

Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция

Выполнил студент гр. 3530901/10005
Калашников О. Ю.

(подпись)

Преподаватель
Коренев Д.А.

(подпись)

“ ____ ” _____ 2022 г.

Санкт-Петербург
2022

Оглавление

ТЗ	3
Метод решения	3
Реализация программ на языке С	4
Тестовая программа	4
Функциональная часть.....	4
Заголовочный файл	4
Препроцессирование	5
Компиляция	6
Ассемблирование	7
Компоновка.....	11
Создание статической библиотеки и Makefile	13
Вывод	15

ТЗ

Написать программу на языке C, которая реализует нахождение максимального элемента массива. Также поместить определение функции в отдельный исходный файл, оформить заголовочный файл и разработать тестовую программу на языке C.

Пошагово собрать программу. Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся в объектных файлах и исполняемом файле.

Выделить разработанную функцию в статическую библиотеку. Разработать Makefile для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

Метод решения

Для нахождения максимального элемента массива необходимо циклически пройти каждый элемент, сравнивая его с максимальным значением, найденным на данный момент (для первого элемента текущее максимальное значение принимается за 0).

Например, рассмотрим массив [3, 5, 9, 4, 1]:

1. $3 > 0 \Rightarrow 3$ — текущий максимальный элемент;
2. $5 > 3 \Rightarrow 5$ — текущий максимальный элемент;
3. $9 > 5 \Rightarrow 9$ — текущий максимальный элемент;
4. $4 < 9 \Rightarrow 9$ — остаётся текущим максимальным элементом;
5. $1 < 9 \Rightarrow 9$ — остаётся текущим максимальным элементом;

Ответ: 9.

Реализация программ на языке C

Тестовая программа

```
1 #include <stdio.h>
2 #include "maxnum.h"
3
4 int main(void) {
5     int array[] = {10, 5, 3, 2, 12, 69, 35, 69, 70, 11};
6     size_t array_length = sizeof(array)/sizeof(array[0]);
7     printf("%d", maxNum(array, array_length));
8     printf("\n");
9     return 0;
10 }
```

Функциональная часть

```
1 #include <stdio.h>
2 #include "maxnum.h"
3
4 int maxNum(int array[], size_t array_length) {
5     int max = array[0];
6     for (int i = 0; i < array_length; i++) {
7         if (array[i] > max){
8             max = array[i];
9         }
10    }
11    return max;
12 }
```

Заголовочный файл

```
1 #ifndef MAXNUM_H
2 #define MAXNUM_H
3
4 int maxNum(int array[], size_t array_length);
5
6 #endif
```

Препроцессирование

Для препроцессирования необходимо выполнить следующие команды:

```
riscv64-unknown-elf-gcc -O1 -E main.c -o main.i
```

```
riscv64-unknown-elf-gcc -O1 -E maxnum.c -o maxnum.i
```

Соответственно результаты препроцессирования будут находится в файлах main.i и maxnum.i:

main.i:

```
1  # 1 "main.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "main.c"
5
6  ...
7
8  # 2 "main.c" 2
9  # 1 "maxnum.h" 1
10
11
12
13
14  # 4 "maxnum.h"
15  int maxNum(int array[], size_t array_length);
16  # 3 "main.c" 2
17
18  int main(void) {
19      int array[] = {10, 5, 3, 2, 12, 69, 35, 69, 70, 11};
20      size_t array_length = sizeof(array)/sizeof(array[0]);
21      printf("%d", maxNum(array, array_length));
22      printf("\n");
23      return 0;
24  }
25
```

maxnum.i:

```
1  # 1 "maxnum.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "maxnum.c"
5
6  ...
7
8  # 2 "maxnum.c" 2
9  # 1 "maxnum.h" 1
10
11
12
13
14  # 4 "maxnum.h"
15  int maxNum(int array[], size_t array_length);
16  # 3 "maxnum.c" 2
17
18  int maxNum(int array[], size_t array_length) {
19      int max = array[0];
20      for (int i = 0; i < array_length; i++) {
21          if (array[i] > max){
22              max = array[i];
23          }
24      }
25      return max;
26  }
```

Компиляция

Для компилирования необходимо выполнить следующие команды:

```
riscv64-unknown-elf-gcc -O1 -S main.i -o main.s
```

```
riscv64-unknown-elf-gcc -O1 -S maxnum.i -o maxnum.s
```

Результат компилирования будет находится в файлах main.s и maxnum.s:

main.s:

```
1  .file "main.c"
2  .option nopic
3  .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
4  .attribute unaligned_access, 0
5  .attribute stack_align, 16
6  .text
7  .align 1
8  .globl main
9  .type main, @function
10 main:
11     addi sp,sp,-64
12     sd ra,56(sp)
13     lui a5,%hi(.LANCHOR0)
14     addi a5,a5,%lo(.LANCHOR0)
15     ld a1,0(a5)
16     ld a2,8(a5)
17     ld a3,16(a5)
18     ld a4,24(a5)
19     ld a5,32(a5)
20     sd a1,8(sp)
21     sd a2,16(sp)
22     sd a3,24(sp)
23     sd a4,32(sp)
24     sd a5,40(sp)
25     li a1,10
26     addi a0,sp,8
27     call maxNum
28     mv a1,a0
29     lui a0,%hi(.LC1)
30     addi a0,a0,%lo(.LC1)
31     call printf
32     li a0,10
33     call putchar
34     li a0,0
35     ld ra,56(sp)
36     addi sp,sp,64
37     jr ra
38     .size main, .-main
39     .section .rodata
40     .align 3
41     .set .LANCHOR0, . + 0
42 .LC0:
43     .word 10
44     .word 5
45     .word 3
46     .word 2
47     .word 12
48     .word 69
49     .word 35
50     .word 69
51     .word 70
52     .word 11
53     .section .rodata.str1.8,"aMS",@progbits,1
54     .align 3
55 .LC1:
56     .string "%d"
57     .ident "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

maxnum.s:

```
1      .file    "maxnum.c"
2      .option nopic
3      .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
4      .attribute unaligned_access, 0
5      .attribute stack_align, 16
6      .text
7      .align 1
8      .globl  maxNum
9      .type   maxNum, @function
10     maxNum:
11         lw    a3,0(a0)
12         beq   a1,zero,.L2
13         mv    a5,a0
14         slli  a1,a1,2
15         add   a0,a0,a1
16         j     .L4
17     .L3:
18         sext.w a3,a4
19         addi  a5,a5,4
20         beq   a5,a0,.L2
21     .L4:
22         lw    a4,0(a5)
23         sext.w a2,a4
24         bge   a2,a3,.L3
25         mv    a4,a3
26         j     .L3
27     .L2:
28         mv    a0,a3
29         ret
30     .size    maxNum, .-maxNum
31     .ident   "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

Ассемблирование

Для ассемблирования необходимо выполнить следующие команды:

```
riscv64-unknown-elf-gcc -v -c main.s -o main.o
```

```
riscv64-unknown-elf-gcc -v -c maxnum.s -o maxnum.o
```

В результате ассемблирования мы получаем объектные файлы, которые не являются текстовыми — используем утилиту `objdump`, чтобы получить содержимое бинарных файлов в текстовом виде. Используем следующую команду для отображения заголовков секций файлов:

```
riscv64-unknown-elf-objdump -h main.o
```

А для получения таблицы символов используем команду:

```
riscv64-unknown-elf-objdump -t main.o
```

Заголовки секций main.o:

main.o: file format elf64-littleriscv

Sections:

Idx	Name	Size	VMA	LMA	File off	Algn
0	.text	00000050	0000000000000000	0000000000000000	00000040	2**1
			CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE			
1	.data	00000000	0000000000000000	0000000000000000	00000090	2**0
			CONTENTS, ALLOC, LOAD, DATA			
2	.bss	00000000	0000000000000000	0000000000000000	00000090	2**0
			ALLOC			
3	.rodata	00000028	0000000000000000	0000000000000000	00000090	2**3
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
4	.rodata.str1.8	00000003	0000000000000000	0000000000000000	000000b8	2**3
			CONTENTS, ALLOC, LOAD, READONLY, DATA			
5	.comment	00000029	0000000000000000	0000000000000000	000000bb	2**0
			CONTENTS, READONLY			
6	.riscv.attributes	00000035	0000000000000000	0000000000000000	000000e4	2**0
			CONTENTS, READONLY			

Таблица символов main.o:

main.o: file format elf64-littleriscv

SYMBOL TABLE:

```
0000000000000000 1 df *ABS* 0000000000000000 main.c
0000000000000000 1 d .text 0000000000000000 .text
0000000000000000 1 d .data 0000000000000000 .data
0000000000000000 1 d .bss 0000000000000000 .bss
0000000000000000 1 d .rodata 0000000000000000 .rodata
0000000000000000 1 .rodata 0000000000000000 .LANCHOR0
0000000000000000 1 d .rodata.str1.8 0000000000000000 .rodata.str1.8
0000000000000000 1 .rodata.str1.8 0000000000000000 .LC1
0000000000000000 1 d .comment 0000000000000000 .comment
0000000000000000 1 d .riscv.attributes 0000000000000000 .riscv.attributes
0000000000000000 g F .text 0000000000000050 main
0000000000000000 *UND* 0000000000000000 maxNum
0000000000000000 *UND* 0000000000000000 printf
0000000000000000 *UND* 0000000000000000 putchar
```

Для получения таблицы перемещений выполним команду:

```
riscv64-unknown-elf-objdump -d -M no-aliases -r main.o
```


Таблица перемещений main.o:

main.o: file format elf64-littleriscv

Disassembly of section .text:

```
0000000000000000 <main>:
 0: 7139          c.addi16sp      sp,-64
 2: fc06          c.sdsp         ra,56(sp)
 4: 000007b7      lui          a5,0x0
                        4: R_RISCV_HI20 .LANCHOR0
                        4: R_RISCV_RELAX *ABS*
 8: 00078793      addi         a5,a5,0 # 0 <main>
                        8: R_RISCV_LO12_I .LANCHOR0
                        8: R_RISCV_RELAX *ABS*
 c: 638c          c.ld           a1,0(a5)
 e: 6790          c.ld           a2,8(a5)
10: 6b94          c.ld           a3,16(a5)
12: 6f98          c.ld           a4,24(a5)
14: 739c          c.ld           a5,32(a5)
16: e42e          c.sdsp         a1,8(sp)
18: e832          c.sdsp         a2,16(sp)
1a: ec36          c.sdsp         a3,24(sp)
1c: f03a          c.sdsp         a4,32(sp)
1e: f43e          c.sdsp         a5,40(sp)
20: 45a9          c.li           a1,10
22: 0028          c.addi4spn     a0,sp,8
24: 00000097      auipc         ra,0x0
                        24: R_RISCV_CALL   maxNum
                        24: R_RISCV_RELAX *ABS*
28: 000080e7      jalr          ra,0(ra) # 24 <main+0x24>
2c: 85aa          c.mv          a1,a0
2e: 00000537      lui          a0,0x0
                        2e: R_RISCV_HI20 .LC1
                        2e: R_RISCV_RELAX *ABS*
32: 00050513      addi         a0,a0,0 # 0 <main>
                        32: R_RISCV_LO12_I .LC1
                        32: R_RISCV_RELAX *ABS*
36: 00000097      auipc         ra,0x0
                        36: R_RISCV_CALL   printf
                        36: R_RISCV_RELAX *ABS*
3a: 000080e7      jalr          ra,0(ra) # 36 <main+0x36>
3e: 4529          c.li           a0,10
40: 00000097      auipc         ra,0x0
                        40: R_RISCV_CALL   putchar
                        40: R_RISCV_RELAX *ABS*
44: 000080e7      jalr          ra,0(ra) # 40 <main+0x40>
48: 4501          c.li           a0,0
4a: 70e2          c.ldsp         ra,56(sp)
4c: 6121          c.addi16sp     sp,64
4e: 8082          c.jr          ra
```

Аналогичные команды выполним для maxnum.o.

Заголовки секций maxnum.o:

maxnum.o: file format elf64-littleriscv

Sections:

Idx	Name	Size	VMA	LMA	File off	Align
0	.text	00000028	0000000000000000	0000000000000000	00000040	2**1
	CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE					
1	.data	00000000	0000000000000000	0000000000000000	00000068	2**0
	CONTENTS, ALLOC, LOAD, DATA					
2	.bss	00000000	0000000000000000	0000000000000000	00000068	2**0
	ALLOC					
3	.comment	00000029	0000000000000000	0000000000000000	00000068	2**0
	CONTENTS, READONLY					
4	.riscv.attributes	00000035	0000000000000000	0000000000000000	00000091	2**0
	CONTENTS, READONLY					

Таблица символов maxnum.o:

maxnum.o: file format elf64-littleriscv

SYMBOL TABLE:

0000000000000000	1	df *ABS*	0000000000000000	maxnum.c
0000000000000000	1	d .text	0000000000000000	.text
0000000000000000	1	d .data	0000000000000000	.data
0000000000000000	1	d .bss	0000000000000000	.bss
0000000000000024	1	.text	0000000000000000	.L2
0000000000000016	1	.text	0000000000000000	.L4
000000000000000c	1	.text	0000000000000000	.L3
0000000000000000	1	d .comment	0000000000000000	.comment
0000000000000000	1	d .riscv.attributes	0000000000000000	.riscv.attributes
0000000000000000	g	F .text	0000000000000028	maxNum

Таблица перемещений maxnum.o:

```
maxnum.o:      file format elf64-littleriscv
```

Disassembly of section .text:

```
0000000000000000 <maxNum>:
   0:  4114                c.lw      a3,0(a0)
   2:  c18d                c.beqz    a1,24 <.L2>
                        2: R_RISCV_RVC_BRANCH    .L2
   4:  87aa                c.mv      a5,a0
   6:  058a                c.slli    a1,0x2
   8:  952e                c.add     a0,a1
  a:  a031                c.j       16 <.L4>
                        a: R_RISCV_RVC_JUMP      .L4

000000000000000c <.L3>:
   c:  0007069b          addiw     a3,a4,0
  10:  0791                c.addi    a5,4
  12:  00a78963          beq       a5,a0,24 <.L2>
                        12: R_RISCV_BRANCH      .L2

0000000000000016 <.L4>:
  16:  4398                c.lw      a4,0(a5)
  18:  0007061b          addiw     a2,a4,0
 1c:  fed658e3          bge       a2,a3,c <.L3>
                        1c: R_RISCV_BRANCH      .L3
 20:  8736                c.mv      a4,a3
 22:  b7ed                c.j       c <.L3>
                        22: R_RISCV_RVC_JUMP      .L3

0000000000000024 <.L2>:
  24:  8536                c.mv      a0,a3
  26:  8082                c.jr      ra
```

Компоновка

Для компоновки необходимо выполнить следующую команду:

```
riscv64-unknown-elf-gcc -v main.o maxnum.o
```

При помощи следующей команды получаем фрагмент исполняемого файла a.out:

```
riscv64-unknown-elf-objdump -j .text -d -M no-aliases a.out > a.ds
```

```

68 0000000000010158 <main>:
69 10158: 7139 c.addi16sp sp,-64
70 1015a: fc06 c.sdsp ra,56(sp)
71 1015c: 67f1 c.lui a5,0x1c
72 1015e: 1c078793 addi a5,a5,448 # 1c1c0 <__clzdi2+0x3e>
73 10162: 638c c.ld a1,0(a5)
74 10164: 6790 c.ld a2,8(a5)
75 10166: 6b94 c.ld a3,16(a5)
76 10168: 6f98 c.ld a4,24(a5)
77 1016a: 739c c.ld a5,32(a5)
78 1016c: e42e c.sdsp a1,8(sp)
79 1016e: e832 c.sdsp a2,16(sp)
80 10170: ec36 c.sdsp a3,24(sp)
81 10172: f03a c.sdsp a4,32(sp)
82 10174: f43e c.sdsp a5,40(sp)
83 10176: 45a9 c.li a1,10
84 10178: 0028 c.addi4spn a0,sp,8
85 1017a: 01e000ef jal ra,10198 <maxNum>
86 1017e: 85aa c.mv a1,a0
87 10180: 6571 c.lui a0,0x1c
88 10182: 1e850513 addi a0,a0,488 # 1c1e8 <__clzdi2+0x66>
89 10186: 1d6000ef jal ra,1035c <printf>
90 1018a: 4529 c.li a0,10
91 1018c: 208000ef jal ra,10394 <putchar>
92 10190: 4501 c.li a0,0
93 10192: 70e2 c.ldsp ra,56(sp)
94 10194: 6121 c.addi16sp sp,64
95 10196: 8082 c.jr ra
96
97 0000000000010198 <maxNum>:
98 10198: 4114 c.lw a3,0(a0)
99 1019a: c18d c.beqz a1,101bc <maxNum+0x24>
100 1019c: 87aa c.mv a5,a0
101 1019e: 058a c.slli a1,0x2
102 101a0: 952e c.add a0,a1
103 101a2: a031 c.j 101ae <maxNum+0x16>
104 101a4: 0007069b addiw a3,a4,0
105 101a8: 0791 c.addi a5,4
106 101aa: 00a78963 beq a5,a0,101bc <maxNum+0x24>
107 101ae: 4398 c.lw a4,0(a5)
108 101b0: 0007061b addiw a2,a4,0
109 101b4: fed658e3 bge a2,a3,101a4 <maxNum+0xc>
110 101b8: 8736 c.mv a4,a3
111 101ba: b7ed c.j 101a4 <maxNum+0xc>
112 101bc: 8536 c.mv a0,a3
113 101be: 8082 c.jr ra

```

Создание статической библиотеки и Makefile

Для создания статической библиотеки необходимо получить объектные файлы всех используемых программ. Используем следующие команды:

```
riscv64-unknown-elf-gcc -c main.c -o main.o
```

```
riscv64-unknown-elf-gcc -c maxnum.c -o maxnum.o
```

Далее объединяем объектные файлы в библиотеку:

```
riscv64-unknown-elf-ar -rsc bsortlib.a maxnum.o
```

После получения библиотеки соберём исполняемый файл программы:

```
riscv64-unknown-elf-gcc -O1 --save-temps main.c bsortlib.a
```

Посмотрим таблицу символов исполняемого файла:

```
riscv64-unknown-elf-objdump -t a.out
```

В результате видим, что исполняемый файл содержит необходимые

символы:

```
000000000000102d0 g F .text 00000000000000aa memset
00000000000010158 g F .text 000000000000003a main
0000000000001e178 g O .sbss 0000000000000008 __malloc_max_total_mem
0000000000001a9f0 g F .text 000000000000000c __swbuf
0000000000001618c g F .text 0000000000000008 __sclose
00000000000018e8e g F .text 000000000000000a fclose
00000000000014d3e g F .text 0000000000000060 __malloc_r
0000000000001aaa0 g F .text 0000000000000024 __ascii_wctomb
00000000000012a48 g F .text 000000000000008a _fwalk
00000000000019420 g F .text 000000000000000a _mbtowc_r
00000000000012706 g F .text 00000000000000d8 __malloc_trim_r
00000000000019b84 g F .text 00000000000000ea strcmp
00000000000018c3c g F .text 0000000000000010 vfiprintf
0000000000001aeef g F .text 0000000000000066 .hidden __multf3
0000000000001606e g F .text 000000000000004c sprintf
0000000000001cd40 g O .rodata 0000000000000100 .hidden __clz_tab
0000000000001e190 g O .sbss 0000000000000008 _PathLocale
00000000000010202 g F .text 000000000000000c atexit
00000000000018ce0 g F .text 0000000000000040 _write_r
000000000000193d4 g F .text 000000000000000c setlocale
0000000000001e160 g O .sdata 0000000000000008 _impure_ptr
000000000000122a0 g F .text 00000000000000196 __sflush_r
0000000000001ad92 g F .text 00000000000000ae .hidden __gttf2
00000000000019dcc g F .text 00000000000000b26 _svfiprintf_r
0000000000001942a g F .text 0000000000000040 __ascii_mbtowc
0000000000001b4f4 g F .text 0000000000000081a .hidden __subtf3
00000000000015bf0 g F .text 0000000000000060 __ulp
000000000000126f6 g F .text 0000000000000010 __fp_unlock_all
00000000000014bfa g F .text 0000000000000006 localeconv
00000000000014c00 g F .text 0000000000000082 __swhatbuf_r
0000000000001d020 g .data 0000000000000000 __DATA_BEGIN__
0000000000001ac62 g F .text 0000000000000032 _write
0000000000001e178 g .sdata 0000000000000000 _edata
0000000000001e208 g .bss 0000000000000000 _end
00000000000018e98 g F .text 00000000000000c6 __fputwc
000000000000160f6 g F .text 0000000000000054 __swrite
0000000000001e170 g O .sdata 0000000000000008 __malloc_trim_threshold
0000000000001020e g F .text 0000000000000020 exit
00000000000017fc8 g F .text 00000000000000c74 _vfiprintf_r
00000000000012ad2 g F .text 0000000000000092 _fwalk_reent
00000000000015aa8 g F .text 00000000000000148 __mdiff
000000000000126e0 g F .text 0000000000000002 __sfp_lock_release
00000000000013cf0 g F .text 00000000000000ec8 _ldtoa_r
0000000000001cc00 g O .rodata 0000000000000101 _ctype_
0000000000001abc6 g F .text 0000000000000032 _read
0000000000001aafa g F .text 000000000000002c _exit
00000000000014c82 g F .text 00000000000000bc __smakebuf_r
00000000000016238 g F .text 0000000000000098 strlen
00000000000017fba g F .text 000000000000000e _sprint_r
00000000000010192 g F .text 0000000000000070 maxNum
0000000000001aa96 g F .text 000000000000000a _wctomb_r
```

Важно отметить, что при компоновке с использование библиотеки компоновщик сам выбрал необходимые объектные файлы.

Создадим Makefile для автоматической сборки проекта, он будет выполнять следующие действия:

1. Получение объектного файла maxnum.o из maxnum.c
2. Архивация объектного файла maxnum.o и создание статической библиотеки bsortlib.a
3. Компоновка объектного файла main.c и библиотеки bsortlib.a для получения исполняемого файла a.out

Makefile

```
.PHONY: all

all: maxnum.c
/Users/oleg/Downloads/riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-apple-darwin/bin/riscv64-unknown-elf-gcc -c maxnum.c -o maxnum.o
/Users/oleg/Downloads/riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-apple-darwin/bin/riscv64-unknown-elf-ar -rsc bsortlib.a -o maxnum.o
/Users/oleg/Downloads/riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-apple-darwin/bin/riscv64-unknown-elf-gcc -O1 --save-temps main.c bsortlib.a -o prog.out
rm *.o *.i *.s
```

Запускаем Makefile при помощи команды make:

```
oleg@Olegs-MacBook-Pro lab04 % make
/Users/oleg/Downloads/riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-apple-darwin/bin/riscv64-unknown-elf-gcc -c maxnum.c -o maxnum.o
/Users/oleg/Downloads/riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-apple-darwin/bin/riscv64-unknown-elf-ar -rsc bsortlib.a -o maxnum.o
/Users/oleg/Downloads/riscv64-unknown-elf-gcc-8.3.0-2020.04.1-x86_64-apple-darwin/bin/riscv64-unknown-elf-gcc -O1 --save-temps main.c bsortlib.a -o prog.out
rm *.o *.i *.s
```

На выходе также получаем исполняемый файл. Если посмотреть таблицу символов, то видно, что содержимое файла точно такое же:

```
000000000010158 g F .text 000000000000003a main
00000000001e178 g O .sbss 0000000000000008 __malloc_max_total_mem
00000000001a9f0 g F .text 000000000000000c __swbuf
00000000001618c g F .text 0000000000000008 __sclose
000000000018e8e g F .text 000000000000000a fclose
000000000014d3e g F .text 0000000000000060 __malloc_r
00000000001aaa0 g F .text 0000000000000024 __ascii_wctomb
000000000012a48 g F .text 000000000000008a __fwalk
000000000019420 g F .text 000000000000000a __mbtowc_r
000000000012706 g F .text 00000000000000d8 __malloc_trim_r
000000000019b84 g F .text 00000000000000ea strcmp
000000000018c3c g F .text 0000000000000010 vfiprintf
00000000001ae0e g F .text 0000000000000006 .hidden __multf3
00000000001606e g F .text 000000000000004c sprintf
00000000001cd40 g O .rodata 0000000000000100 .hidden __clz_tab
00000000001e190 g O .sbss 0000000000000008 __PathLocale
000000000010202 g F .text 000000000000000c atexit
000000000018ce0 g F .text 0000000000000040 __write_r
0000000000193d4 g F .text 000000000000000c setlocale
00000000001e160 g O .sdata 0000000000000008 __impure_ptr
0000000000122a0 g F .text 0000000000000196 __sflush_r
00000000001ad92 g F .text 00000000000000ae .hidden __gttf2
000000000019dcc g F .text 0000000000000b26 __svfiprintf_r
00000000001942a g F .text 0000000000000040 __ascii_mbtowc
00000000001b4f4 g F .text 000000000000081a .hidden __subtf3
000000000015bf0 g F .text 0000000000000060 __ulp
0000000000126f6 g F .text 0000000000000010 __fp_unlock_all
000000000014bfa g F .text 0000000000000006 localeconv
000000000014c00 g F .text 0000000000000082 __swatbuf_r
00000000001d020 g .data 0000000000000000 __DATA_BEGIN__
00000000001ac62 g F .text 0000000000000032 __write
00000000001e178 g .sdata 0000000000000000 __edata
00000000001e208 g .bss 0000000000000000 __end
000000000018e98 g F .text 00000000000000c6 __fputc
0000000000160f6 g F .text 0000000000000054 __swrite
00000000001e170 g O .sdata 0000000000000008 __malloc_trim_threshold
00000000001020e g F .text 0000000000000020 exit
000000000017fc8 g F .text 0000000000000c74 __vfiprintf_r
000000000012ad2 g F .text 0000000000000092 __fwalk_reent
000000000015aa8 g F .text 0000000000000148 __mdiff
0000000000126e0 g F .text 0000000000000002 __sfp_lock_release
000000000013cf0 g F .text 0000000000000ec8 __ldtoa_r
00000000001cc00 g O .rodata 0000000000000101 _ctype_
00000000001abc6 g F .text 0000000000000032 __read
00000000001aafa g F .text 000000000000002c __exit
000000000014c82 g F .text 00000000000000bc __smakebuf_r
000000000016238 g F .text 0000000000000098 strlen
000000000017fba g F .text 000000000000000e __sprintf_r
000000000010192 g F .text 0000000000000070 maxNum
```

Вывод

Я реализовал программу для нахождения максимального элемента на языке C.

Проделал пошаговую сборку программы, а именно препроцессирование, компиляция, ассемблирование и компоновка.

Создал статическую библиотеку и автоматизировал сборку программы с помощью Makefile.