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

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

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
исполняеммом 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 => 3 — текущий максимальный элемент;
2. 5 > 3 => 5 — текущий максимальный элемент;
3. 9 > 5 => 9 — текущий максимальный элемент;
4. 4 < 9 => 9 — остаётся текущим максимальным элементом;
5. 1 < 9 => 9 — остаётся текущим максимальным элементом;
Ответ: 9.

Реализация программ на языке C

Тестовая программа

main.c > ...

```c
#include <stdio.h>
#include "maxnum.h"

int main(void) {
    int array[] = {10, 5, 3, 2, 12, 69, 35, 69, 70, 11};
    int array_length = sizeof(array)/sizeof(array[0]);
    maxNum(array, array_length);
    for (int i = 0; i < array_length; i++) {
        printf("%d", array[i]);
    }
    return 0;
    }

```

Функциональная часть

maxnum.c > maxNum(int [], int)

```c
#include <stdio.h>
#include "maxnum.h"

void maxNum(int array[], int array_length) {
    int max = array[0];
    for (int i = 0; i < array_length-1; i++)
    {
        if (array[i] > max)
        {
            max = array[i];
        }
    }
}
```

Заголовочный файл

maxnum.h > ...

```c
#ifndef MAXNUM_H
#define MAXNUM_H

void maxNum(int array[], int array_length);

#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:

main.i > ...

```c
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"

...

# 2 "main.c" 2
# 1 "maxnum.h" 1




# 4 "maxnum.h"
void maxNum(int array[], int array_length);
# 3 "main.c" 2

int main(void) {
    int array[] = {10, 5, 3, 2, 12, 69, 35, 69, 70, 11};
    int array_length = sizeof(array)/sizeof(array[0]);
    maxNum(array, array_length);
    for (int i = 0; i < array_length; i++) {
        printf("%d", array[i]);
    }
    return 0;
    }

```

maxnum.i:

maxnum.i > ...

```
# 1 "maxnum.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "maxnum.c"

...

# 2 "maxnum.c" 2
# 1 "maxnum.h" 1



# 4 "maxnum.h"
void maxNum(int array[], int array_length);
# 3 "maxnum.c" 2

void maxNum(int array[], int array_length) {
    int max = array[0];
    for (int i = 0; i < array_length-1; i++)
    {
        if (array[i] > max)
        {
            max = array[i];
        }
    }
}

```

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:

main.s

```
    .file   "main.c"
    .option nopic
    .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .align  1
    .globl  main
    .type   main, @function
main:
    addi    sp,sp,-80
    sd  ra,72(sp)
    sd  s0,64(sp)
    sd  s1,56(sp)
    sd  s2,48(sp)
    lui a5,%hi(.LANCHOR0)
    addi    a5,a5,%lo(.LANCHOR0)
    ld  a1,0(a5)
    ld  a2,8(a5)
    ld  a3,16(a5)
    ld  a4,24(a5)
    ld  a5,32(a5)
    sd  a1,8(sp)
    sd  a2,16(sp)
    sd  a3,24(sp)
    sd  a4,32(sp)
    sd  a5,40(sp)
    li  a1,10
    addi    a0,sp,8
    call    maxNum
    addi    s0,sp,8
    addi    s2,sp,48
    lui s1,%hi(.LC1)
.L2:
    lw  a1,0(s0)
    addi    a0,s1,%lo(.LC1)
    call    printf
    addi    s0,s0,4
    bne s0,s2,.L2
    li  a0,0
    ld  ra,72(sp)
    ld  s0,64(sp)
    ld  s1,56(sp)
    ld  s2,48(sp)
    addi    sp,sp,80
    jr  ra
    .size   main, .-main
    .section    .rodata
    .align  3
    .set    .LANCHOR0,. + 0
.LC0:
    .word   10
    .word   5
    .word   3
    .word   2
    .word   12
    .word   69
    .word   35
    .word   69
    .word   70
    .word   11
    .section    .rodata.str1.8,"aMS",@progbits,1
    .align  3
.LC1:
    .string "%d"
    .ident  "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

maxnum.s:

maxnum.s

```
    .file   "maxnum.c"
    .option nopic
    .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .align  1
    .globl  maxNum
    .type   maxNum, @function
maxNum:
    li  a5,1
    ble a1,a5,.L1
    addiw   a1,a1,-1
    li  a5,0
.L3:
    addiw   a5,a5,1
    bne a1,a5,.L3
.L1:
    ret
    .size   maxNum, .-maxNum
    .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 main.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         0000005e  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  0000009e  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  0000009e  2**0
                  ALLOC
  3 .rodata       00000028  0000000000000000  0000000000000000  000000a0  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .rodata.str1.8 00000003  0000000000000000  0000000000000000  000000c8  2**3
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .comment      00000029  0000000000000000  0000000000000000  000000cb  2**0
                  CONTENTS, READONLY
  6 .riscv.attributes 00000035  0000000000000000  0000000000000000  000000f4  2**0
                  CONTENTS, READONLY
```

Таблица символов main.o:

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

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 main.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000000 l    d  .rodata        0000000000000000 .rodata
0000000000000000 l       .rodata        0000000000000000 .LANCHOR0
0000000000000000 l    d  .rodata.str1.8 0000000000000000 .rodata.str1.8
0000000000000000 l       .rodata.str1.8 0000000000000000 .LC1
000000000000003c l       .text  0000000000000000 .L2
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  000000000000005e main
0000000000000000         *UND*  0000000000000000 maxNum
0000000000000000         *UND*  0000000000000000 printf
```

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:   715d                    c.addi16sp      sp,-80
   2:   e486                    c.sdsp  ra,72(sp)
   4:   e0a2                    c.sdsp  s0,64(sp)
   6:   fc26                    c.sdsp  s1,56(sp)
   8:   f84a                    c.sdsp  s2,48(sp)
   a:   000007b7                lui     a5,0x0
                        a: R_RISCV_HI20 .LANCHOR0
                        a: R_RISCV_RELAX        *ABS*
   e:   00078793                addi    a5,a5,0 # 0 <main>
                        e: R_RISCV_LO12_I       .LANCHOR0
                        e: R_RISCV_RELAX        *ABS*
  12:   638c                    c.ld    a1,0(a5)
  14:   6790                    c.ld    a2,8(a5)
  16:   6b94                    c.ld    a3,16(a5)
  18:   6f98                    c.ld    a4,24(a5)
  1a:   739c                    c.ld    a5,32(a5)
  1c:   e42e                    c.sdsp  a1,8(sp)
  1e:   e832                    c.sdsp  a2,16(sp)
  20:   ec36                    c.sdsp  a3,24(sp)
  22:   f03a                    c.sdsp  a4,32(sp)
  24:   f43e                    c.sdsp  a5,40(sp)
  26:   45a9                    c.li    a1,10
  28:   0028                    c.addi4spn      a0,sp,8
  2a:   00000097                auipc   ra,0x0
                        2a: R_RISCV_CALL        maxNum
                        2a: R_RISCV_RELAX       *ABS*
  2e:   000080e7                jalr    ra,0(ra) # 2a <main+0x2a>
  32:   0020                    c.addi4spn      s0,sp,8
  34:   03010913                addi    s2,sp,48
  38:   000004b7                lui     s1,0x0
                        38: R_RISCV_HI20        .LC1
                        38: R_RISCV_RELAX       *ABS*

000000000000003c <.L2>:
  3c:   400c                    c.lw    a1,0(s0)
  3e:   00048513                addi    a0,s1,0 # 0 <main>
                        3e: R_RISCV_LO12_I      .LC1
                        3e: R_RISCV_RELAX       *ABS*
  42:   00000097                auipc   ra,0x0
                        42: R_RISCV_CALL        printf
                        42: R_RISCV_RELAX       *ABS*
  46:   000080e7                jalr    ra,0(ra) # 42 <.L2+0x6>
  4a:   0411                    c.addi  s0,4
  4c:   ff2418e3                bne     s0,s2,3c <.L2>
                        4c: R_RISCV_BRANCH      .L2
  50:   4501                    c.li    a0,0
  52:   60a6                    c.ldsp  ra,72(sp)
  54:   6406                    c.ldsp  s0,64(sp)
  56:   74e2                    c.ldsp  s1,56(sp)
  58:   7942                    c.ldsp  s2,48(sp)
  5a:   6161                    c.addi16sp      sp,80
  5c:   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  Algn
  0 .text         00000012  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000052  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000052  2**0
                  ALLOC
  3 .comment      00000029  0000000000000000  0000000000000000  00000052  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 00000035  0000000000000000  0000000000000000  0000007b  2**0
                  CONTENTS, READONLY
```

Таблица символов maxnum.o:

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

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 maxnum.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
0000000000000010 l       .text  0000000000000000 .L1
000000000000000a l       .text  0000000000000000 .L3
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  0000000000000012 maxNum
```

Таблица перемещений maxnum.o:

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


Disassembly of section .text:

0000000000000000 <maxNum>:
   0:   4785                    c.li    a5,1
   2:   00b7d763                bge     a5,a1,10 <.L1>
                        2: R_RISCV_BRANCH       .L1
   6:   35fd                    c.addiw a1,-1
   8:   4781                    c.li    a5,0

000000000000000a <.L3>:
   a:   2785                    c.addiw a5,1
   c:   fef59fe3                bne     a1,a5,a <.L3>
                        c: R_RISCV_BRANCH       .L3

0000000000000010 <.L1>:
  10:   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

```
0000000000010158 <main>:
   10158:   715d                addi16sp  sp,-80
   1015a:   e486                sdsp   ra,72(sp)
   1015c:   e0a2                sdsp   s0,64(sp)
   1015e:   fc26                sdsp   s1,56(sp)
   10160:   f84a                sdsp   s2,48(sp)
   10162:   67f1                lui a5,0x1c
   10164:   0e078793            addi  a5,a5,224 # 1c0e0 <__clzdi2+0x3e>
   10168:   638c                ld  a1,0(a5)
   1016a:   6790                ld  a2,8(a5)
   1016c:   6b94                ld  a3,16(a5)
   1016e:   6f98                ld  a4,24(a5)
   10170:   739c                ld  a5,32(a5)
   10172:   e42e                sdsp   a1,8(sp)
   10174:   e832                sdsp   a2,16(sp)
   10176:   ec36                sdsp   a3,24(sp)
   10178:   f03a                sdsp   a4,32(sp)
   1017a:   f43e                sdsp   a5,40(sp)
   1017c:   45a9                li  a1,10
   1017e:   0028                addi4spn  a0,sp,8
   10180:   02a000ef            jal   ra,101aa <maxNum>
   10184:   0020                addi4spn  s0,sp,8
   10186:   03010913            addi  s2,sp,48
   1018a:   64f1                lui s1,0x1c
   1018c:   400c                lw  a1,0(s0)
   1018e:   10848513            addi  a0,s1,264 # 1c108 <__clzdi2+0x66>
   10192:   1c6000ef            jal   ra,10358 <printf>
   10196:   0411                addi   s0,4
   10198:   ff241ae3            bne   s0,s2,1018c <main+0x34>
   1019c:   4501                li  a0,0
   1019e:   60a6                ldsp   ra,72(sp)
   101a0:   6406                ldsp   s0,64(sp)
   101a2:   74e2                ldsp   s1,56(sp)
   101a4:   7942                ldsp   s2,48(sp)
   101a6:   6161                addi16sp  sp,80
   101a8:   8082                jr  ra

00000000000101aa <maxNum>:
   101aa:   4785                li  a5,1
   101ac:   00b7d763            bge   a5,a1,101ba <maxNum+0x10>
   101b0:   35fd                addiw  a1,-1
   101b2:   4781                li  a5,0
   101b4:   2785                addiw  a5,1
   101b6:   fef59fe3            bne   a1,a5,101b4 <maxNum+0xa>
   101ba:   8082                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:

```
000000000001e178 g       .sbss  0000000000000000 __bss_start
00000000000102e8 g     F .text  00000000000000aa memset
0000000000010158 g     F .text  0000000000000052 main
000000000001e178 g     O .sbss  0000000000000008 __malloc_max_total_mem
000000000001aa08 g     F .text  000000000000000c __swbuf
00000000000161a4 g     F .text  0000000000000008 __sclose
0000000000018ea6 g     F .text  000000000000000a fclose
0000000000014d56 g     F .text  0000000000000660 _malloc_r
000000000001aab8 g     F .text  0000000000000024 __ascii_wctomb
0000000000012a60 g     F .text  000000000000008a _fwalk
0000000000019438 g     F .text  000000000000000a _mbtowc_r
000000000001271e g     F .text  00000000000000d8 _malloc_trim_r
0000000000019b9c g     F .text  00000000000000ea strcmp
0000000000018c54 g     F .text  0000000000000010 vfiprintf
000000000001af06 g     F .text  0000000000000606 .hidden __multf3
0000000000016086 g     F .text  000000000000004c sprintf
000000000001cd50 g     O .rodata        0000000000000100 .hidden __clz_tab
000000000001e190 g     O .sbss  0000000000000008 _PathLocale
000000000001021a g     F .text  000000000000000c atexit
00000000000018cf8 g     F .text  0000000000000040 _write_r
00000000000193ec g     F .text  000000000000000c setlocale
000000000001e160 g     O .sdata 0000000000000008 _impure_ptr
00000000000122b8 g     F .text  0000000000000196 __sflush_r
000000000001adaa g     F .text  00000000000000ae .hidden __gttf2
0000000000019de4 g     F .text  0000000000000b26 _svfiprintf_r
0000000000019442 g     F .text  0000000000000040 __ascii_mbtowc
000000000001b50c g     F .text  000000000000081a .hidden __subtf3
0000000000015c08 g     F .text  0000000000000060 __ulp
000000000001270e g     F .text  0000000000000010 __fp_unlock_all
0000000000014c12 g     F .text  0000000000000006 localeconv
0000000000014c18 g     F .text  0000000000000082 __swhatbuf_r
000000000001d020 g       .data  0000000000000000 __DATA_BEGIN__
000000000001ac7a g     F .text  0000000000000032 _write
000000000001e178 g       .sdata 0000000000000000 _edata
000000000001e208 g       .bss   0000000000000000 _end
00000000000018eb0 g     F .text  00000000000000c6 __fputwc
000000000001610e g     F .text  0000000000000054 __swrite
000000000001e170 g     O .sdata 0000000000000008 __malloc_trim_threshold
0000000000010226 g     F .text  0000000000000020 exit
0000000000017fe0 g     F .text  0000000000000c74 _vfiprintf_r
0000000000012aea g     F .text  0000000000000092 _fwalk_reent
0000000000015ac0 g     F .text  0000000000000148 __mdiff
00000000000126f8 g     F .text  0000000000000002 __sfp_lock_release
0000000000013d08 g     F .text  0000000000000ec8 _ldtoa_r
000000000001cc10 g     O .rodata        0000000000000101 _ctype_
000000000001abde g     F .text  0000000000000032 _read
000000000001ab12 g     F .text  000000000000002c _exit
0000000000014c9a g     F .text  00000000000000bc __smakebuf_r
0000000000016250 g     F .text  0000000000000098 strlen
0000000000017fd2 g     F .text  000000000000000e _sprint_r
00000000000101aa g     F .text  0000000000000070 maxNum
000000000001aaae 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 просмотреть таблицу символов, то видно, что содержимое файла точно такое же:

```
00000000000102e8 g     F .text  00000000000000aa memset
0000000000010158 g     F .text  0000000000000052 main
000000000001e178 g     O .sbss  0000000000000008 __malloc_max_total_mem
000000000001aa08 g     F .text  000000000000000c __swbuf
00000000000161a4 g     F .text  0000000000000008 __sclose
0000000000018ea6 g     F .text  000000000000000a fclose
0000000000014d56 g     F .text  0000000000000660 _malloc_r
000000000001aab8 g     F .text  0000000000000024 __ascii_wctomb
0000000000012a60 g     F .text  000000000000008a _fwalk
0000000000019438 g     F .text  000000000000000a _mbtowc_r
000000000001271e g     F .text  00000000000000d8 _malloc_trim_r
0000000000019b9c g     F .text  00000000000000ea strcmp
0000000000018c54 g     F .text  0000000000000010 vfiprintf
000000000001af06 g     F .text  0000000000000606 .hidden __multf3
0000000000016086 g     F .text  000000000000004c sprintf
000000000001cd50 g     O .rodata        0000000000000100 .hidden __clz_tab
000000000001e190 g     O .sbss  0000000000000008 _PathLocale
000000000001021a g     F .text  000000000000000c atexit
0000000000018cf8 g     F .text  0000000000000040 _write_r
00000000000193ec g     F .text  000000000000000c setlocale
000000000001e160 g     O .sdata 0000000000000008 _impure_ptr
00000000000122b8 g     F .text  0000000000000196 __sflush_r
000000000001adaa g     F .text  00000000000000ae .hidden __gttf2
0000000000019de4 g     F .text  0000000000000b26 _svfiprintf_r
0000000000019442 g     F .text  0000000000000040 __ascii_mbtowc
000000000001b50c g     F .text  000000000000081a .hidden __subtf3
0000000000015c08 g     F .text  0000000000000060 __ulp
000000000001270e g     F .text  0000000000000010 __fp_unlock_all
0000000000014c12 g     F .text  0000000000000006 localeconv
0000000000014c18 g     F .text  0000000000000082 __swhatbuf_r
000000000001d020 g       .data  0000000000000000 __DATA_BEGIN__
000000000001ac7a g     F .text  0000000000000032 _write
000000000001e178 g       .sdata 0000000000000000 _edata
000000000001e208 g       .bss   0000000000000000 _end
0000000000018eb0 g     F .text  00000000000000c6 __fputwc
000000000001610e g     F .text  0000000000000054 __swrite
000000000001e170 g     O .sdata 0000000000000008 __malloc_trim_threshold
0000000000010226 g     F .text  0000000000000020 exit
0000000000017fe0 g     F .text  0000000000000c74 _vfiprintf_r
0000000000012aea g     F .text  0000000000000092 _fwalk_reent
0000000000015ac0 g     F .text  0000000000000148 __mdiff
00000000000126f8 g     F .text  0000000000000002 __sfp_lock_release
0000000000013d08 g     F .text  0000000000000ec8 _ldtoa_r
000000000001cc10 g     O .rodata        0000000000000101 _ctype_
000000000001abde g     F .text  0000000000000032 _read
000000000001ab12 g     F .text  000000000000002c _exit
0000000000014c9a g     F .text  00000000000000bc __smakebuf_r
0000000000016250 g     F .text  0000000000000098 strlen
0000000000017fd2 g     F .text  000000000000000e __sprint_r
00000000000101aa g     F .text  0000000000000070 maxNum
000000000001aaae g     F .text  000000000000000a _wctomb_r
000000000001c100 g     F .text  0000000000000030 .hidden __clzdi2
```

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.