

Лабораторная работа-13

**Средства, применяемые при разработке программного обеспечения в
ОС типа UNIX/Linux**

Бровкин Александр НБИбд-01-21

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Вывод:	17
5	Ответы на контрольные вопросы:	18

Список иллюстраций

3.1	Создаю подкаталог	8
3.2	Создаю файлы	9
3.3	Скрипт	9
3.4	Пишу скрипт	9
3.5	Скрипт	10
3.6	Основной файл мэйн	10
3.7	Компиляция	11
3.8	Файлы	11
3.9	Создал Мэйкфайл	11
3.10	Исправил мэйкфайл	12
3.11	Запустил программу	12
3.12	Использую команды	13
3.13	Использую команды	13
3.14	Параметры Лист	14
3.15	Информация о точках останова	14
3.16	Запуск программы	15
3.17	Сравнение	15
3.18	Удаляю точки останова	15
3.19	splint	16
3.20	splint	16

Список таблиц

1 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.

2 Задание

1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`.
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 результаты и остановится. Реализация функций калькулятора в файле `calculate.h`:

Интерфейсный файл `calculate.h`, описывающий формат вызова функции-калькулятора:

3. Выполните компиляцию программы посредством `gcc`:
4. При необходимости исправьте синтаксические ошибки.
5. Создайте `Makefile` со следующим содержанием: Поясните в отчёте его содержание.
6. С помощью `gdb` выполните отладку программы `calcul` (перед использованием `gdb` исправьте `Makefile`): – Запустите отладчик GDB, загрузив в него программу для отладки: Кулябов Д.С. и др. Операционные системы 109 1
`gdb ./calcul` – Для запуска программы внутри отладчика введите команду `run`: 1 `run` – Для постраничного (по 9 строк) просмотра исходного код

используйте команду list: 1 list – Для просмотра строк 12 по 15 основного файла используйте list с параметрами: 1 list 12,15 – Для просмотра определённых строк не основного файла используйте list с параметрами: 1 list calculate.c:20,29 – Установите точку останова в файле calculate.c на строке номер 21: 1 list calculate.c:20,27 2 break 21 – Выведите информацию об имеющихся в проекте точках останова: 1 info breakpoints – Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова: 1 run 2 5 3 - 4 backtrace – Отладчик выдаст следующую информацию: 1 #0 Calculate (Numeral=5, Operation=0x7fffffff280 “-”) 2 at calculate.c:21 3 #1 0x00000000400b2b in main () at main.c:17 а команда backtrace покажет весь стек вызываемых функций от начала программы до текущего места. – Посмотрите, чему равно на этом этапе значение переменной Numeral, введя: 1 10 Лабораторная работа No 13. Средства, применяемые при разработке программного... 1 print Numeral На экран должно быть выведено число 5. – Сравните с результатом вывода на экран после использования команды: 1 display Numeral – Уберите точки останова: 1 info breakpoints 2 delete 1

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

3 Выполнение лабораторной работы

1. В домашнем каталоге создал подкаталог ~/work/os/lab_prog.(рис. 3.1)

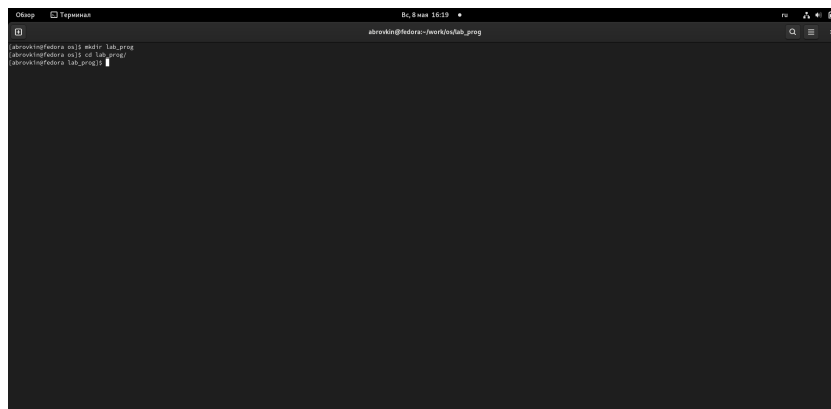


Рис. 3.1: Создаю подкаталог

2. Создал в нём файлы: calculate.h, calculate.c, main.c. Это примитивнейший калькулятор, способный складывать, вычитать, умножать, делить, возводить число в степень, вычислять квадратный корень, вычислять sin, cos, tan. При запуске он запрашивает первое число, операцию, второе число. После этого программа выводит результат и останавливается.(рис. 3.2)

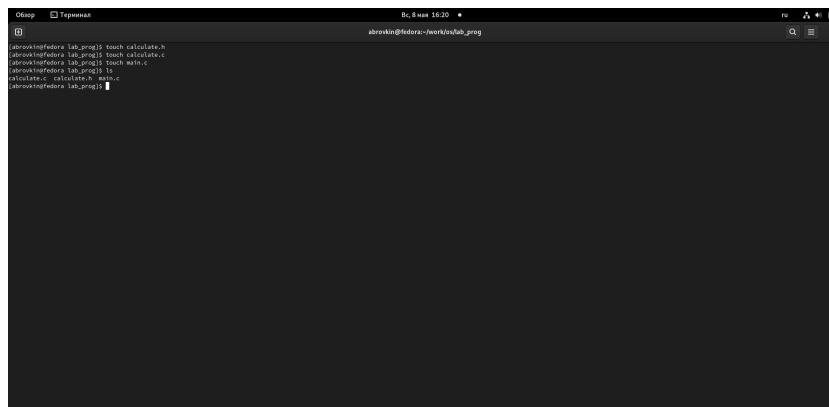


Рис. 3.2: Создаю файлы

Реализация функций калькулятора в файле calculate.c:(рис. 3.3)(рис. 3.4)

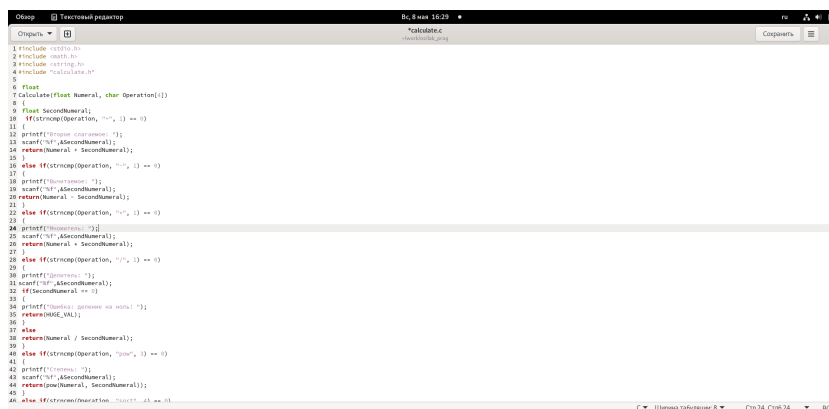


Рис. 3.3: Скрипт

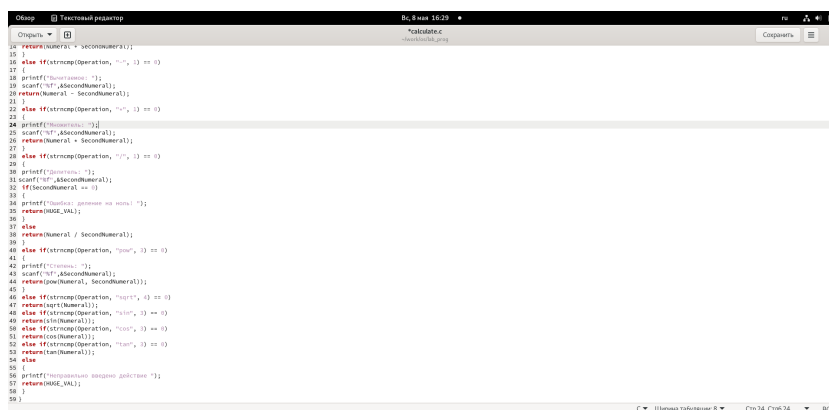
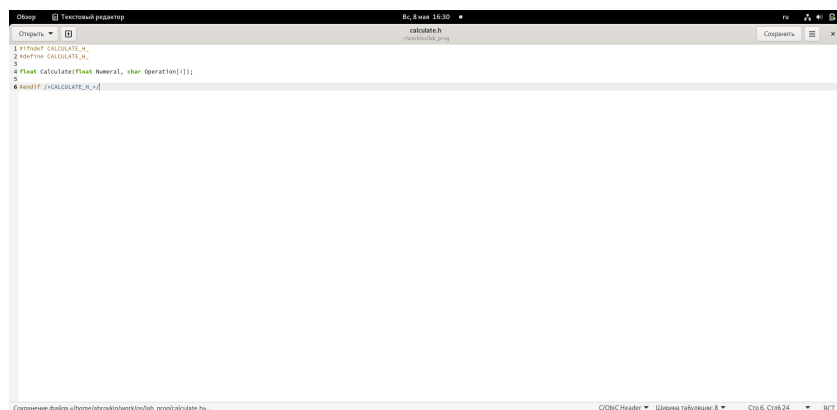


Рис. 3.4: Пишу скрипт

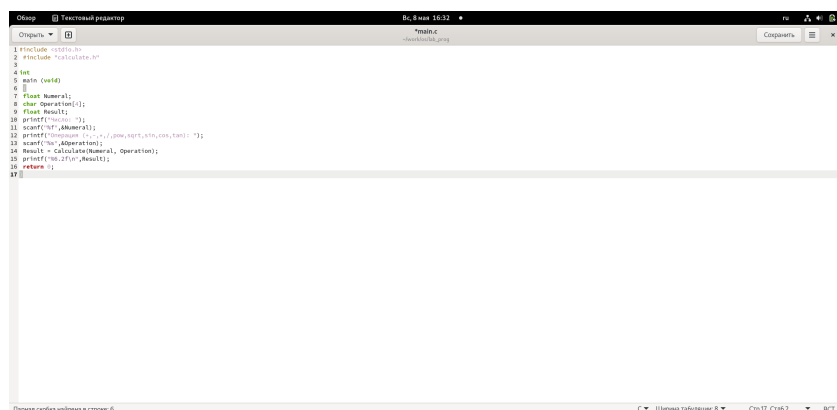
Интерфейсный файл calculate.h, описывающий формат вызова функции калькулятора:(рис. 3.5)



```
1 #ifndef CALCULATE_H
2 #define CALCULATE_H
3
4 float Calculate(float Number1, char Operation[]);
5
6 #endif // !CALCULATE_H
```

Рис. 3.5: Скрипт

Основной файл main.c, реализующий интерфейс пользователя к калькулятору:(рис. 3.6)



```
1 #include <stdio.h>
2 #include "calculate.h"
3
4 int
5 main(void)
6 {
7     float Number1;
8     char Operation[];
9     float Result;
10    printf("Введите: ");
11    scanf("%f", &Number1);
12    printf("Операция (+, -, *, /, pow, sqrt, xln, cos, tan): ");
13    scanf("%s", &Operation);
14    Result = Calculate(Number1, Operation);
15    printf("Результат: %f\n", Result);
16    return 0;
17 }
```

Рис. 3.6: Основной файл мейн

3.Выполнил компиляцию программы посредством gcc:(рис. 3.7)(рис. 3.8)

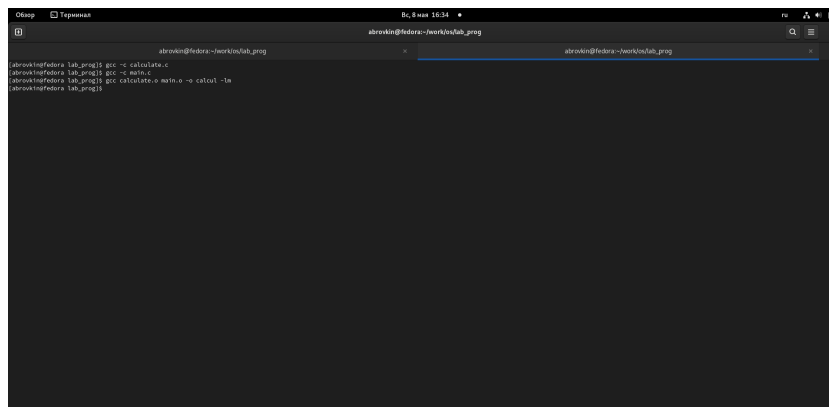


Рис. 3.7: Компиляция

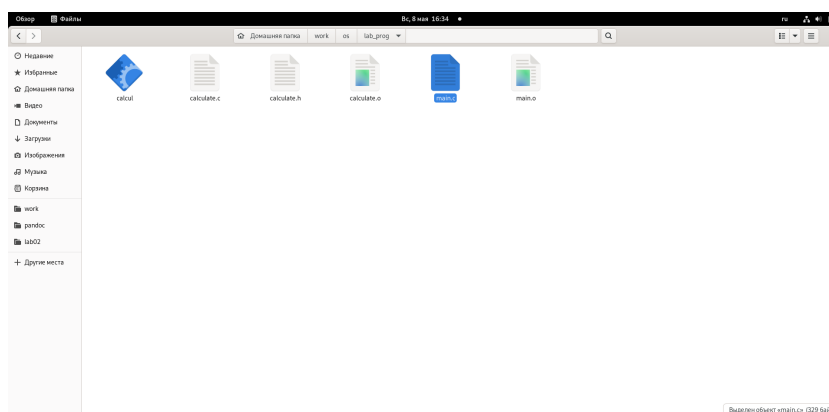


Рис. 3.8: Файлы

4. Исправил синтаксические ошибки.

5. Создал Makefile.(рис. 3.9)

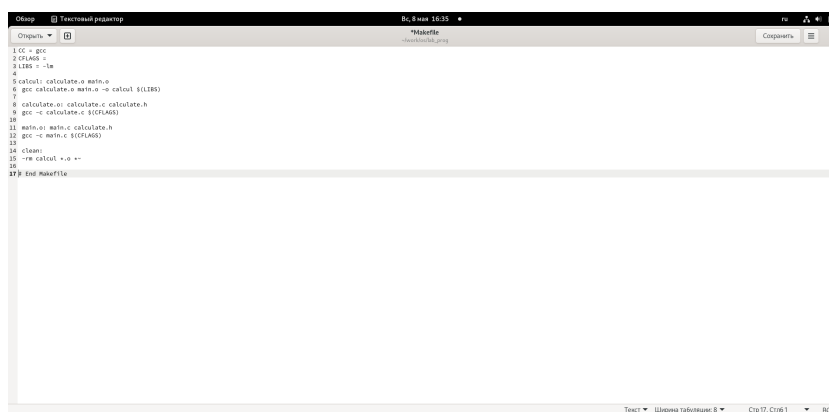


Рис. 3.9: Создал Мэйкфайл

В содержании файла указаны флаги компиляции, тип компилятора и файлы, которые должен собрать сборщик.

6. С помощью gdb выполнил отладку программы calcul (перед использованием gdb исправил Makefile): – запустите отладчик GDB, загрузив в него программу для отладки: `gdb ./calcul` – для запуска программы внутри отладчика ввел команду `run`(рис. 3.10)(рис. 3.11)

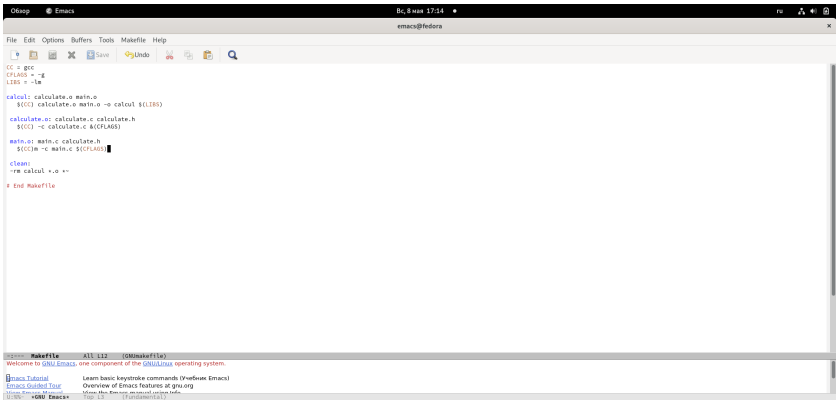


Рис. 3.10: Исправил мэйкфайл

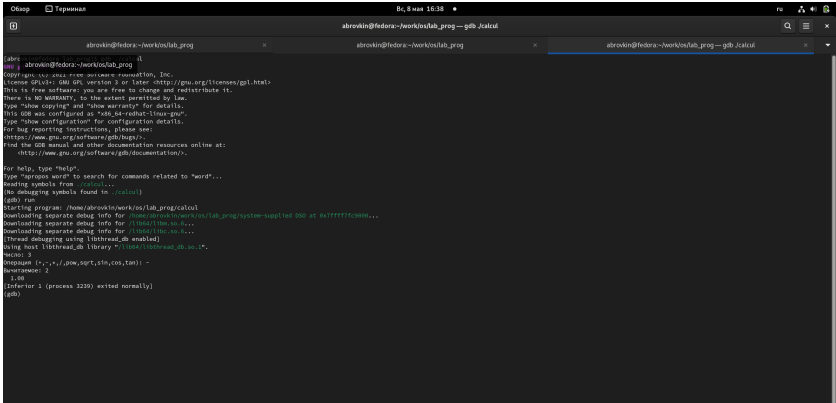


Рис. 3.11: Запустил программу

– для постраничного (по 9 строк) просмотра исходного код использовал команду `list` – для просмотра строк с 12 по 15 основного файла использовал `list` с параметрами: `list 12,15`(рис. 3.12)(рис. 3.13)

```

abrovin@fedora-lab-prog ~$ gcc calculate.o
abrovin@fedora-lab-prog ~$ ./calculate
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Рис. 3.12: Используя команды

```

abrovin@fedora-lab-prog ~$ ./calculate
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Рис. 3.13: Используя команды

– для просмотра определённых строк не основного файла использовал list с параметрами: list calculate.c:20,29 – установил точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 20 – вывел информацию об имеющихся в проекте точка останова: info breakpoints(рис. 3.14)(рис. 3.15)

```

abrokin@fedora-workbook:~$ gdb ./calcul
(gdb) list
19      }
20      }
21      }
22      }
23      }
24      }
25      }
26      }
27      }
(gdb) list calculate.c:20,27
20      }
21      }
22      }
23      }
24      }
25      }
26      }
27      }
(gdb) list calculate.c:20,27
20      }
21      }
22      }
23      }
24      }
25      }
26      }
27      }
(gdb) break 21
Breakpoint 1 at calculate.c:21:22.
(gdb)

```

Рис. 3.14: Параметры Лист

```

abrokin@fedora-workbook:~$ gdb ./calcul
(gdb) info breakpoints
Breakpoint 1 at 0x0000000000401000: File calculate.c, line 22.
Breakpoint 2 at 0x0000000000401004: File calculate.c, line 26.
Breakpoint 3 at 0x0000000000401008: File calculate.c, line 28.
Breakpoint 4 at 0x000000000040100c: File calculate.c, line 22.
(gdb)

```

Рис. 3.15: Информация о точках останова

– запустил программу внутри отладчика и убедился, что программа остановится в момент прохождения точки останова – отладчик выдал следующую информацию, а команда backtrace показала весь стек вызываемых функций от начала программы до текущего места: – посмотрел, чему равно на этом этапе значение переменной Numeral, введя: print Numeral – сравнил с результатом вывода на экран после использования команды: display Numeral – убрал точки останова: info breakpoints delete 1(рис. 3.16)(рис. 3.17)(рис. 3.18)

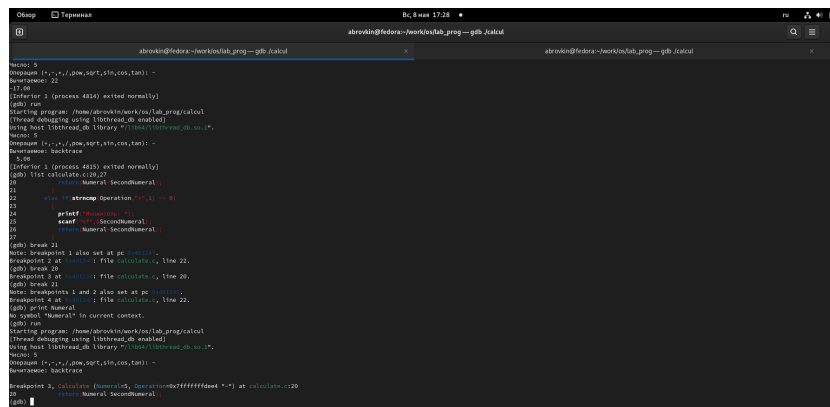


Рис. 3.16: Запуск программы

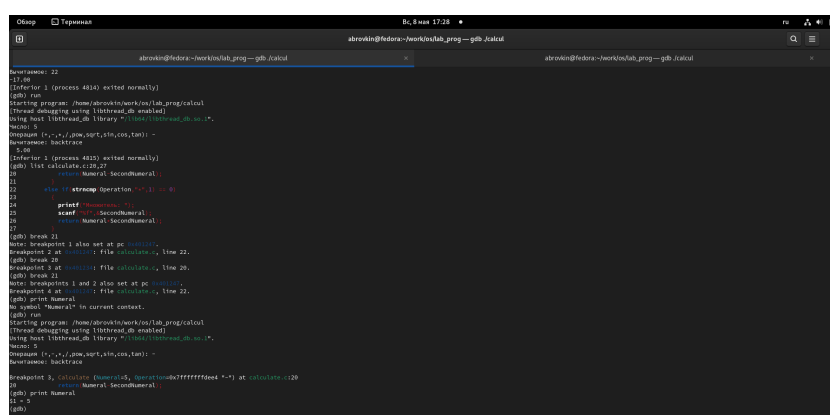


Рис. 3.17: Сравнение

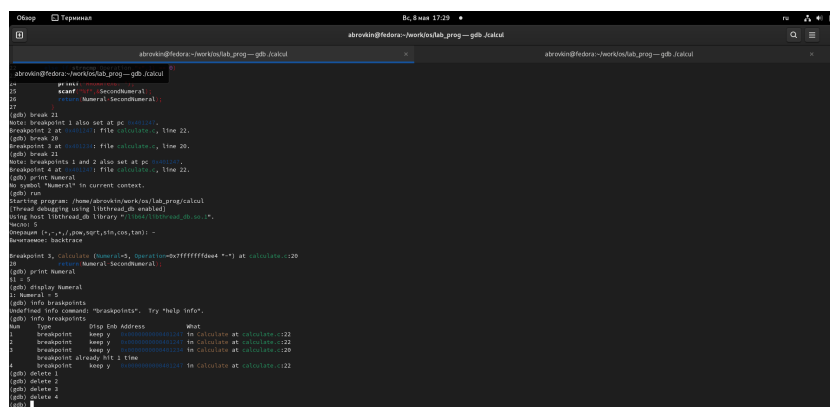
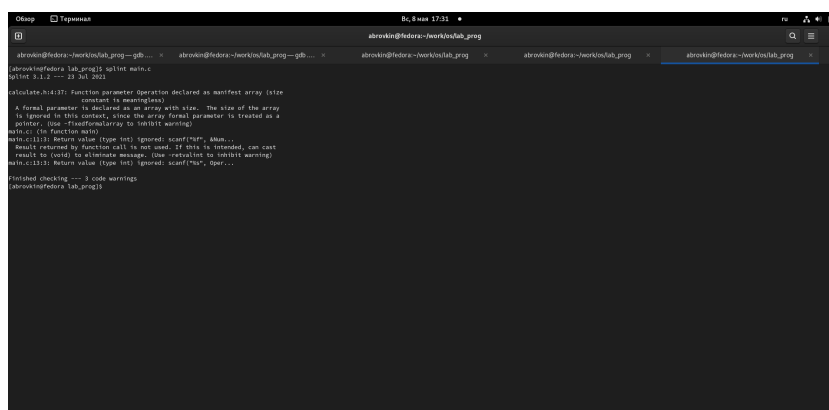
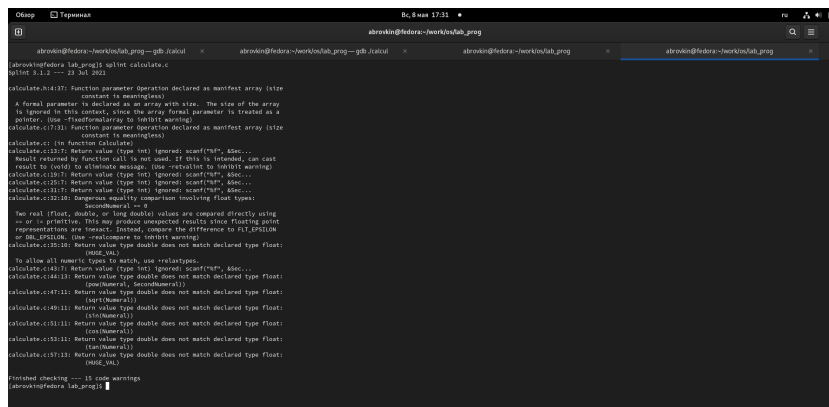


Рис. 3.18: Удаляю точки останова

7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c` (рис. 3.19)(рис. 3.20)



4 Вывод:

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.

5 Ответы на контрольные вопросы:

1. Информацию об этих программах можно получить с помощью функций `info` и `man`.
2. Unix поддерживает следующие основные этапы разработки приложений:
 - создание исходного кода программы; - представляется в 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 кода в 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 в 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 файл `abcd.c` должен компилироваться, а по суффиксу .o, что файл `abcd.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 к этому команда `bzr diff -p1` выводит префиксы в форме которая подходит для команды `patch -p1`.

4. Основное назначение компилятора с языка Си заключается в 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 в так называемом `make-файле`, который по умолчанию имеет имя `makefile` или `Makefile`.
6. 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 с литеры 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 в одной строке `make-файла` (файла описаний), есть возможность переноса команд `()`, но она считается как одна строка; `::` — последовательность команд ОС UNIX может содержаться в

нескольких последовательных строках файла описаний. Приведённый выше make-файл для программы abcd.c 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. Пример можно найти в задании 5.

7. Пошаговая отладка программ заключается в том, что выполняется один оператор программы и, затем контролируются те переменные, на которые должен был воздействовать данный оператор. Если в программе имеются уже отлаженные подпрограммы, то подпрограмму можно рассматривать, как один оператор программы и воспользоваться вторым способом отладки программ. Если в программе существует достаточно большой участок программы, уже отлаженный ранее, то его можно выполнить, не контролируя переменные, на которые он воздействует. Использование точек останова позволяет пропускать уже отлаженную часть программы. Точка останова устанавливается в местах, где необходимо проверить содержимое переменных или просто проконтролировать, передаётся ли управление данному оператору. Практически во всех отладчиках поддерживается это свойство (а также выполнение программы до курсора и выход из подпрограммы). Затем отладка программы продолжается в пошаговом режиме с контролем локальных и глобальных переменных, а также внутренних регистров микроконтроллера и напряжений на выводах этой микросхемы.
8. backtrace - вывод на экран пути к текущей точке останова (по сути вывод названий всех функций) break - установить точку останова (в качестве параметра может быть указан номер строки или название функции) clear - удалить все точки останова в функции continue - продолжить выполнение программы delete - удалить точку останова display - добавить выражение

в список выражений, значения которых отображаются при достижении точки останова программы `finish` - выполнить программу до момента выхода из функции `info breakpoints` - вывести на экран список используемых точек останова `info watchpoints` - вывести на экран список используемых контрольных выражений `list` - вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк) `next` - выполнить программу пошагово, но без выполнения вызываемых в программе функций `print` - вывести значение указываемого в качестве параметра выражения `run` - запуск программы на выполнение `set` - установить новое значение переменной `step` - пошаговое выполнение программы `watch` - установить контрольное выражение, при изменении значения которого программа будет остановлена

9. 1) Выполнила компиляцию программы 2) Увидела ошибки в программе
 3) Открыла редактор и исправила программу 4) Загрузила программу в отладчик `gdb` 5) `run` — отладчик выполнил программу, ввела требуемые значения. 6) Использовала другие команды отладчика и проверила работу программы
10. 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.
12. 1. Проверка корректности задания аргументов всех использованных в программе функций, а также типов возвращаемых ими значений; 2. Поиск фрагментов исходного текста, корректных с точки зрения синтак-

сиса языка Си, но малоэффективных с точки зрения их реализации или содержащих в себе семантические ошибки; 3.Общая оценка мобильности пользовательской программы.