

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

**Факультет физико-математических и естественных
наук**

**Фундаментальная Информатика и Информационные
технологии**

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

дисциплина: операционные системы

Этук Нсе-Абаси Акпан

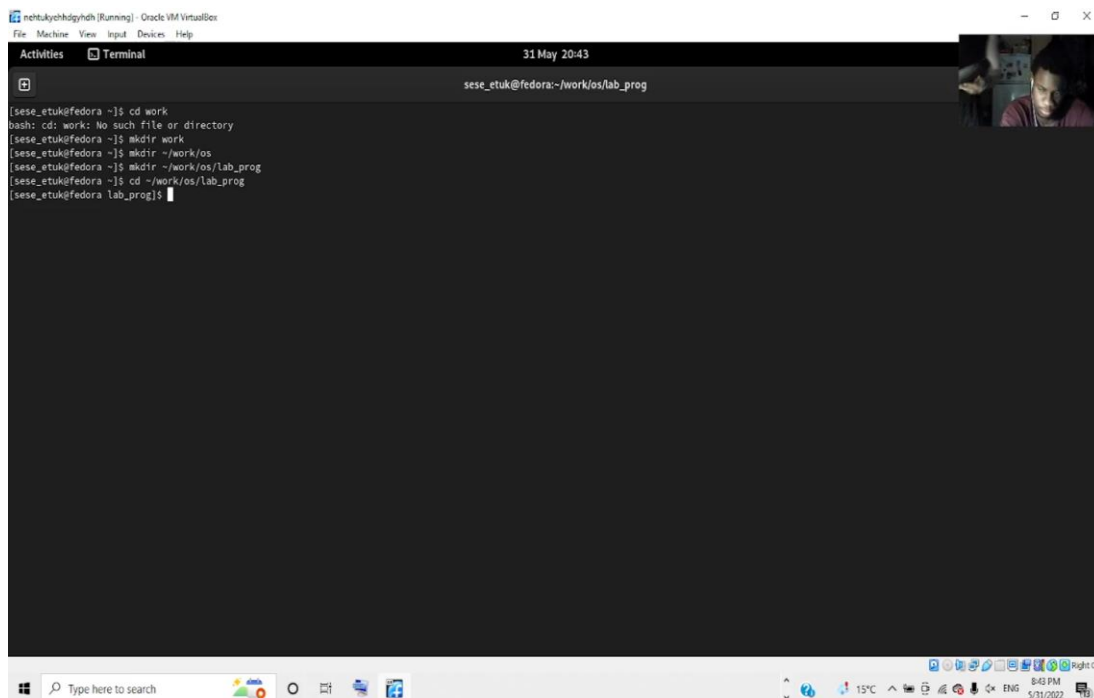
НФИбд-02-21

Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования. С калькулятора с простейшими функциями.

Ход работы

1. В домашнем каталоге создайте подкаталог ~/work/os/lab_prog.



The screenshot shows a terminal window titled "sese_etuk@fedora: ~/work/os/lab_prog" running inside an Oracle VM VirtualBox. The terminal output shows the following commands and their results:

```
[sese_etuk@fedora ~]$ cd work
bash: cd: work: No such file or directory
[sese_etuk@fedora ~]$ mkdir work
[sese_etuk@fedora ~]$ mkdir ~/work/os
[sese_etuk@fedora ~]$ mkdir ~/work/os/lab_prog
[sese_etuk@fedora ~]$ cd ~/work/os/lab_prog
[sese_etuk@fedora lab_prog]$
```

The terminal window is part of a desktop environment with a taskbar at the bottom showing various application icons and system status information (15°C, 8:43 PM, 5/31/2022).

2. Создайте в нём файлы: calculate.h, calculate.c, main.c. Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять sin, cos, tan. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится. Реализация функций калькулятора в файле calculate.h:

nehtukyohdghdh [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal 31 May 20:53

sese_etuk@fedora:~/work/os/lab_prog — mcedit calculate.h

```
calculate.h [-----] 8 L: 1+ 8 9/ 9) +(151 / 174b) 8835 8x823
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

float Calculate(float Numeral, char Operation[4]);

#endif // CALCULATE_H_
```

help save mark replace copy move search delete pull down quit

Type here to search 15°C 8:53 PM 5/31/2022

nehtukyohdghdh [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Activities Terminal 31 May 21:01

sese_etuk@fedora:~/work/os/lab_prog — mcedit main.c

```
main.c [-----] 4 L: 1+16 17/ 19) +(167 / 483b) 8112 8x870
// main.c

#include <stdio.h>
#include <math.h>
int
main(void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Input: ");
    scanf("%f", &Numeral);
    printf("Operation (+, -, *, /, pow, sqrt, sin, cos, tan): ");
    scanf("%s", Operation);
    Result = Calculate(Numeral, Operation);
    printf("Result: %f\n", Result);
    return 0;
}
```

help save mark replace copy move search delete pull down quit

Type here to search 15°C 9:01 PM 5/31/2022

nehtukyhghghh [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

31 May 21:21

sese_etuk@fedora:~/work/os/lab_prog -- mcedit calculate.c

```
calculate.c [x86-] 0 L: 1:34 35/ 62) x(787 /1583b) 8032 8x020
//
// calculate.c
//
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <limits.h>

float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if (strncmp(Operation, "+", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        return Numeral + SecondNumeral;
    }
    else if (strncmp(Operation, "-", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        return Numeral - SecondNumeral;
    }
    else if (strncmp(Operation, "*", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        return Numeral * SecondNumeral;
    }
    else if (strncmp(Operation, "/", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        if (SecondNumeral == 0)
        {
            printf("Error: деление на ноль!");
        }
    }
}
```

1help 2save 3mark 4replac 5copy 6move 7search 8delete 9fullm 10quit

Type here to search

15°C 9:21 PM 5/31/2022

nehtukyhghghh [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

31 May 21:25

sese_etuk@fedora:~/work/os/lab_prog -- mcedit calculate.c

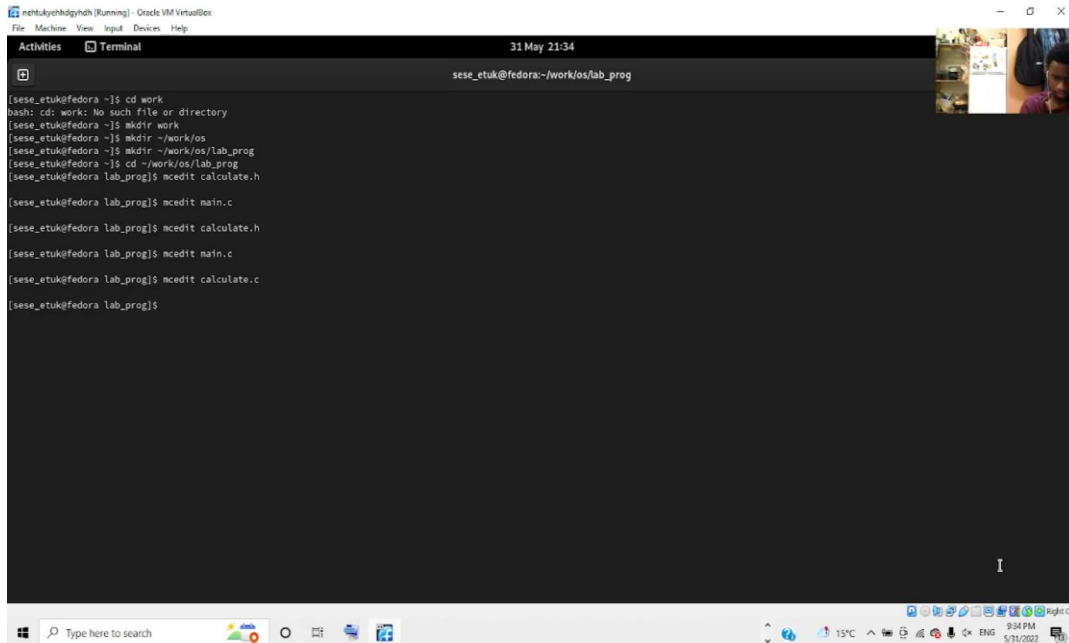
```
calculate.c [x86-] 6 L: 26+ 6 32/ 62) x(725 /1521b) 8123 8x078
//
// calculate.c
//
#include <stdio.h>
#include <math.h>
#include <string.h>
#include <limits.h>

float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if (strncmp(Operation, "+", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        return Numeral + SecondNumeral;
    }
    else if (strncmp(Operation, "-", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        if (SecondNumeral == 0)
        {
            printf("Error: деление на ноль!");
            return HUGE_VAL;
        }
        return Numeral - SecondNumeral;
    }
    else if (strncmp(Operation, "*", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        return Numeral * SecondNumeral;
    }
    else if (strncmp(Operation, "/", 1) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        if (SecondNumeral == 0)
        {
            printf("Error: деление на ноль!");
            return HUGE_VAL;
        }
        return Numeral / SecondNumeral;
    }
    else if (strncmp(Operation, "pow", 3) == 0)
    {
        printf("Please choose: ");
        scanf("%f", &SecondNumeral);
        return pow(Numeral, SecondNumeral);
    }
    else if (strncmp(Operation, "sqrt", 4) == 0)
    {
        return sqrt(Numeral);
    }
    else if (strncmp(Operation, "sin", 3) == 0)
    {
        return sin(Numeral);
    }
    else if (strncmp(Operation, "cos", 3) == 0)
    {
        return cos(Numeral);
    }
    else if (strncmp(Operation, "tan", 3) == 0)
    {
        return tan(Numeral);
    }
    else
    {
        printf("Неправильно введено действие!");
        return HUGE_VAL;
    }
}
```

1help 2save 3mark 4replac 5copy 6move 7search 8delete 9fullm 10quit

Type here to search

15°C 9:25 PM 5/31/2022

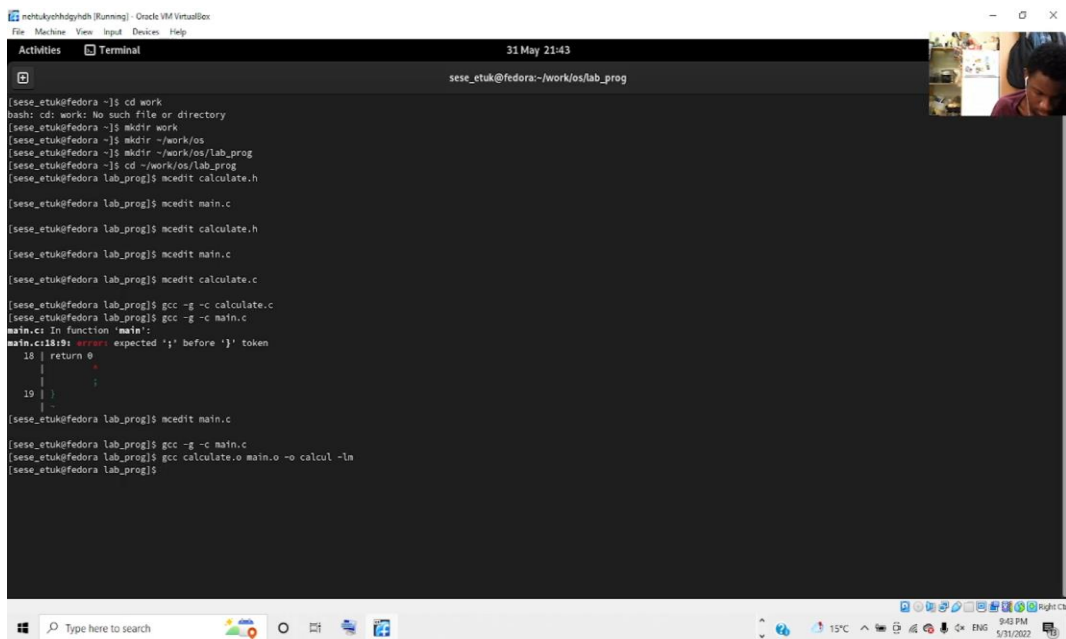


The screenshot shows a terminal window titled "Terminal" with the date "31 May 21:34". The user is logged in as "sese_etuk@fedora" in the directory "~/work/os/lab_prog". The terminal output shows the following commands and their results:

```
[sese_etuk@fedora ~]$ cd work
bash: cd: work: No such file or directory
[sese_etuk@fedora ~]$ mkdir work
[sese_etuk@fedora ~]$ mkdir ~/work/os
[sese_etuk@fedora ~]$ mkdir ~/work/os/lab_prog
[sese_etuk@fedora ~]$ cd ~/work/os/lab_prog
[sese_etuk@fedora lab_prog]$ ncedit calculate.h
[sese_etuk@fedora lab_prog]$ ncedit main.c
[sese_etuk@fedora lab_prog]$ ncedit calculate.h
[sese_etuk@fedora lab_prog]$ ncedit main.c
[sese_etuk@fedora lab_prog]$ ncedit calculate.c
[sese_etuk@fedora lab_prog]$
```

3. Выполните компиляцию программы посредством gcc:

4. При необходимости исправьте синтаксические ошибки.



The screenshot shows a terminal window titled "Terminal" with the date "31 May 21:43". The user is logged in as "sese_etuk@fedora" in the directory "~/work/os/lab_prog". The terminal output shows the following commands and their results:

```
[sese_etuk@fedora ~]$ cd work
bash: cd: work: No such file or directory
[sese_etuk@fedora ~]$ mkdir work
[sese_etuk@fedora ~]$ mkdir ~/work/os
[sese_etuk@fedora ~]$ mkdir ~/work/os/lab_prog
[sese_etuk@fedora ~]$ cd ~/work/os/lab_prog
[sese_etuk@fedora lab_prog]$ ncedit calculate.h
[sese_etuk@fedora lab_prog]$ ncedit main.c
[sese_etuk@fedora lab_prog]$ ncedit calculate.h
[sese_etuk@fedora lab_prog]$ ncedit main.c
[sese_etuk@fedora lab_prog]$ ncedit calculate.c
[sese_etuk@fedora lab_prog]$ gcc -g -c calculate.c
[sese_etuk@fedora lab_prog]$ gcc -g -c main.c
main.c: In function 'main':
main.c:18:19: error: expected ';' before '}' token
   18 |     return 0
      |             ^
   19 | }
      | ~~~~~
[sese_etuk@fedora lab_prog]$ ncedit main.c
[sese_etuk@fedora lab_prog]$ gcc -g -c main.c
[sese_etuk@fedora lab_prog]$ gcc calculate.o main.o -o calcul -ln
[sese_etuk@fedora lab_prog]$
```

5. Создайте Makefile со следующим содержанием:

```
Makefile
#
# Makefile
#
CC = gcc
CFLAGS =
LIBS = -lm

calcul: calcul.o main.o
gcc calcul.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
gcc -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
gcc -c main.c $(CFLAGS)

clean:
rm calcul *.o *.~

End Makefile
```

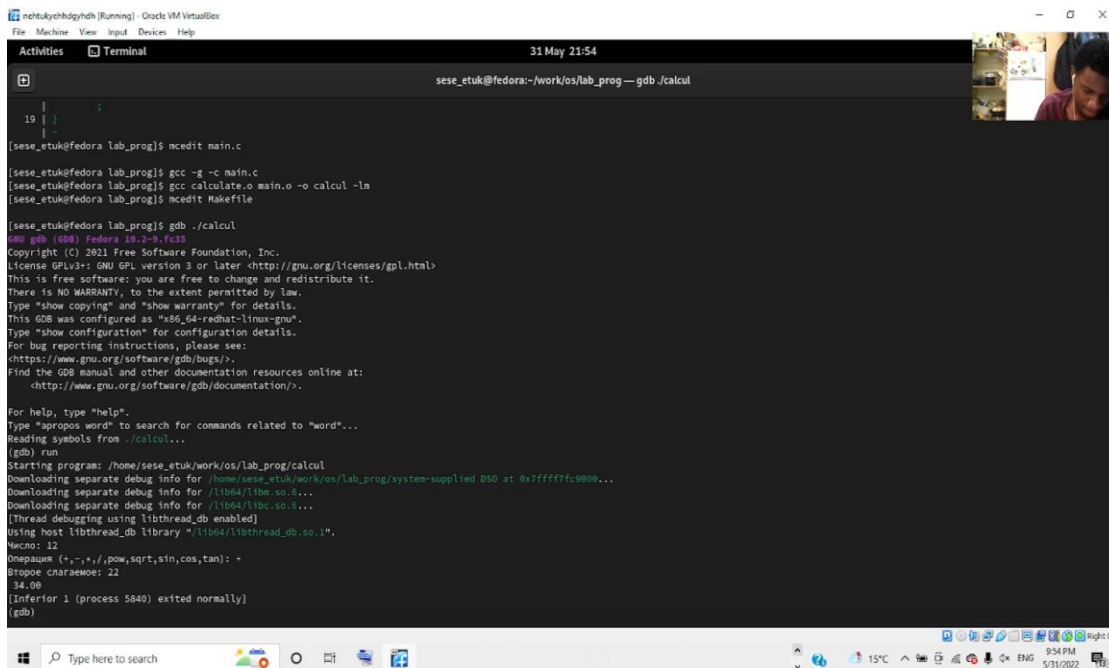
6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

– Запустите отладчик GDB, загрузив в него программу для отладки:

```
[sese_etuk@fedora lab_prog]$ mcedit calculate.h
[sese_etuk@fedora lab_prog]$ mcedit main.c
[sese_etuk@fedora lab_prog]$ mcedit calculate.c
[sese_etuk@fedora lab_prog]$ gcc -g -c calculate.c
[sese_etuk@fedora lab_prog]$ gcc -g -c main.c
main.c: In function 'main':
main.c:18:9: error: expected ';' before '}' token
   18 |     return 0;
      |         ^
   19 | }
      |
   20 |
[sese_etuk@fedora lab_prog]$ mcedit main.c
[sese_etuk@fedora lab_prog]$ gcc -g -c main.c
[sese_etuk@fedora lab_prog]$ gcc calcul.o main.o -o calcul -lm
[sese_etuk@fedora lab_prog]$ mcedit Makefile
[sese_etuk@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 10.2-9.fc33
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb)
```

– Для запуска программы внутри отладчика введите команду run:



```
nehtukyohdghdh (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
31 May 21:54
sese_etuk@fedora:~/work/os/lab_prog - gdb ./calcul

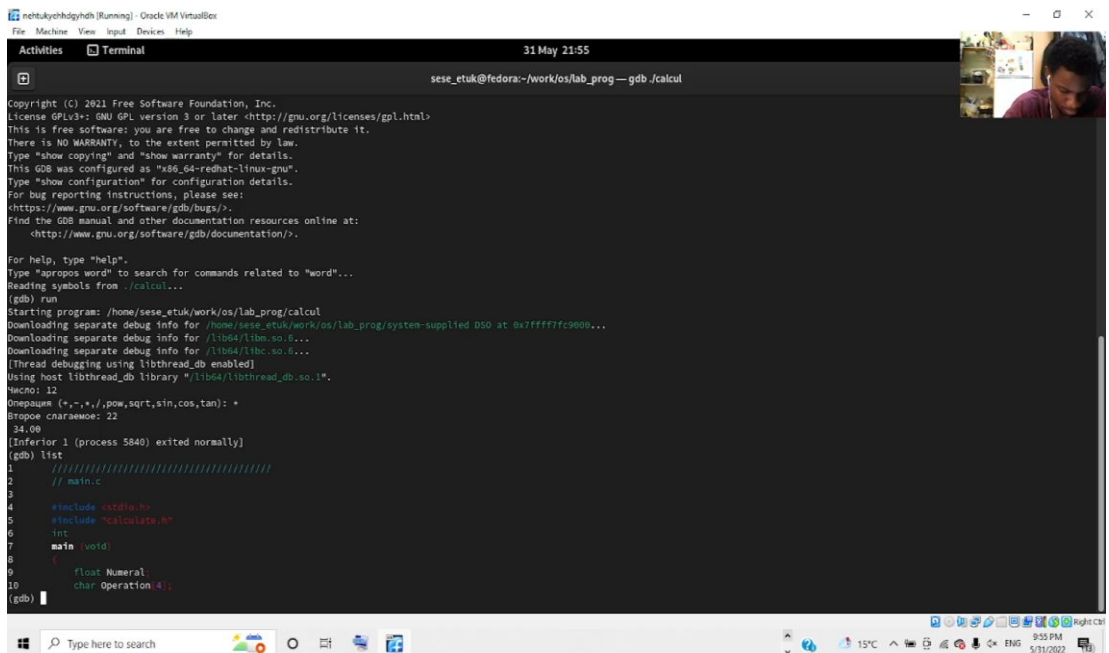
19 | }
|
|
|
[sese_etuk@fedora lab_prog]$ nccedit main.c

[sese_etuk@fedora lab_prog]$ gcc -g -c main.c
[sese_etuk@fedora lab_prog]$ gcc calculate.o main.o -o calcul -ln
[sese_etuk@fedora lab_prog]$ nccedit Makefile

[sese_etuk@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 10.2-9.fc35
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb) run
Starting program: /home/sese_etuk/work/os/lab_prog/calcul
Downloading separate debug info for /home/sese_etuk/work/os/lab_prog/system-supplied 050 at 0x7ffff7fc9000...
Downloading separate debug info for /lib64/libm.so.6...
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
%cnco: 12
Onepaqum (+,-,*,/,pow,sqrt,sin,cos,tan): +
Bropoe cnarawoe: 22
34.00
[Inferior 1 (process 5840) exited normally]
(gdb)
```

– Для постраничного (по 9 строк) просмотра исходного код используйте команду list:

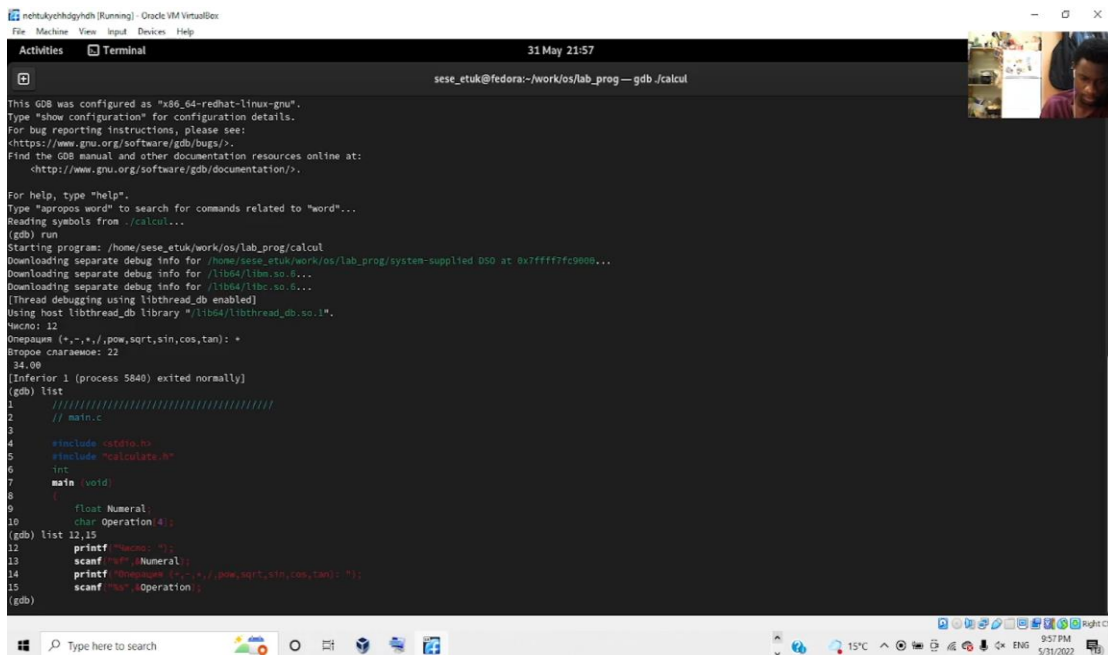


```
nehtukyohdghdh (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
31 May 21:55
sese_etuk@fedora:~/work/os/lab_prog - gdb ./calcul

Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb) run
Starting program: /home/sese_etuk/work/os/lab_prog/calcul
Downloading separate debug info for /home/sese_etuk/work/os/lab_prog/system-supplied 050 at 0x7ffff7fc9000...
Downloading separate debug info for /lib64/libm.so.6...
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
%cnco: 12
Onepaqum (+,-,*,/,pow,sqrt,sin,cos,tan): +
Bropoe cnarawoe: 22
34.00
[Inferior 1 (process 5840) exited normally]
(gdb) list
1 ///////////////////////////////////////////////////
2 // main.c
3
4 #include <stdio.h>
5 #include "calculate.h"
6 int
7 main(void)
8 {
9     float Numeral;
10    char Operation[4];
(gdb)
```

– Для просмотра строк с 12 по 15 основного файла используйте list с параметрами:

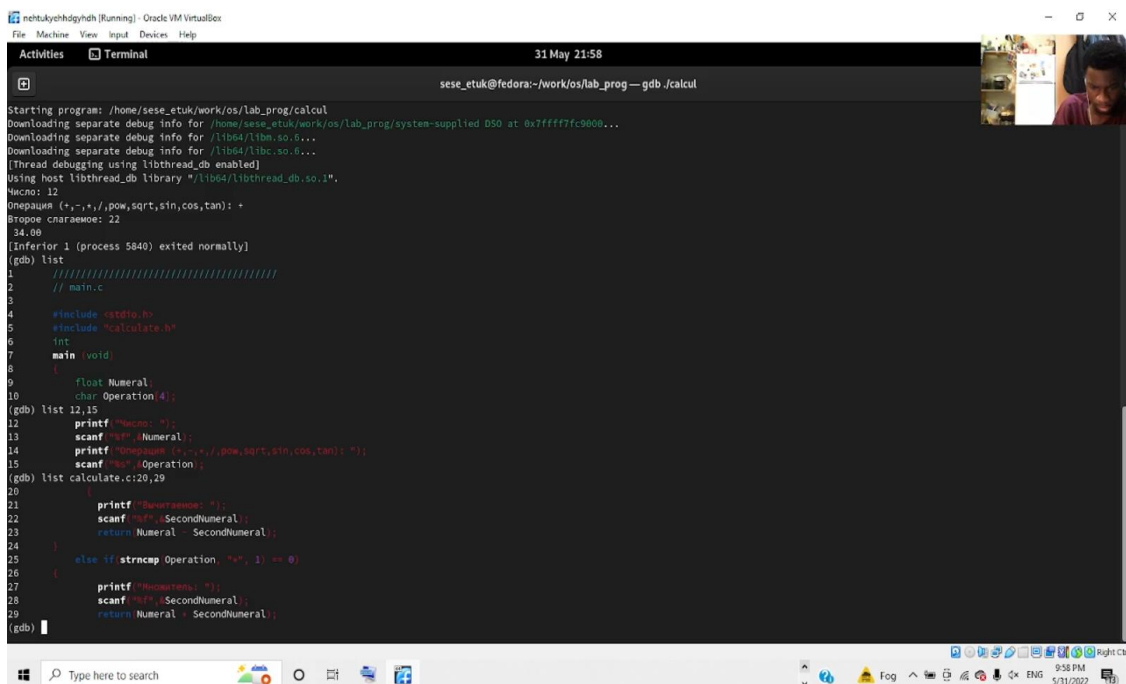


The screenshot shows a GDB terminal window with the following content:

```
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb) run
Starting program: /home/sese_etuk/work/os/lab_prog/calcul
Downloading separate debug info for /home/sese_etuk/work/os/lab_prog/system-supplied DSO at 0x7ffff7c9908...
Downloading separate debug info for /lib64/libm.so.6...
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 12
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе число: 22
34.00
[Inferior 1 (process 5840) exited normally]
(gdb) list
1  // main.c
2
3
4  #include <stdio.h>
5  #include "calculate.h"
6  int
7  main(void)
8  {
9      float Numeral;
10     char Operation[4];
(gdb) list 12,15
12     printf("Число: ");
13     scanf("%f",&Numeral);
14     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15     scanf("%s",&Operation);
(gdb)
```

– Для просмотра определённых строк не основного файла используйте list с параметрами:



The screenshot shows a GDB terminal window with the following content:

```
Starting program: /home/sese_etuk/work/os/lab_prog/calcul
Downloading separate debug info for /home/sese_etuk/work/os/lab_prog/system-supplied DSO at 0x7ffff7c9908...
Downloading separate debug info for /lib64/libm.so.6...
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 12
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе число: 22
34.00
[Inferior 1 (process 5840) exited normally]
(gdb) list
1  // main.c
2
3
4  #include <stdio.h>
5  #include "calculate.h"
6  int
7  main(void)
8  {
9      float Numeral;
10     char Operation[4];
(gdb) list 12,15
12     printf("Число: ");
13     scanf("%f",&Numeral);
14     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15     scanf("%s",&Operation);
(gdb) list calculate.c:20,29
20     {
21         printf("Введите второе число: ");
22         scanf("%f",&SecondNumeral);
23         return Numeral - SecondNumeral;
24     }
25     else if (strcmp(Operation, "<div>") == 0)
26     {
27         printf("Введите второе число: ");
28         scanf("%f",&SecondNumeral);
29         return Numeral / SecondNumeral;
(gdb)
```

– Установите точку останова в файле calculate.c на строке номер 21:

The screenshot shows a GDB terminal window with the following commands and output:

```
(gdb) list
1  //////////////////////////////////////////////////
2  // main.c
3
4  #include <stdio.h>
5  #include "calculate.h"
6  int
7  main(void)
8  {
9      float Numeral;
10     char Operation[4];
(gdb) list 12,15
12     printf("Numeral: ");
13     scanf("%f",&Numeral);
14     printf("Operation (+, -, *, /, pow, sqrt, sin, cos, tan): ");
15     scanf("%s",&Operation);
(gdb) list calculate.c:20,29
20     {
21         printf("Numeral: ");
22         scanf("%f",&SecondNumeral);
23         return Numeral - SecondNumeral;
24     }
25     else if (strcmp(Operation, "-") == 0)
26     {
27         printf("Numeral: ");
28         scanf("%f",&SecondNumeral);
29         return Numeral - SecondNumeral;
(gdb) list calculate.c:20,27
20     {
21         printf("Numeral: ");
22         scanf("%f",&SecondNumeral);
23         return Numeral - SecondNumeral;
24     }
25     else if (strcmp(Operation, "-") == 0)
26     {
27         printf("Numeral: ");
(gdb) break 21
Breakpoint 1 at 0x401207: file calculate.c, line 21.
(gdb)
```

– Выведите информацию об имеющихся в проекте точка останова:

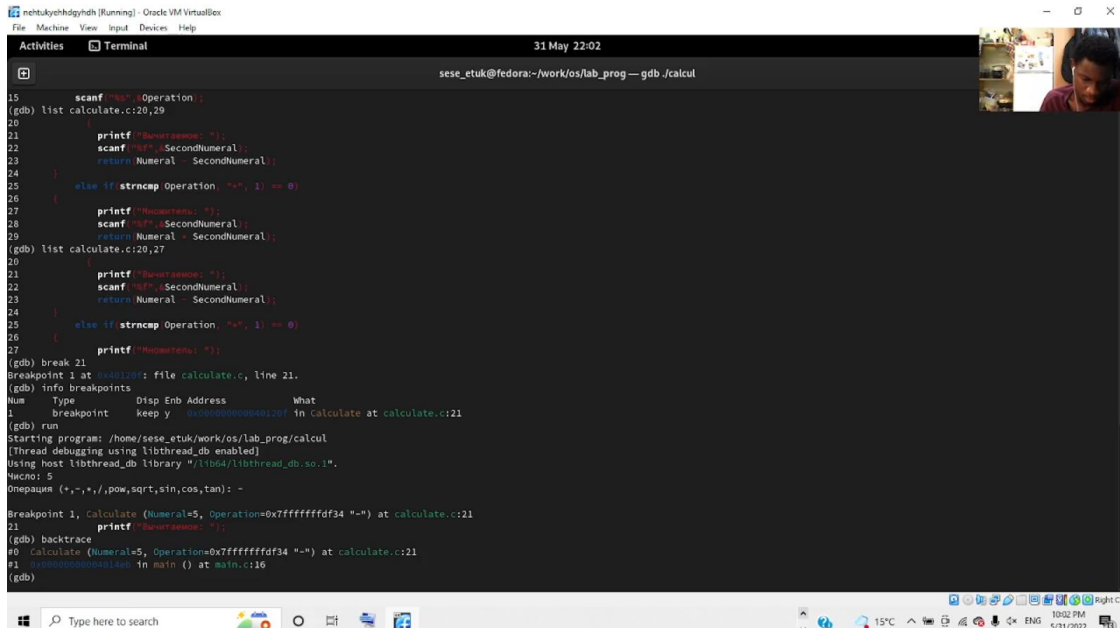
The screenshot shows a GDB terminal window with the following commands and output:

```
(gdb) break 21
Breakpoint 1 at 0x401207: file calculate.c, line 21.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x0000000000401207 in calculate at calculate.c:21
(gdb)
```

– Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова:

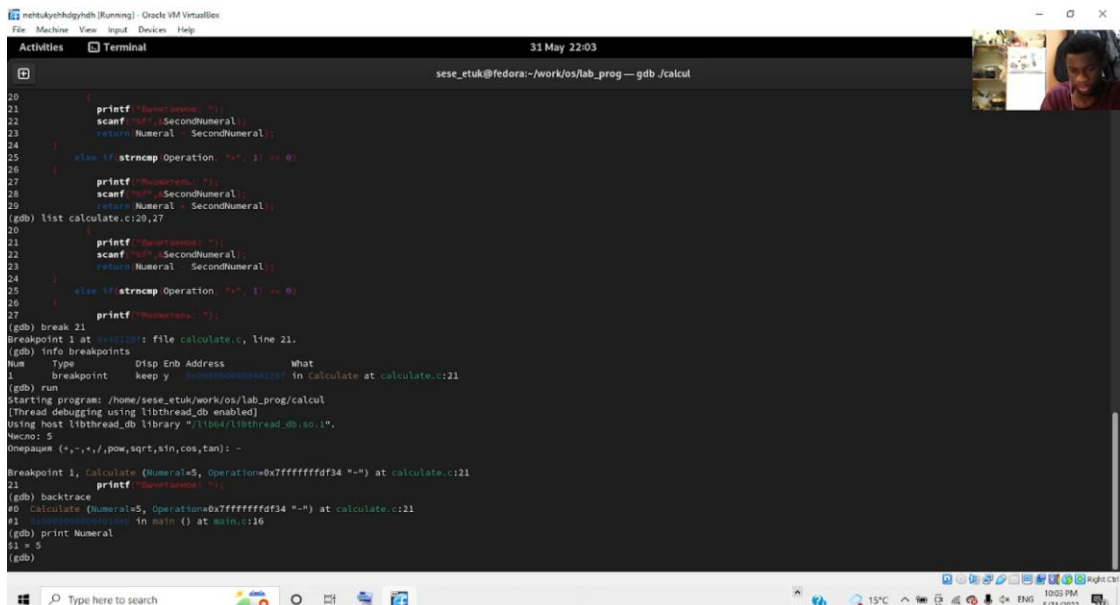
– Отладчик выдаст следующую информацию:

а команда `backtrace` покажет весь стек вызываемых функций от начала программы до текущего места.



```
15 scanf("%d", &Operation);
(gdb) list calculate.c:20,29
20
21     printf("Numeral: ");
22     scanf("%d", &SecondNumeral);
23     return Numeral * SecondNumeral;
24 }
25 else if (strcmp Operation, "+", 1) == 0)
26 {
27     printf("Numeral: ");
28     scanf("%d", &SecondNumeral);
29     return Numeral + SecondNumeral;
(gdb) list calculate.c:20,27
20
21     printf("Numeral: ");
22     scanf("%d", &SecondNumeral);
23     return Numeral * SecondNumeral;
24 }
25 else if (strcmp Operation, "+", 1) == 0)
26 {
27     printf("Numeral: ");
(gdb) break 21
Breakpoint 1 at 0x401201: file calculate.c, line 21.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x00000000401201 in Calculate at calculate.c:21
(gdb) run
Starting program: /home/sese_etuk/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+, -, *, /, pow, sqrt, sin, cos, tan): -
Breakpoint 1, calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
21     printf("Numeral: ");
(gdb) backtrace
#0 calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
#1 0x00000000401100 in main () at main.c:16
(gdb)
```

– Посмотрите, чему равно на этом этапе значение переменной `Numeral`, введя:



```
20
21     printf("Numeral: ");
22     scanf("%d", &SecondNumeral);
23     return Numeral * SecondNumeral;
24 }
25 else if (strcmp Operation, "+", 1) == 0)
26 {
27     printf("Numeral: ");
28     scanf("%d", &SecondNumeral);
29     return Numeral + SecondNumeral;
(gdb) list calculate.c:20,27
20
21     printf("Numeral: ");
22     scanf("%d", &SecondNumeral);
23     return Numeral * SecondNumeral;
24 }
25 else if (strcmp Operation, "+", 1) == 0)
26 {
27     printf("Numeral: ");
(gdb) break 21
Breakpoint 1 at 0x401201: file calculate.c, line 21.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x00000000401201 in Calculate at calculate.c:21
(gdb) run
Starting program: /home/sese_etuk/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+, -, *, /, pow, sqrt, sin, cos, tan): -
Breakpoint 1, calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
21     printf("Numeral: ");
(gdb) backtrace
#0 calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
#1 0x00000000401100 in main () at main.c:16
(gdb) print Numeral
$1 = 5
(gdb)
```

– Сравните с результатом вывода на экран после использования команды:

```
root@yehjshgdn: [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal 31 May 22:04
sese_etuk@fedora:~/work/os/lab_prog -- gdb ./calc

22     scanf("%f", &SecondNumeral);
23     return Numeral - SecondNumeral;
24 }
25 else if (strcmp Operation, "+", 1) == 0)
26 {
27     printf("Power2code: ");
28     scanf("%f", &SecondNumeral);
29     return Numeral + SecondNumeral;
(gdb) list calculate.c:20,27
20 }
21     printf("Power2code: ");
22     scanf("%f", &SecondNumeral);
23     return Numeral - SecondNumeral;
24 }
25 else if (strcmp Operation, "+", 1) == 0)
26 {
27     printf("Power2code: ");
(gdb) break 21
Breakpoint 1 at 0x00000000: file calculate.c, line 21.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x0000000000000021 in calculate at calculate.c:21
(gdb) run
Starting program: /home/sese_etuk/work/os/lab_prog/calc
[thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
21     printf("Power2code: ");
(gdb) backtrace
#0 calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
#1 calculate (Numeral=5) in main () at main.c:16
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb)
```

– Уберите точки останова:

The image shows a Linux desktop environment with a terminal window open. The terminal title is "Terminal" and the date is "31 May 22:05". The user is running a C program named "calculate.c" using GDB. The program takes two numbers as input and prints their sum. The terminal output shows the program running successfully, printing "Sum=10". A video call window is visible in the top right corner, showing a person's face. The terminal window has a dark background and white text. The video call window has a light background and shows a person's face. The desktop background is dark. The terminal window is in the foreground, and the video call window is in the background.

7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`.

The image shows a Windows 10 desktop environment. At the top, a taskbar contains icons for a web browser, file explorer, and other applications. The main area is occupied by a terminal window titled "Terminal" with the date "31 May 22:12" and the user "este_ekufedora:~/work/os/lab_prog". The terminal displays C code for a function named "calculate" and its implementation. The code includes various mathematical operations and type casting. The compiler output shows several warnings, such as "Function parameter Operation declared as manifest array (size constant is meaningless)" and "Return value type double does not match declared type float". A video call window is visible in the top right corner, showing a person's face. The bottom of the screen shows the Windows taskbar with the search bar and several pinned applications.

Вывод

Я приобрел простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования. С калькулятора с простейшими функциями.

Контрольные вопросы

1. Дополнительную информацию о этих программах можно получить с помощью функций `info` и `man`.
2. Unix поддерживает следующие основные этапы разработки приложений:
 - создание исходного кода программы;
 - представляется виде файла;
 - сохранение различных вариантов исходного текста;

анализ исходного текста; необходимо отслеживать изменения исходного кода, а также при работе более двух программистов над проектом программы нужно, чтобы они не делали изменений кодов одно время.

компиляция исходного текста и построение исполняемого модуля;
тестирование и отладка;

проверка кода на наличие ошибок

сохранение всех изменений, выполняемых при тестировании и отладке.

3. Использование суффикса ".c" для имени файла с программой на языке Си отражает удобное и полезное соглашение, принятое ОС UNIX. Для любого имени входного файла суффикс определяет какая компиляция требуется. Суффиксы и префиксы указывают тип объекта. Одно из полезных свойств компилятора Си — его способность по суффиксам определять типы файлов. По суффиксу .c компилятор распознает, что файлabcd.c должен компилироваться, а по суффиксу .o, что файлabcd.o является объектным модулем и для получения исполняемой программы необходимо выполнить редактирование связей. Простейший пример командной строки для компиляции программы abcd.c и построения исполняемого модуля abcd имеет вид: gcc -o abcd abcd.c. Некоторые проекты предпочитают показывать префиксы в начале текста изменений для старых (old) и новых (new) файлов. Опция – prefix может быть использована для установки такого префикса. Плюс этому команда bzr diff -p1 выводит префиксы в форме, которая подходит для команды patch -p1.

4. Основное назначение компилятора с языка C и заключается компиляции всей программы в целом и получении исполняемого модуля.

5. При разработке большой программы, состоящей из нескольких исходных файлов заголовков, приходится постоянно следить за файлами, которые требуют перекомпиляции после внесения изменений. Программа make освобождает пользователя от такой рутинной работы и служит для документирования взаимосвязей между файлами. Описание взаимосвязей и соответствующих действий хранится так называемом make-файле, который по умолчанию имеет имя makefile или Makefile.

6. makefile для программы abcd.c мог бы иметь вид:

#

#

Makefile

```
#
CC = gcc
CFLAGS =
LIBS = -lm

calcul: calculate.o main.o gcc calculate.o main.o -o calcul $(LIBS) calculate.o: calculate.c
calculate.h gcc -c calculate.c $(CFLAGS) main.o: main.c calculate.h gcc -c main.c $(CFLAGS)
clean: -rm calcul *.o *~

#End Makefile
```

В общем случае make-файл содержит последовательность записей (строк), определяющих зависимости между файлами. Первая строка записи представляет собой список целевых (зависимых) файлов, разделенных пробелами, за которыми следует двоеточие и список файлов, от которых зависят целевые. Текст, следующий за точкой с запятой, и все последующие строки, начинающиеся солитёры табуляции, являются командами ОС UNIX, которые необходимо выполнить для обновления целевого файла. Таким образом, спецификация взаимосвязей имеет формат: target1 [target2...]: [:] [dependment1...] [(tab)commands] [#commentary] [(tab)commands] [#commentary], где # — специфицирует начало комментария, так как содержимое строки, начиная с # и до конца строки, не будет обрабатываться командой make; : — последовательность команд ОС UNIX должна содержаться одной строке make-файла (файла описаний), есть возможность переноса команд (), но она считается как одна строка; :: — последовательность команд ОС UNIX может содержаться нескольких последовательных строках файла описаний. Приведённый выше make-файл для программы abcd. включает два способа компиляции и построения исполняемого модуля. Первый способ предусматривает обычную компиляцию с построением исполняемого модуля с именем abcd. Второй способ позволяет включать в исполняемый модуль testabcd возможность выполнить процесс отладки на уровне исходного текста.

9. 1. Выполнили компиляцию программы

2. Увидели ошибки в программе

3. Открыли редактор и исправили программу

4. Загрузили программу в отладчик gdb

5. run — отладчик выполнил программу, мы ввели требуемые значения.

6. программа завершена, gdb невидит ошибок.

10. 1 и 2.) Мы действительно забыли закрыть комментарии; 3.) отладчику не понравился формат %s для &Operation, т.к %s — символьный формат, а значит необходим только Operation.

11. Если вы работаете с исходным кодом, который не вами разрабатывался, то назначение различных конструкций может быть не совсем понятным. Система разработки приложений UNIX предоставляет различные средства, повышающие понимание исходного кода. К ним относятся: — cscope- исследование функций, содержащихся в программе; — splint — критическая проверка программ, написанных на языке Си.