file_name = optarg;
            break;
        case '?':
            printf("Unknown option ignored\n");
            break;
        default:
            printf("Unknown error!!!\n");
            return -1;
      }
   }

   if (file_name != NULL) {
      printf("Output file: %s\n", file_name);
   }

   if (workMode) {
      printf("Specific work mode\n");
   } else {
      printf("Ordinary work mode\n");
   }

   printf("\n");

   return 0;
}
```

Task 3

Main

```
#include <stdio.h>
#include <getopt.h>

int main(int argc, char **argv) {
    int  opt;
    char *file_name = NULL;
    int workMode = 0;

    const struct option long_opts[] = {
            {"help", no_argument, NULL, 'h'},
            {"output", required_argument, NULL, 'o'},
            {"compile", no_argument, NULL, 'c'},
            {NULL, 0, NULL, 0}
    };

    while((opt = getopt_long(argc, argv, "ho:c", long_opts, NULL)) != -1) {
        switch(opt) {
            case 'h':
                printf("Help string\n");
                break;
            case 'c':
                workMode = 1;
                break;
            case 'o':
                file_name = optarg;
```

```
            break;
        case '?':
            printf("Unknown option ignored\n");
            break;
        default:
            printf("Unknown error!!!\n");
            return -1;
      }
   }

   if (file_name != NULL) {
      printf("Output file: %s\n", file_name);
   }

   if (workMode) {
      printf("Specific work mode\n");
   } else {
      printf("Ordinary work mode\n");
   }

   printf("\n");

   return 0;
}
```

Task 4

main

Task 4

Main

```
#include <stdio.h>

int main() {

    extern char **environ;

    char **env = environ;
    if (env) {
        printf("Environment variable list start\n");
        while(*env) {
            printf("\t%s\n", *env);
            env++;
        }
        printf("Environment variable list end\n");
    } else {
        printf("Environment variable list is empty\n");
    }

    return 0;
}
```

Task 5

Main

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main(int argc, char **argv) {

    if (argc == 2) {
        char *environmentVariableValue = getenv(argv[1]);
        char *value = malloc(strlen(environmentVariableValue)+1);
        strcpy(value, environmentVariableValue);

        if (value) {
            printf("Variable %s possesses the value: %s\n", argv[1], value);
        } else {
            printf("Variable %s doesn't has the value\n", argv[1]);
        }
    } else {
        printf("Error!!! Wrong number of arguments (expected 1 after call program)\n");
    }

    return 0;
}
```

Task 6

setenv-method

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    char *var, *value;

    if (argc == 1 || argc > 3) {
```

```
    fprintf(stderr, "usage: envir1 var [value]\n");
    return 1;
}

var = argv[1];
value = getenv(var);
if (value) {
    printf("Variable %s has value %s\n", var, value);
} else {
    printf("Variable %s doesn't have value\n", var);
}

if (argc == 3) {
    printf("Calling setenv with: variable %s, value %s\n", var, argv[2]);

    if (setenv(var, argv[2], 1) != 0) {
        fprintf(stderr, "setenv failed\n");
        return 1;
    }

    value = getenv(var);
    if (value) {
        printf("New value %s equals %s\n", var, value);
    } else {
        printf("New value %s equals null", var);
    }
}
```

```
    printf("\n");

    return 0;
}
```

putenv-method

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char **argv) {
    char *var, *value;

    if (argc == 1 || argc > 3) {
        fprintf(stderr, "usage: envir1 var [value]\n");
        return 1;
    }

    var = argv[1];
    value = getenv(var);
    if (value) {
        printf("Variable %s has value %s\n", var, value);
    } else {
        printf("Variable %s doesn't have value\n", var);
    }

    if (argc == 3) {
```

```
    char *string;
    value = argv[2];
    string = malloc(strlen(var)+strlen(value)+2);
    if (!string) {
        fprintf(stderr, "out of memory\n");
        return 1;
    }
    strcpy(string, var);
    strcat(string, "=");
    strcat(string, value);
    printf("Calling putenv with: %s\n", string);
    if (putenv(string) != 0) {
        fprintf(stderr, "putenv failed\n");
        free(string);
        return 1;
    }

    value = getenv(var);
    if (value) {
        printf("New value %s equals %s\n", var, value);
    } else {
        printf("New value %s equals null", var);
    }
}

printf("\n");
```

```
    return 0;
}
```

Task 7

Main

```
#include <stdio.h>
#include <stdlib.h>

void showEnvironment();

int main(int argc, char **argv) {

    if (argc > 2) {
        fprintf(stderr, "Too many arguments\n");
        return 1;
    }

    showEnvironment();

    if (argc == 2) {
        printf("Deleting variable %s from environment\n", argv[1]);
        if (unsetenv(argv[1]) != 0) {
            fprintf(stderr, "Cannot unset %s\n", argv[1]);
            return 1;
        }
    } else {
        printf("Deleting all environment\n");
```

```
        clearenv();
    }

    showEnvironment();

    printf("\n");

    return 0;
}

void showEnvironment() {
    extern char **environ;

    char **env = environ;
    if (env) {
        printf("Environment variable list start\n");
        while(*env) {
            printf("\t%s\n", *env);
            env++;
        }
        printf("Environment variable list end\n");
    } else {
        printf("Environment variable list is empty\n");
    }
}
```

Task 8

Main

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <pwd.h>

int main() {
    uid_t uid = getuid();
    gid_t gid = getgid();

    struct passwd *pw;

    printf("User is %s\n", getlogin());

    printf("User IDs: uid = %d, gid = %d\n", uid, gid);

    pw = getpwuid(uid);
    printf("UID passwd entry:\n");
    printf("Name = %s, UserID = %d, GroupID = %d, Home = %s, Shell = %s\n", pw->pw_name, pw->pw_uid, pw->pw_gid, pw->pw_dir, pw->pw_shell);

    pw = getpwnam("root");
    printf("Root passwd entry:\n");
    printf("Name = %s, UserID = %d, GroupID = %d, Home = %s, Shell = %s\n\n", pw->pw_name, pw->pw_uid, pw->pw_gid, pw->pw_dir, pw->pw_shell);

    return 0;
}
```

Task 9

Main

```
#include <sys/utsname.h>
#include <unistd.h>
#include <stdio.h>

int main() {
    char computer[256];
    struct utsname uts;

    if (gethostname(computer, 255) != 0 || uname(&uts) < 0) {
        fprintf(stderr, "Could not get host information\n");
        return 1;
    }

    printf("Computer host name is %s\n", computer);
    printf("System is %s on %s hardware\n", uts.sysname, uts.machine);
    printf("Nodename is %s\n", uts.nodename);
    printf("Version is %s, %s\n",uts.release, uts.version);
    printf("Unique computer id: %ld\n\n", gethostid());

    return 0;
}
```

Результат:

Task 1

```
gcc main.c quadraticEquation.c -o main.out -lm
./main.out 1 4 3
Quadratic equation
x1 = -3, x2 = -1
```

Task 2

```
gcc main.c -o main.out
./main.out -h
Help string
Ordinary work mode

./main.out -c
Specific work mode

./main.out -o main.out
Output file: main.out
Ordinary work mode
```

Task 3

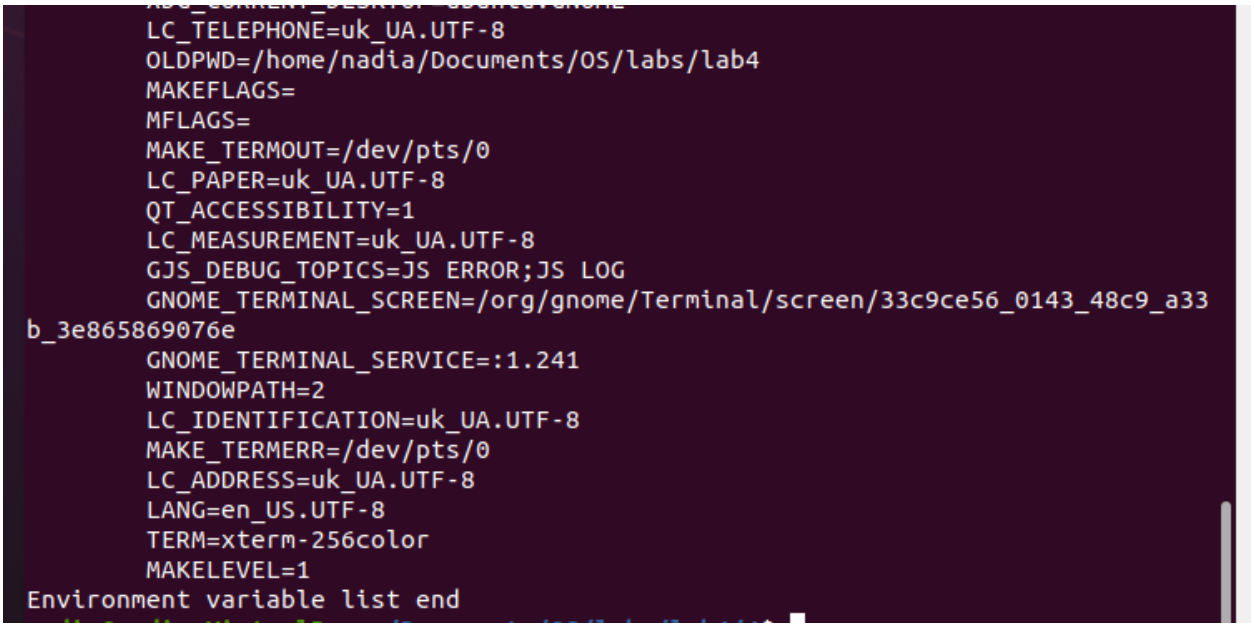
```
gcc main.c -o main.out
./main.out -h
Help string
Ordinary work mode

./main.out -c
Specific work mode

./main.out -o main.out
Output file: main.out
Ordinary work mode

./main.out --help
Help string
Ordinary work mode

./main.out --compile
Specific work mode

./main.out --output main.out
Output file: main.out
Ordinary work mode
```

Task 4

```
gcc main.c -o main.out
./main.out
Environment variable list start
        DESKTOP_SESSION=ubuntu
        XDG_SESSION_CLASS=user
        XDG_SESSION_TYPE=x11
        XAUTHORITY=/run/user/1000/gdm/Xauthority
        GDMSESSION=ubuntu
        XMODIFIERS=@im=ibus
        SHELL=/bin/bash
        VTE_VERSION=6003
        JOURNAL_STREAM=8:34039
        XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
        IM_CONFIG_PHASE=1
        USERNAME=nadia
        XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/
snapd/desktop
        _=/usr/bin/make
        MANAGERPID=1304
        LESSOPEN=| /usr/bin/lesspipe %s
        LC_NAME=uk_UA.UTF-8
        PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
games:/usr/local/games:/snap/bin
        GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
        GJS_DEBUG_OUTPUT=stderr
```

```
        LC_NAME=uk_UA.UTF-8
        PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
games:/usr/local/games:/snap/bin
        GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
        GJS_DEBUG_OUTPUT=stderr
        SESSION_MANAGER=local/nadia-VirtualBox:@/tmp/.ICE-unix/1539,unix/nadia-
VirtualBox:/tmp/.ICE-unix/1539
        XDG_RUNTIME_DIR=/run/user/1000
        XDG_MENU_PREFIX=gnome-
        LC_NUMERIC=uk_UA.UTF-8
        INVOCATION_ID=7d2e98b166a345689faabce907905b74
        LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40
;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;
42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;3
1:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tz
o=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.l
z=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:
*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=0
1;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.
cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;3
1:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*
.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01
;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*
.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=
01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*
.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01
;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv
=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*
.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=
```

```
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DISPLAY=:0
HOME=/home/nadia
PWD=/home/nadia/Documents/OS/labs/lab4/4
SSH_AGENT_PID=1491
GTK_MODULES=gail:atk-bridge
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
LESSCLOSE=/usr/bin/lesspipe %s %s
LOGNAME=nadia
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LC_TIME=uk_UA.UTF-8
COLORTERM=truecolor
QT_IM_MODULE=ibus
SHLVL=1
GNOME_SHELL_SESSION_MODE=ubuntu
USER=nadia
LC_MONETARY=uk_UA.UTF-8
XDG_CURRENT_DESKTOP=ubuntu:GNOME
LC_TELEPHONE=uk_UA.UTF-8
OLDPWD=/home/nadia/Documents/OS/labs/lab4
MAKEFLAGS=
MFLAGS=
MAKE_TERMOUT=/dev/pts/0
LC_PAPER=uk_UA.UTF-8
QT_ACCESSIBILITY=1
LC_MEASUREMENT=uk_UA.UTF-8
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
```



Task 5

```
gcc main.c -o main.out
./main.out USERNAME
Variable USERNAME possesses the value: nadia
```

Task 6

```
gcc putenv-method.c -o putenv-method.out
gcc setenv-method.c -o setenv-method.out
./putenv-method.out USERNAME someone
Variable USERNAME has value nadia
Calling putenv with: USERNAME=someone
New value USERNAME equals someone

./setenv-method.out NEW-ENVIRONMENT-VARIABLE somevalue
Variable NEW-ENVIRONMENT-VARIABLE doesn't have value
Calling setenv with: variable NEW-ENVIRONMENT-VARIABLE, value somevalue
New value NEW-ENVIRONMENT-VARIABLE equals somevalue
```

Task 7

```
gcc main.c -o main.out
./main.out
Environment variable list start
        DESKTOP_SESSION=ubuntu
        XDG_SESSION_CLASS=user
        XDG_SESSION_TYPE=x11
        XAUTHORITY=/run/user/1000/gdm/Xauthority
        GDMSESSION=ubuntu
        XMODIFIERS=@im=ibus
        SHELL=/bin/bash
        VTE_VERSION=6003
        JOURNAL_STREAM=8:34039
        XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
        IM_CONFIG_PHASE=1
        USERNAME=nadia
        XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/
snapd/desktop
        _=/usr/bin/make
        MANAGERPID=1304
        LESSOPEN=| /usr/bin/lesspipe %s
        LC_NAME=uk_UA.UTF-8
        PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/
games:/usr/local/games:/snap/bin
        GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
        GJS_DEBUG_OUTPUT=stderr
        SESSION_MANAGER=local/nadia-VirtualBox:@/tmp/.ICE-unix/1539,unix/nadia-
VirtualBox:/tmp/.ICE-unix/1539
        XDG_RUNTIME_DIR=/run/user/1000
```

.

.

.

```
b_3e865869076e
        GNOME_TERMINAL_SERVICE=:1.241
        WINDOWPATH=2
        LC_IDENTIFICATION=uk_UA.UTF-8
        MAKE_TERMERR=/dev/pts/0
        LC_ADDRESS=uk_UA.UTF-8
        LANG=en_US.UTF-8
        TERM=xterm-256color
        MAKELEVEL=1
Environment variable list end
Deleting all environment
Environment variable list is empty
```

Task 8

```
gcc main.c -o main.out
./main.out
User is nadia
User IDs: uid = 1000, gid = 1000
UID passwd entry:
Name = nadia, UserID = 1000, GroupID = 1000, Home = /home/nadia, Shell = /bin/b
ash
Root passwd entry:
Name = root, UserID = 0, GroupID = 0, Home = /root, Shell = /bin/bash
```

Task 9

```
gcc main.c -o main.out
./main.out
Computer host name is nadia-VirtualBox
System is Linux on x86_64 hardware
Nodename is nadia-VirtualBox
Version is 5.11.0-27-generic, #29~20.04.1-Ubuntu SMP Wed Aug 11 15:58:17 UTC 20
21
Unique computer id: 1376597032
```