Паламарчук Андрей ИУ7-23Б

# **Этапы получения исполняемого файла**

## **Программа**

#include <stdio.h>  
#include <stdlib.h>  
  
#define HANDLER\_OWERFLOW 100  
#define ERROR\_INPUT\_ARR 1  
#define N 10  
typedef int arr\_t[N];  
  
  
int input\_arr(int \*a, size\_t \*n)  
{  
    int count = 0;  
    int tmp;  
  
    while ((count < N) && (scanf("%d", &a[\*n]) == 1))  
    {  
        \*n = (\*n) + 1;  
        count = count + 1;  
    }  
    if ((count == N) && (scanf("%d", &tmp) == 1))  
        return HANDLER\_OWERFLOW;  
    //Пустой массив  
    if (\*n == 0)  
        return ERROR\_INPUT\_ARR;  
  
    return EXIT\_SUCCESS;  
}  
  
  
int bubble\_sort(int \*a, size\_t n)  
{  
    for (size\_t i = 0; i < n - 1; i++)  
    {  
        for (size\_t j = 0; j < n - i - 1; j++)  
        {  
            if (a[j] > a[j + 1])  
            {  
                int tmp = a[j];  
                a[j] = a[j + 1];  
                a[j + 1] = tmp;  
            }  
        }  
    }  
    return 0;  
}  
  
  
void print\_arr(int \*dst, size\_t dlen)  
{  
    for (size\_t i = 0; i < dlen; i++)  
        printf("%d ", dst[i]);  
}  
  
  
int main(void)  
{  
    arr\_t a;  
    size\_t n = 0;  
  
    printf("Введите элементы массива: ");  
  
    int code\_rn = input\_arr(a, &n);  
  
    if (code\_rn == ERROR\_INPUT\_ARR)  
    {  
        printf("Некорректный ввод массива\n");  
        return ERROR\_INPUT\_ARR;  
    }  
  
    bubble\_sort(a, n);  
  
    printf("Массив упорядоченных элементов: ");  
    print\_arr(a, n);  
    printf("\n");  
  
    return code\_rn;  
}

## **Этапы получения исполняемого файла**

Четыре этапа получения исполняемого файла:

### **Обработка препроцессором**

* + Удаление комментариев (замена их на пробельные символы)
  + Вставка файлов (директива include)
  + Текстовые замены (директива define)
  + Условная компиляция (директива if)

Данный этап можно выполнить командой cpp:

$ cpp main.c > main.i

Можно заметить замену директивы define:

# 7 "main.c"  
typedef int arr\_t[10];  
  
  
int input\_arr(int \*a, size\_t \*n)  
{  
    int count = 0;  
    int tmp;  
  
    while ((count < 10) && (scanf("%d", &a[\*n]) == 1))  
    {  
        \*n = (\*n) + 1;  
        count = count + 1;  
    }  
    if ((count == 10) && (scanf("%d", &tmp) == 1))  
        return 100;  
  
    if (\*n == 0)  
        return 1;  
  
    return  
# 26 "main.c" 3 4  
          0  
# 26 "main.c"  
                      ;  
}  
  
  
int bubble\_sort(int \*a, size\_t n)  
{  
    for (size\_t i = 0; i < n - 1; i++)  
    {  
        for (size\_t j = 0; j < n - i - 1; j++)  
        {  
            if (a[j] > a[j + 1])  
            {  
                int tmp = a[j];  
                a[j] = a[j + 1];  
                a[j + 1] = tmp;  
            }  
        }  
    }  
    return 0;  
}  
  
  
void print\_arr(int \*dst, size\_t dlen)  
{  
    for (size\_t i = 0; i < dlen; i++)  
        printf("%d ", dst[i]);  
}  
  
  
int main(void)  
{  
    arr\_t a;  
    size\_t n = 0;  
  
    printf("Введите элементы массива: ");  
  
    int code\_rn = input\_arr(a, &n);  
  
    if (code\_rn == 1)  
    {  
        printf("Некорректный ввод массива\n");  
        return 1;  
    }  
  
    bubble\_sort(a, n);  
  
    printf("Массив упорядоченных элементов: ");  
    print\_arr(a, n);  
    printf("\n");  
  
    return code\_rn;  
}

1. **Компиляция**  
   Код после препроцессирования, «переводится» с языка Си на язык Ассемблера. Это даёт несколько преимуществ:
   * Упрощение реализации и отладки транслятора
   * Повышение переносимости с одной платформы на другую

$ c99 -S -fverbose-asm main.i

Флагом «-S» мы запускаем только процесс компиляции, «-fverbose-asm» позволяет добавить некоторые «полезные» комментарии.

1. **Ассемблирование**  
   На данном этапе код, что в текущий момент находится на языке ассемблера, переводится в машинный при помощи ретранслятора.

$ as main.s -o main.o

На выходе получаем объектный файл в двоичном виде.

### **Компоновка**

* объединение нескольких объектных файлов в единый исполняемый файл;
* выполнение связывания переменных и функций, которые требуются очередному объектному файлу, но находятся где-то в другом месте
* добавление специального кода, который подготавливает окружение для вызова функции main, а после ее завершения выполняет обратные действия.

Данный этап выполняется командой ld:

$ ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 /usr/lib/x86\_64\

x86\_64-redhat-linux/crt1.o /usr/lib/ x86\_64-redhat-linux /crti.o -lc\

/usr/lib/x86\_64-linux-gnu/crtn.o main.o -o main.exe -lm

## **Почему gcc и clang называются «программами-драйверами»**

**gcc** и **clang** называются «программами-драйверами» из-за того, что они запускают другие программы с определёнными ключами.

## **Этапы и их выполнение для получения исполняемого файла компилятором gcc**

Для начала с помощью флага «--help» узнаем, за что отвечают ключи «-v» и «-save-temps»:

* Ключ «-v» позволяет нам узнать, какие программы запускает компилятор **gcc** для сборки исполняемого файла
* Ключ «-save-temps» не очищает промежуточные файлы, которые создаются во время работы других вызываемых программ

Используя эти ключи, мы можем заметить, что, в отличие от наших 4 этапов для получения исполняемого файла в gcc их всего 3:

### **Обработка препроцессором**

На данном этапе **gcc** делает препроцессирование:

/usr/libexec/gcc/x86\_64-redhat-linux/12/cc1 -E -quiet -v main.c\  
-mtune=generic -march=x86-64 -std=c99 -fpch-preprocess -o main.i

1. **Компиляция**На данном этапе **gcc** делает компиляцию в ассемблерный код на одном этапе с помощью следующей команды:

/usr/libexec/gcc/x86\_64-redhat-linux/12/cc1 -fpreprocessed main.i\  
-quiet -dumpbase main.c -dumpbase-ext .c -mtune=generic\

-march=x86-64 -std=c99 -version -o main.s

1. **Ассемблирование**На данном этапе **gcc** делает перевод ассемблерный код в машинный с помощью следующей команды:

as -v --64 -o main.o main.s

1. **Компоновка**На данном этапе **gcc** связывает объектный файл, созданный этапом выше, с другими объектными файлами, создавая исполняемый:

/usr/libexec/gcc/x86\_64-redhat-linux/12/collect2 -plugin  
/usr/libexec/gcc/x86\_64-redhat-linux/12/liblto\_plugin.so  
-plugin-opt=/usr/libexec/gcc/x86\_64-redhat-linux/12/lto-wrapper  
-plugin-opt=-fresolution=main.res -plugin-opt=-pass-through=-lgcc  
-plugin-opt=-pass-through=-lgcc\_s -plugin-opt=-pass-through=-lc  
-plugin-opt=-pass-through=-lgcc -plugin-opt=-pass-through=-lgcc\_s  
--build-id --no-add-needed --eh-frame-hdr --hash-style=gnu -m  
elf\_x86\_64 -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o main.exe  
/usr/lib/gcc/x86\_64-redhat-linux/12/../../../../lib64/crt1.o  
/usr/lib/gcc/x86\_64-redhat-linux/12/../../../../lib64/crti.o  
/usr/lib/gcc/x86\_64-redhat-linux/12/crtbegin.o  
-L/usr/lib/gcc/x86\_64-redhat-linux/12  
-L/usr/lib/gcc/x86\_64-redhat-linux/12/../../../../lib64  
-L/lib/../lib64 -L/usr/lib/../lib64  
-L/usr/lib/gcc/x86\_64-redhat-linux/12/../../.. main.o -lgcc  
--push-state --as-needed -lgcc\_s --pop-state -lc -lgcc --push-state  
--as-needed -lgcc\_s --pop-state  
/usr/lib/gcc/x86\_64-redhat-linux/12/crtend.o  
/usr/lib/gcc/x86\_64-redhat-linux/12/../../../../lib64/crtn.o

Сборка исполняемого файла вручную и при помощи компилятора gcc отличаются в следующем, на этапе компоновки подключается библиотека «libc» при помощи объектных файлов, из-за чего объектные файлы нашего кода (созданные при помощи компилятора и вручную) отличаются. Это можно проверить с помощью команды diff.

## **Этапы и их выполнение для получения исполняемого файла компилятором clang**

При помощи уже известных нам ключей мы можем заметить, что clang получает исполняемый файл при помощи всего двух этапов:

1. **Получение объектного файла «.o»**На данном этапе **clang** получает объектный файл:

"/usr/bin/clang-15" -cc1as –triple\

x86\_64-redhat-linux-gnu -filetype\  
obj -main-file-name main.c -target-cpu x86-64\  
-fdebug-compilation-dir=/home/Natalia/practic\_PTP/Task\_2\  
-dwarf-version=4 -mrelocation-model static -mrelax-all\  
--mrelax-relocations -o main.o main.s

1. **Компоновка**На данном этапе **clang** получает исполняемый файл с помощью команды ld:

## "/usr/bin/ld" --hash-style=gnu --build-id --eh-frame-hdr\

## –m elf\_x86\_64\ -dynamic-linker /lib64/ld-linux-x86-64.so.2 -o main.exe\ /usr/bin/../lib/gcc\

## /x86\_64-redhat-linux/12/../../../../lib64/crt1.o\ /usr/bin/../lib/gcc\

## /x86\_64-redhat-linux/12/../../../../lib64/crti.o\ /usr/bin/../lib/gcc/x86\_64-redhat-linux/12/crtbegin.o\ -L/usr/bin/../lib/gcc/x86\_64-redhat-linux/12\ -L/usr/bin/../lib/gcc/x86\_64-redhat-linux/12/../../../../lib64\ -L/lib/../lib64 -L/usr/lib/../lib64 -L/lib -L/usr\

## /lib main.o –lgcc --as-needed -lgcc\_s --no-as-needed\

## -lc -lgcc --as-needed\

## -lgcc\_s --no-as-needed /usr/bin/../lib/gcc\

## /x86\_64-redhat-linux/12/crtend.o /usr/bin/../lib/gcc\

## /x86\_64-redhat-linux/12/../../../../lib64/crtn.o

## **Ассемблерный листинг в gcc**

С помощью команды с помощью флага «--help» узнаем, какие ключи нужны для передачи параметров компилятору с языка ассемблера:

* -Xassembler

$ gcc -Xassembler -a=main\_asm.s main.c –lm

* -Wa

$ gcc -Wa,-a=main\_asm.s main.c -lm

Ассемблерный листинг (Xassembler):

GAS LISTING /tmp/cc3Hheud.s             page 1  
  
  
   1                      .file    "main.c"  
   2                      .text  
   3                      .section    .rodata  
   4                  .LC0:  
   5 0000 256400           .string    "%d"  
   6                      .text  
   7                      .globl    input\_arr  
   8                      .type    input\_arr, @function  
   9                  input\_arr:  
  10                  .LFB6:  
  11                      .cfi\_startproc  
  12 0000 55               pushq    %rbp  
  13                      .cfi\_def\_cfa\_offset 16  
  14                      .cfi\_offset 6, -16  
  15 0001 4889E5           movq    %rsp, %rbp  
  16                      .cfi\_def\_cfa\_register 6  
  17 0004 4883EC20         subq    $32, %rsp  
  18 0008 48897DE8         movq    %rdi, -24(%rbp)  
  19 000c 488975E0         movq    %rsi, -32(%rbp)  
  20 0010 C745FC00         movl    $0, -4(%rbp)  
  20      000000  
  21 0017 EB16             jmp    .L2  
  22                  .L4:  
  23 0019 488B45E0         movq    -32(%rbp), %rax  
  24 001d 488B00           movq    (%rax), %rax  
  25 0020 488D5001         leaq    1(%rax), %rdx  
  26 0024 488B45E0         movq    -32(%rbp), %rax  
  27 0028 488910           movq    %rdx, (%rax)  
  28 002b 8345FC01         addl    $1, -4(%rbp)  
  29                  .L2:  
  30 002f 837DFC09         cmpl    $9, -4(%rbp)  
  31 0033 7F2D             jg    .L3  
  32 0035 488B45E0         movq    -32(%rbp), %rax  
  33 0039 488B00           movq    (%rax), %rax  
  34 003c 488D1485         leaq    0(,%rax,4), %rdx  
  34      00000000  
  35 0044 488B45E8         movq    -24(%rbp), %rax  
  36 0048 4801D0           addq    %rdx, %rax  
  37 004b 4889C6           movq    %rax, %rsi  
  38 004e BF000000         movl    $.LC0, %edi  
  38      00  
  39 0053 B8000000         movl    $0, %eax  
  39      00  
  40 0058 E8000000         call    \_\_isoc99\_scanf  
  40      00  
  41 005d 83F801           cmpl    $1, %eax  
  42 0060 74B7             je    .L4  
  43                  .L3:  
  44 0062 837DFC0A         cmpl    $10, -4(%rbp)  
  45 0066 7522             jne    .L5  
  46 0068 488D45F8         leaq    -8(%rbp), %rax  
  47 006c 4889C6           movq    %rax, %rsi  
  48 006f BF000000         movl    $.LC0, %edi  
  48      00  
  49 0074 B8000000         movl    $0, %eax  
  49      00  
  50 0079 E8000000         call    \_\_isoc99\_scanf  
  
GAS LISTING /tmp/cc3Hheud.s             page 2  
  
  
  50      00  
  51 007e 83F801           cmpl    $1, %eax  
  52 0081 7507             jne    .L5  
  53 0083 B8640000         movl    $100, %eax  
  53      00  
  54 0088 EB18             jmp    .L8  
  55                  .L5:  
  56 008a 488B45E0         movq    -32(%rbp), %rax  
  57 008e 488B00           movq    (%rax), %rax  
  58 0091 4885C0           testq    %rax, %rax  
  59 0094 7507             jne    .L7  
  60 0096 B8010000         movl    $1, %eax  
  60      00  
  61 009b EB05             jmp    .L8  
  62                  .L7:  
  63 009d B8000000         movl    $0, %eax  
  63      00  
  64                  .L8:  
  65 00a2 C9               leave  
  66                      .cfi\_def\_cfa 7, 8  
  67 00a3 C3               ret  
  68                      .cfi\_endproc  
  69                  .LFE6:  
  70                      .size    input\_arr, .-input\_arr  
  71                      .globl    bubble\_sort  
  72                      .type    bubble\_sort, @function  
  73                  bubble\_sort:  
  74                  .LFB7:  
  75                      .cfi\_startproc  
  76 00a4 55               pushq    %rbp  
  77                      .cfi\_def\_cfa\_offset 16  
  78                      .cfi\_offset 6, -16  
  79 00a5 4889E5           movq    %rsp, %rbp  
  80                      .cfi\_def\_cfa\_register 6  
  81 00a8 48897DD8         movq    %rdi, -40(%rbp)  
  82 00ac 488975D0         movq    %rsi, -48(%rbp)  
  83 00b0 48C745F8         movq    $0, -8(%rbp)  
  83      00000000  
  84 00b8 E9C10000         jmp    .L10  
  84      00  
  85                  .L14:  
  86 00bd 48C745F0         movq    $0, -16(%rbp)  
  86      00000000  
  87 00c5 E9990000         jmp    .L11  
  87      00  
  88                  .L13:  
  89 00ca 488B45F0         movq    -16(%rbp), %rax  
  90 00ce 488D1485         leaq    0(,%rax,4), %rdx  
  90      00000000  
  91 00d6 488B45D8         movq    -40(%rbp), %rax  
  92 00da 4801D0           addq    %rdx, %rax  
  93 00dd 8B10             movl    (%rax), %edx  
  94 00df 488B45F0         movq    -16(%rbp), %rax  
  95 00e3 4883C001         addq    $1, %rax  
  96 00e7 488D0C85         leaq    0(,%rax,4), %rcx  
  96      00000000  
  97 00ef 488B45D8         movq    -40(%rbp), %rax  
  
GAS LISTING /tmp/cc3Hheud.s             page 3  
  
  
  98 00f3 4801C8           addq    %rcx, %rax  
  99 00f6 8B00             movl    (%rax), %eax  
 100 00f8 39C2             cmpl    %eax, %edx  
 101 00fa 7E62             jle    .L12  
 102 00fc 488B45F0         movq    -16(%rbp), %rax  
 103 0100 488D1485         leaq    0(,%rax,4), %rdx  
 103      00000000  
 104 0108 488B45D8         movq    -40(%rbp), %rax  
 105 010c 4801D0           addq    %rdx, %rax  
 106 010f 8B00             movl    (%rax), %eax  
 107 0111 8945EC           movl    %eax, -20(%rbp)  
 108 0114 488B45F0         movq    -16(%rbp), %rax  
 109 0118 4883C001         addq    $1, %rax  
 110 011c 488D1485         leaq    0(,%rax,4), %rdx  
 110      00000000  
 111 0124 488B45D8         movq    -40(%rbp), %rax  
 112 0128 4801D0           addq    %rdx, %rax  
 113 012b 488B55F0         movq    -16(%rbp), %rdx  
 114 012f 488D0C95         leaq    0(,%rdx,4), %rcx  
 114      00000000  
 115 0137 488B55D8         movq    -40(%rbp), %rdx  
 116 013b 4801CA           addq    %rcx, %rdx  
 117 013e 8B00             movl    (%rax), %eax  
 118 0140 8902             movl    %eax, (%rdx)  
 119 0142 488B45F0         movq    -16(%rbp), %rax  
 120 0146 4883C001         addq    $1, %rax  
 121 014a 488D1485         leaq    0(,%rax,4), %rdx  
 121      00000000  
 122 0152 488B45D8         movq    -40(%rbp), %rax  
 123 0156 4801C2           addq    %rax, %rdx  
 124 0159 8B45EC           movl    -20(%rbp), %eax  
 125 015c 8902             movl    %eax, (%rdx)  
 126                  .L12:  
 127 015e 488345F0         addq    $1, -16(%rbp)  
 127      01  
 128                  .L11:  
 129 0163 488B45D0         movq    -48(%rbp), %rax  
 130 0167 482B45F8         subq    -8(%rbp), %rax  
 131 016b 4883E801         subq    $1, %rax  
 132 016f 483945F0         cmpq    %rax, -16(%rbp)  
 133 0173 0F8251FF         jb    .L13  
 133      FFFF  
 134 0179 488345F8         addq    $1, -8(%rbp)  
 134      01  
 135                  .L10:  
 136 017e 488B45D0         movq    -48(%rbp), %rax  
 137 0182 4883E801         subq    $1, %rax  
 138 0186 483945F8         cmpq    %rax, -8(%rbp)  
 139 018a 0F822DFF         jb    .L14  
 139      FFFF  
 140 0190 B8000000         movl    $0, %eax  
 140      00  
 141 0195 5D               popq    %rbp  
 142                      .cfi\_def\_cfa 7, 8  
 143 0196 C3               ret  
 144                      .cfi\_endproc  
 145                  .LFE7:  
  
GAS LISTING /tmp/cc3Hheud.s             page 4  
  
  
 146                      .size    bubble\_sort, .-bubble\_sort  
 147                      .section    .rodata  
 148                  .LC1:  
 149 0003 25642000         .string    "%d "  
 150                      .text  
 151                      .globl    print\_arr  
 152                      .type    print\_arr, @function  
 153                  print\_arr:  
 154                  .LFB8:  
 155                      .cfi\_startproc  
 156 0197 55               pushq    %rbp  
 157                      .cfi\_def\_cfa\_offset 16  
 158                      .cfi\_offset 6, -16  
 159 0198 4889E5           movq    %rsp, %rbp  
 160                      .cfi\_def\_cfa\_register 6  
 161 019b 4883EC20         subq    $32, %rsp  
 162 019f 48897DE8         movq    %rdi, -24(%rbp)  
 163 01a3 488975E0         movq    %rsi, -32(%rbp)  
 164 01a7 48C745F8         movq    $0, -8(%rbp)  
 164      00000000  
 165 01af EB2B             jmp    .L17  
 166                  .L18:  
 167 01b1 488B45F8         movq    -8(%rbp), %rax  
 168 01b5 488D1485         leaq    0(,%rax,4), %rdx  
 168      00000000  
 169 01bd 488B45E8         movq    -24(%rbp), %rax  
 170 01c1 4801D0           addq    %rdx, %rax  
 171 01c4 8B00             movl    (%rax), %eax  
 172 01c6 89C6             movl    %eax, %esi  
 173 01c8 BF000000         movl    $.LC1, %edi  
 173      00  
 174 01cd B8000000         movl    $0, %eax  
 174      00  
 175 01d2 E8000000         call    printf  
 175      00  
 176 01d7 488345F8         addq    $1, -8(%rbp)  
 176      01  
 177                  .L17:  
 178 01dc 488B45F8         movq    -8(%rbp), %rax  
 179 01e0 483B45E0         cmpq    -32(%rbp), %rax  
 180 01e4 72CB             jb    .L18  
 181 01e6 90               nop  
 182 01e7 90               nop  
 183 01e8 C9               leave  
 184                      .cfi\_def\_cfa 7, 8  
 185 01e9 C3               ret  
 186                      .cfi\_endproc  
 187                  .LFE8:  
 188                      .size    print\_arr, .-print\_arr  
 189                      .section    .rodata  
 190 0007 00               .align 8  
 191                  .LC2:  
 192 0008 D092D0B2         .string  
"\320\222\320\262\320\265\320\264\320\270\321\202\320\265  
\321\215\320\273\320\265\320\274  
 192      D0B5D0B4  
 192      D0B8D182  
 192      D0B520D1  
 192      8DD0BBD0  
  
GAS LISTING /tmp/cc3Hheud.s             page 5  
  
  
 193 0039 00000000         .align 8  
 193      000000  
 194                  .LC3:  
 195 0040 D09DD0B5         .string  
"\320\235\320\265\320\272\320\276\321\200\321\200\320\265\320\272\321\202\320\275\321\213\  
 195      D0BAD0BE  
 195      D180D180  
 195      D0B5D0BA  
 195      D182D0BD  
 196 0071 00000000         .align 8  
 196      000000  
 197                  .LC4:  
 198 0078 D09CD0B0         .string  
"\320\234\320\260\321\201\321\201\320\270\320\262  
\321\203\320\277\320\276\321\200\321\217  
 198      D181D181  
 198      D0B8D0B2  
 198      20D183D0  
 198      BFD0BED1  
 199                      .text  
 200                      .globl    main  
 201                      .type    main, @function  
 202                  main:  
 203                  .LFB9:  
 204                      .cfi\_startproc  
 205 01ea 55               pushq    %rbp  
 206                      .cfi\_def\_cfa\_offset 16  
 207                      .cfi\_offset 6, -16  
 208 01eb 4889E5           movq    %rsp, %rbp  
 209                      .cfi\_def\_cfa\_register 6  
 210 01ee 4883EC40         subq    $64, %rsp  
 211 01f2 48C745C8         movq    $0, -56(%rbp)  
 211      00000000  
 212 01fa BF000000         movl    $.LC2, %edi  
 212      00  
 213 01ff B8000000         movl    $0, %eax  
 213      00  
 214 0204 E8000000         call    printf  
 214      00  
 215 0209 488D55C8         leaq    -56(%rbp), %rdx  
 216 020d 488D45D0         leaq    -48(%rbp), %rax  
 217 0211 4889D6           movq    %rdx, %rsi  
 218 0214 4889C7           movq    %rax, %rdi  
 219 0217 E8000000         call    input\_arr  
 219      00  
 220 021c 8945FC           movl    %eax, -4(%rbp)  
 221 021f 837DFC01         cmpl    $1, -4(%rbp)  
 222 0223 7511             jne    .L20  
 223 0225 BF000000         movl    $.LC3, %edi  
 223      00  
 224 022a E8000000         call    puts  
 224      00  
 225 022f B8010000         movl    $1, %eax  
 225      00  
 226 0234 EB42             jmp    .L22  
 227                  .L20:  
 228 0236 488B55C8         movq    -56(%rbp), %rdx  
 229 023a 488D45D0         leaq    -48(%rbp), %rax  
 230 023e 4889D6           movq    %rdx, %rsi  
 231 0241 4889C7           movq    %rax, %rdi  
  
GAS LISTING /tmp/cc3Hheud.s             page 6  
  
  
 232 0244 E8000000         call    bubble\_sort  
 232      00  
 233 0249 BF000000         movl    $.LC4, %edi  
 233      00  
 234 024e B8000000         movl    $0, %eax  
 234      00  
 235 0253 E8000000         call    printf  
 235      00  
 236 0258 488B55C8         movq    -56(%rbp), %rdx  
 237 025c 488D45D0         leaq    -48(%rbp), %rax  
 238 0260 4889D6           movq    %rdx, %rsi  
 239 0263 4889C7           movq    %rax, %rdi  
 240 0266 E8000000         call    print\_arr  
 240      00  
 241 026b BF0A0000         movl    $10, %edi  
 241      00  
 242 0270 E8000000         call    putchar  
 242      00  
 243 0275 8B45FC           movl    -4(%rbp), %eax  
 244                  .L22:  
 245 0278 C9               leave  
 246                      .cfi\_def\_cfa 7, 8  
 247 0279 C3               ret  
 248                      .cfi\_endproc  
 249                  .LFE9:  
 250                      .size    main, .-main  
 251                      .ident    "GCC: (GNU) 12.2.1 20221121 (Red  
Hat 12.2.1-4)"  
 252                      .section    .note.GNU-stack,"",@progbits  
  
GAS LISTING /tmp/cc3Hheud.s             page 7  
  
  
DEFINED SYMBOLS  
                            \*ABS\*:0000000000000000 main.c  
     /tmp/cc3Hheud.s:9      .text:0000000000000000 input\_arr  
     /tmp/cc3Hheud.s:73     .text:00000000000000a4 bubble\_sort  
     /tmp/cc3Hheud.s:153    .text:0000000000000197 print\_arr  
     /tmp/cc3Hheud.s:202    .text:00000000000001ea main  
  
UNDEFINED SYMBOLS  
\_\_isoc99\_scanf  
printf  
puts  
putchar

1. **Создание map-файла в gcc**

С помощью команды с помощью флага «--help» узнаем, какие ключи нужны для передачи параметров компоновщику:

* -Xlinker

$ gcc -Xlinker -Map=main.map main.c –lm

* -Wl

$ gcc -Wl,-Map=main.map main.c –lm

Создание map-файла(Xlinker):

Объединение программных свойств

…

Отброшенные входные разделы

…

Настройки памяти  
  
Имя           Происхождение Длина         Атрибуты  
\*default\*        0x0000000000000000 0xffffffffffffffff  
  
Сценарий компоновщика и карта памяти

…

/DISCARD/  
 \*(.note.GNU-stack)  
 \*(.gnu\_debuglink)  
 \*(.gnu.lto\_\*)  
OUTPUT(a.out elf64-x86-64)

Файл a.out содержит информацию о дополнительных библиотеках, карту памяти и сценарий компоновщика.

## **Дизассемблирование полученного объектного файла**

Дизассемблировать объектный файл можно командой objdump -d следующим образом:

$ objdump -d main.o > main\_disasm.s

Дизассемблированный объектный файл в main\_disasm.s:

main.o:     формат файла elf64-x86-64  
  
  
Дизассемблирование раздела .text:  
  
0000000000000000 <input\_arr>:  
   0:    55                       push   %rbp  
   1:    48 89 e5                 mov    %rsp,%rbp  
   4:    48 83 ec 20              sub    $0x20,%rsp  
   8:    48 89 7d e8              mov    %rdi,-0x18(%rbp)  
   c:    48 89 75 e0              mov    %rsi,-0x20(%rbp)  
  10:    c7 45 fc 00 00 00 00     movl   $0x0,-0x4(%rbp)  
  17:    eb 16                    jmp    2f <input\_arr+0x2f>  
  19:    48 8b 45 e0              mov    -0x20(%rbp),%rax  
  1d:    48 8b 00                 mov    (%rax),%rax  
  20:    48 8d 50 01              lea    0x1(%rax),%rdx  
  24:    48 8b 45 e0              mov    -0x20(%rbp),%rax  
  28:    48 89 10                 mov    %rdx,(%rax)  
  2b:    83 45 fc 01              addl   $0x1,-0x4(%rbp)  
  2f:    83 7d fc 09              cmpl   $0x9,-0x4(%rbp)  
  33:    7f 2d                    jg     62 <input\_arr+0x62>  
  35:    48 8b 45 e0              mov    -0x20(%rbp),%rax  
  39:    48 8b 00                 mov    (%rax),%rax  
  3c:    48 8d 14 85 00 00 00     lea    0x0(,%rax,4),%rdx  
  43:    00  
  44:    48 8b 45 e8              mov    -0x18(%rbp),%rax  
  48:    48 01 d0                 add    %rdx,%rax  
  4b:    48 89 c6                 mov    %rax,%rsi  
  4e:    bf 00 00 00 00           mov    $0x0,%edi  
  53:    b8 00 00 00 00           mov    $0x0,%eax  
  58:    e8 00 00 00 00           call   5d <input\_arr+0x5d>  
  5d:    83 f8 01                 cmp    $0x1,%eax  
  60:    74 b7                    je     19 <input\_arr+0x19>  
  62:    83 7d fc 0a              cmpl   $0xa,-0x4(%rbp)  
  66:    75 22                    jne    8a <input\_arr+0x8a>  
  68:    48 8d 45 f8              lea    -0x8(%rbp),%rax  
  6c:    48 89 c6                 mov    %rax,%rsi  
  6f:    bf 00 00 00 00           mov    $0x0,%edi  
  74:    b8 00 00 00 00           mov    $0x0,%eax  
  79:    e8 00 00 00 00           call   7e <input\_arr+0x7e>  
  7e:    83 f8 01                 cmp    $0x1,%eax  
  81:    75 07                    jne    8a <input\_arr+0x8a>  
  83:    b8 64 00 00 00           mov    $0x64,%eax  
  88:    eb 18                    jmp    a2 <input\_arr+0xa2>  
  8a:    48 8b 45 e0              mov    -0x20(%rbp),%rax  
  8e:    48 8b 00                 mov    (%rax),%rax  
  91:    48 85 c0                 test   %rax,%rax  
  94:    75 07                    jne    9d <input\_arr+0x9d>  
  96:    b8 01 00 00 00           mov    $0x1,%eax  
  9b:    eb 05                    jmp    a2 <input\_arr+0xa2>  
  9d:    b8 00 00 00 00           mov    $0x0,%eax  
  a2:    c9                       leave  
  a3:    c3                       ret  
  
00000000000000a4 <bubble\_sort>:  
  a4:    55                       push   %rbp  
  a5:    48 89 e5                 mov    %rsp,%rbp  
  a8:    48 89 7d d8              mov    %rdi,-0x28(%rbp)  
  ac:    48 89 75 d0              mov    %rsi,-0x30(%rbp)  
  b0:    48 c7 45 f8 00 00 00     movq   $0x0,-0x8(%rbp)  
  b7:    00  
  b8:    e9 c1 00 00 00           jmp    17e <bubble\_sort+0xda>  
  bd:    48 c7 45 f0 00 00 00     movq   $0x0,-0x10(%rbp)  
  c4:    00  
  c5:    e9 99 00 00 00           jmp    163 <bubble\_sort+0xbf>  
  ca:    48 8b 45 f0              mov    -0x10(%rbp),%rax  
  ce:    48 8d 14 85 00 00 00     lea    0x0(,%rax,4),%rdx  
  d5:    00  
  d6:    48 8b 45 d8              mov    -0x28(%rbp),%rax  
  da:    48 01 d0                 add    %rdx,%rax  
  dd:    8b 10                    mov    (%rax),%edx  
  df:    48 8b 45 f0              mov    -0x10(%rbp),%rax  
  e3:    48 83 c0 01              add    $0x1,%rax  
  e7:    48 8d 0c 85 00 00 00     lea    0x0(,%rax,4),%rcx  
  ee:    00  
  ef:    48 8b 45 d8              mov    -0x28(%rbp),%rax  
  f3:    48 01 c8                 add    %rcx,%rax  
  f6:    8b 00                    mov    (%rax),%eax  
  f8:    39 c2                    cmp    %eax,%edx  
  fa:    7e 62                    jle    15e <bubble\_sort+0xba>  
  fc:    48 8b 45 f0              mov    -0x10(%rbp),%rax  
 100:    48 8d 14 85 00 00 00     lea    0x0(,%rax,4),%rdx  
 107:    00  
 108:    48 8b 45 d8              mov    -0x28(%rbp),%rax  
 10c:    48 01 d0                 add    %rdx,%rax  
 10f:    8b 00                    mov    (%rax),%eax  
 111:    89 45 ec                 mov    %eax,-0x14(%rbp)  
 114:    48 8b 45 f0              mov    -0x10(%rbp),%rax  
 118:    48 83 c0 01              add    $0x1,%rax  
 11c:    48 8d 14 85 00 00 00     lea    0x0(,%rax,4),%rdx  
 123:    00  
 124:    48 8b 45 d8              mov    -0x28(%rbp),%rax  
 128:    48 01 d0                 add    %rdx,%rax  
 12b:    48 8b 55 f0              mov    -0x10(%rbp),%rdx  
 12f:    48 8d 0c 95 00 00 00     lea    0x0(,%rdx,4),%rcx  
 136:    00  
 137:    48 8b 55 d8              mov    -0x28(%rbp),%rdx  
 13b:    48 01 ca                 add    %rcx,%rdx  
 13e:    8b 00                    mov    (%rax),%eax  
 140:    89 02                    mov    %eax,(%rdx)  
 142:    48 8b 45 f0              mov    -0x10(%rbp),%rax  
 146:    48 83 c0 01              add    $0x1,%rax  
 14a:    48 8d 14 85 00 00 00     lea    0x0(,%rax,4),%rdx  
 151:    00  
 152:    48 8b 45 d8              mov    -0x28(%rbp),%rax  
 156:    48 01 c2                 add    %rax,%rdx  
 159:    8b 45 ec                 mov    -0x14(%rbp),%eax  
 15c:    89 02                    mov    %eax,(%rdx)  
 15e:    48 83 45 f0 01           addq   $0x1,-0x10(%rbp)  
 163:    48 8b 45 d0              mov    -0x30(%rbp),%rax  
 167:    48 2b 45 f8              sub    -0x8(%rbp),%rax  
 16b:    48 83 e8 01              sub    $0x1,%rax  
 16f:    48 39 45 f0              cmp    %rax,-0x10(%rbp)  
 173:    0f 82 51 ff ff ff        jb     ca <bubble\_sort+0x26>  
 179:    48 83 45 f8 01           addq   $0x1,-0x8(%rbp)  
 17e:    48 8b 45 d0              mov    -0x30(%rbp),%rax  
 182:    48 83 e8 01              sub    $0x1,%rax  
 186:    48 39 45 f8              cmp    %rax,-0x8(%rbp)  
 18a:    0f 82 2d ff ff ff        jb     bd <bubble\_sort+0x19>  
 190:    b8 00 00 00 00           mov    $0x0,%eax  
 195:    5d                       pop    %rbp  
 196:    c3                       ret  
  
0000000000000197 <print\_arr>:  
 197:    55                       push   %rbp  
 198:    48 89 e5                 mov    %rsp,%rbp  
 19b:    48 83 ec 20              sub    $0x20,%rsp  
 19f:    48 89 7d e8              mov    %rdi,-0x18(%rbp)  
 1a3:    48 89 75 e0              mov    %rsi,-0x20(%rbp)  
 1a7:    48 c7 45 f8 00 00 00     movq   $0x0,-0x8(%rbp)  
 1ae:    00  
 1af:    eb 2b                    jmp    1dc <print\_arr+0x45>  
 1b1:    48 8b 45 f8              mov    -0x8(%rbp),%rax  
 1b5:    48 8d 14 85 00 00 00     lea    0x0(,%rax,4),%rdx  
 1bc:    00  
 1bd:    48 8b 45 e8              mov    -0x18(%rbp),%rax  
 1c1:    48 01 d0                 add    %rdx,%rax  
 1c4:    8b 00                    mov    (%rax),%eax  
 1c6:    89 c6                    mov    %eax,%esi  
 1c8:    bf 00 00 00 00           mov    $0x0,%edi  
 1cd:    b8 00 00 00 00           mov    $0x0,%eax  
 1d2:    e8 00 00 00 00           call   1d7 <print\_arr+0x40>  
 1d7:    48 83 45 f8 01           addq   $0x1,-0x8(%rbp)  
 1dc:    48 8b 45 f8              mov    -0x8(%rbp),%rax  
 1e0:    48 3b 45 e0              cmp    -0x20(%rbp),%rax  
 1e4:    72 cb                    jb     1b1 <print\_arr+0x1a>  
 1e6:    90                       nop  
 1e7:    90                       nop  
 1e8:    c9                       leave  
 1e9:    c3                       ret  
  
00000000000001ea <main>:  
 1ea:    55                       push   %rbp  
 1eb:    48 89 e5                 mov    %rsp,%rbp  
 1ee:    48 83 ec 40              sub    $0x40,%rsp  
 1f2:    48 c7 45 c8 00 00 00     movq   $0x0,-0x38(%rbp)  
 1f9:    00  
 1fa:    bf 00 00 00 00           mov    $0x0,%edi  
 1ff:    b8 00 00 00 00           mov    $0x0,%eax  
 204:    e8 00 00 00 00           call   209 <main+0x1f>  
 209:    48 8d 55 c8              lea    -0x38(%rbp),%rdx  
 20d:    48 8d 45 d0              lea    -0x30(%rbp),%rax  
 211:    48 89 d6                 mov    %rdx,%rsi  
 214:    48 89 c7                 mov    %rax,%rdi  
 217:    e8 00 00 00 00           call   21c <main+0x32>  
 21c:    89 45 fc                 mov    %eax,-0x4(%rbp)  
 21f:    83 7d fc 01              cmpl   $0x1,-0x4(%rbp)  
 223:    75 11                    jne    236 <main+0x4c>  
 225:    bf 00 00 00 00           mov    $0x0,%edi  
 22a:    e8 00 00 00 00           call   22f <main+0x45>  
 22f:    b8 01 00 00 00           mov    $0x1,%eax  
 234:    eb 42                    jmp    278 <main+0x8e>  
 236:    48 8b 55 c8              mov    -0x38(%rbp),%rdx  
 23a:    48 8d 45 d0              lea    -0x30(%rbp),%rax  
 23e:    48 89 d6                 mov    %rdx,%rsi  
 241:    48 89 c7                 mov    %rax,%rdi  
 244:    e8 00 00 00 00           call   249 <main+0x5f>  
 249:    bf 00 00 00 00           mov    $0x0,%edi  
 24e:    b8 00 00 00 00           mov    $0x0,%eax  
 253:    e8 00 00 00 00           call   258 <main+0x6e>  
 258:    48 8b 55 c8              mov    -0x38(%rbp),%rdx  
 25c:    48 8d 45 d0              lea    -0x30(%rbp),%rax  
 260:    48 89 d6                 mov    %rdx,%rsi  
 263:    48 89 c7                 mov    %rax,%rdi  
 266:    e8 00 00 00 00           call   26b <main+0x81>  
 26b:    bf 0a 00 00 00           mov    $0xa,%edi  
 270:    e8 00 00 00 00           call   275 <main+0x8b>  
 275:    8b 45 fc                 mov    -0x4(%rbp),%eax  
 278:    c9                       leave  
 279:    c3                       ret

Отличие дизассемблированного файла от ассемблерного заключается в следующем:

* Рядом с командами находится их интерпретация в виде машинной инструкции
* Метки заменены цифрами
* Числа переведены в 16-ричную систему счисления

## **Глобальные и локальные переменные в исходной программе**

Исходный код после добавления глобальной проинициализированной переменной и глобальной неинициализированной переменной:

#include <stdio.h>

#include <stdlib.h>

#define HANDLER\_OWERFLOW 100

#define ERROR\_INPUT\_ARR 1

#define N 10

typedef int arr\_t[N];

int input\_arr(int \*a, size\_t \*n)

{

int count = 0;

int tmp;

while ((count < N) && (scanf("%d", &a[\*n]) == 1))

{

\*n = (\*n) + 1;

count = count + 1;

}

if ((count == N) && (scanf("%d", &tmp) == 1))

return HANDLER\_OWERFLOW;

//Пустой массив

if (\*n == 0)

return ERROR\_INPUT\_ARR;

return EXIT\_SUCCESS;

}

int bubble\_sort(int \*a, size\_t n)

{

for (size\_t i = 0; i < n - 1; i++)

{

for (size\_t j = 0; j < n - i - 1; j++)

{

if (a[j] > a[j + 1])

{

int tmp = a[j];

a[j] = a[j + 1];

a[j + 1] = tmp;

}

}

}

return 0;

}

void print\_arr(int \*dst, size\_t dlen)

{

for (size\_t i = 0; i < dlen; i++)

printf("%d ", dst[i]);

}

double global\_var\_init = 10;

double global\_var;

int main(void)

{

arr\_t a;

size\_t n = 0;

double local\_var\_init = 0;

double local\_var;

printf("Введите элементы массива: ");

int code\_rn = input\_arr(a, &n);

if (code\_rn == ERROR\_INPUT\_ARR)

{

printf("Некорректный ввод массива\n");

return ERROR\_INPUT\_ARR;

}

bubble\_sort(a, n);

printf("Массив упорядоченных элементов: ");

print\_arr(a, n);

printf("\n");

return code\_rn;

}

Таблица символов (команда nm):

$ nm main.o  
00000000000000a4 T bubble\_sort  
0000000000000000 B global\_var  
0000000000000000 D global\_var\_init  
0000000000000000 T input\_arr  
                 U \_\_isoc99\_scanf  
00000000000001ea T main  
0000000000000197 T print\_arr  
                 U printf  
                 U putchar  
                 U puts

Ключ «-s» для команды objdump отображает разделы файла.

Разделы файла:

$ objdump -s main.o  
  
main.o:     формат файла elf64-x86-64  
  
Содержимое раздела .text:  
 0000 554889e5 4883ec20 48897de8 488975e0  UH..H.. H.}.H.u.  
 0010 c745fc00 000000eb 16488b45 e0488b00  .E.......H.E.H..  
 0020 488d5001 488b45e0 48891083 45fc0183  H.P.H.E.H...E...  
 0030 7dfc097f 2d488b45 e0488b00 488d1485  }...-H.E.H..H...  
 0040 00000000 488b45e8 4801d048 89c6bf00  ....H.E.H..H....  
 0050 000000b8 00000000 e8000000 0083f801  ................  
 0060 74b7837d fc0a7522 488d45f8 4889c6bf  t..}..u"H.E.H...  
 0070 00000000 b8000000 00e80000 000083f8  ................  
 0080 017507b8 64000000 eb18488b 45e0488b  .u..d.....H.E.H.  
 0090 004885c0 7507b801 000000eb 05b80000  .H..u...........  
 00a0 0000c9c3 554889e5 48897dd8 488975d0  ....UH..H.}.H.u.  
 00b0 48c745f8 00000000 e9c10000 0048c745  H.E..........H.E  
 00c0 f0000000 00e99900 0000488b 45f0488d  ..........H.E.H.  
 00d0 14850000 0000488b 45d84801 d08b1048  ......H.E.H....H  
 00e0 8b45f048 83c00148 8d0c8500 00000048  .E.H...H.......H  
 00f0 8b45d848 01c88b00 39c27e62 488b45f0  .E.H....9.~bH.E.  
 0100 488d1485 00000000 488b45d8 4801d08b  H.......H.E.H...  
 0110 008945ec 488b45f0 4883c001 488d1485  ..E.H.E.H...H...  
 0120 00000000 488b45d8 4801d048 8b55f048  ....H.E.H..H.U.H  
 0130 8d0c9500 00000048 8b55d848 01ca8b00  .......H.U.H....  
 0140 8902488b 45f04883 c001488d 14850000  ..H.E.H...H.....  
 0150 0000488b 45d84801 c28b45ec 89024883  ..H.E.H...E...H.  
 0160 45f00148 8b45d048 2b45f848 83e80148  E..H.E.H+E.H...H  
 0170 3945f00f 8251ffff ff488345 f801488b  9E...Q...H.E..H.  
 0180 45d04883 e8014839 45f80f82 2dffffff  E.H...H9E...-...  
 0190 b8000000 005dc355 4889e548 83ec2048  .....].UH..H.. H  
 01a0 897de848 8975e048 c745f800 000000eb  .}.H.u.H.E......  
 01b0 2b488b45 f8488d14 85000000 00488b45  +H.E.H.......H.E  
 01c0 e84801d0 8b0089c6 bf000000 00b80000  .H..............  
 01d0 0000e800 00000048 8345f801 488b45f8  .......H.E..H.E.  
 01e0 483b45e0 72cb9090 c9c35548 89e54883  H;E.r.....UH..H.  
 01f0 ec5048c7 45b80000 0000660f efc0f20f  .PH.E.....f.....  
 0200 1145f8bf 00000000 b8000000 00e80000  .E..............  
 0210 0000488d 55b8488d 45c04889 d64889c7  ..H.U.H.E.H..H..  
 0220 e8000000 008945f4 837df401 7511bf00  ......E..}..u...  
 0230 000000e8 00000000 b8010000 00eb4248  ..............BH  
 0240 8b55b848 8d45c048 89d64889 c7e80000  .U.H.E.H..H.....  
 0250 0000bf00 000000b8 00000000 e8000000  ................  
 0260 00488b55 b8488d45 c04889d6 4889c7e8  .H.U.H.E.H..H...  
 0270 00000000 bf0a0000 00e80000 00008b45  ...............E  
 0280 f4c9c3                               ...  
Содержимое раздела .data:  
 0000 00000000 00002440                    ......$@  
Содержимое раздела .rodata:  
 0000 25640025 64200000 d092d0b2 d0b5d0b4  %d.%d ..........  
 0010 d0b8d182 d0b520d1 8dd0bbd0 b5d0bcd0  ...... .........  
 0020 b5d0bdd1 82d18b20 d0bcd0b0 d181d181  ....... ........  
 0030 d0b8d0b2 d0b03a20 00000000 00000000  ......: ........  
 0040 d09dd0b5 d0bad0be d180d180 d0b5d0ba  ................  
 0050 d182d0bd d18bd0b9 20d0b2d0 b2d0bed0  ........ .......  
 0060 b420d0bc d0b0d181 d181d0b8 d0b2d0b0  . ..............  
 0070 00000000 00000000 d09cd0b0 d181d181  ................  
 0080 d0b8d0b2 20d183d0 bfd0bed1 80d18fd0  .... ...........  
 0090 b4d0bed1 87d0b5d0 bdd0bdd1 8bd18520  ...............  
 00a0 d18dd0bb d0b5d0bc d0b5d0bd d182d0be  ................  
 00b0 d0b23a20 00                          ..: .  
Содержимое раздела .comment:  
 0000 00474343 3a202847 4e552920 31322e32  .GCC: (GNU) 12.2  
 0010 2e312032 30323231 31323120 28526564  .1 20221121 (Red  
 0020 20486174 2031322e 322e312d 342900     Hat 12.2.1-4).  
Содержимое раздела .note.gnu.property:  
 0000 04000000 20000000 05000000 474e5500  .... .......GNU.  
 0010 020001c0 04000000 01000000 00000000  ................  
 0020 010001c0 04000000 09000000 00000000  ................  
Содержимое раздела .eh\_frame:  
 0000 14000000 00000000 017a5200 01781001  .........zR..x..  
 0010 1b0c0708 90010000 1c000000 1c000000  ................  
 0020 00000000 a4000000 00410e10 8602430d  .........A....C.  
 0030 06029f0c 07080000 1c000000 3c000000  ............<...  
 0040 00000000 f3000000 00410e10 8602430d  .........A....C.  
 0050 0602ee0c 07080000 1c000000 5c000000  ............\...  
 0060 00000000 53000000 00410e10 8602430d  ....S....A....C.  
 0070 06024e0c 07080000 1c000000 7c000000  ..N.........|...  
 0080 00000000 99000000 00410e10 8602430d  .........A....C.  
 0090 0602940c 07080000                    ........

Ключ «-t» для команды objdump отображает секции переменных и функций

Секции переменных и функций:

$ objdump -t main.o  
  
main.o:     формат файла elf64-x86-64  
  
SYMBOL TABLE:  
0000000000000000 l    df \*ABS\*    0000000000000000 main.c  
0000000000000000 l    d  .text    0000000000000000 .text  
0000000000000000 l    d  .rodata    0000000000000000 .rodata  
0000000000000000 g     F .text    00000000000000a4 input\_arr  
0000000000000000         \*UND\*    0000000000000000 \_\_isoc99\_scanf  
00000000000000a4 g     F .text    00000000000000f3 bubble\_sort  
0000000000000197 g     F .text    0000000000000053 print\_arr  
0000000000000000         \*UND\*    0000000000000000 printf  
0000000000000000 g     O .data    0000000000000008 global\_var\_init  
0000000000000000 g     O .bss    0000000000000008 global\_var  
00000000000001ea g     F .text    0000000000000099 main  
0000000000000000         \*UND\*    0000000000000000 puts  
0000000000000000         \*UND\*    0000000000000000 putchar

Функции находятся в секции .txt

Проинициализированные глобальные и локальные переменные находятся в секции .data

Неинициализированные глобальные и локальные переменные находятся в секции .bss

## **Отладочная информация**

Добавляем отладочную информацию в файл при помощи ключа «-g» в команде gcc. Количество информации (подробность) указываем цифрами от 1 до 3. Будем использовать ключ «-g3»:

$ gcc -std=c99 -c main.c -g3 -lm -o main\_debug.o

Таблица символов:

$ nm main\_debug.o  
00000000000000a4 T bubble\_sort  
0000000000000000 B global\_var  
0000000000000000 D global\_var\_init  
0000000000000000 T input\_arr  
                 U \_\_isoc99\_scanf  
00000000000001ea T main  
0000000000000197 T print\_arr  
                 U printf  
                 U putchar  
                 U puts  
0000000000000000 n wm4.0.8a34639624ab4113341b5db51650960f  
0000000000000000 n wm4.cdefs.h.20.d078fe069b8c68005efa8ff3a158391b  
0000000000000000 n wm4.cdefs.h.616.8d7ca1b9d01e52f5b2c040c19a111f7b  
0000000000000000 n wm4.features.h.19.be13bb4b33b2be4d5fdeac670166e1a8  
0000000000000000 n wm4.features.h.432.600d23269265a0a96d6ec6df7a9f596a  
0000000000000000 n wm4.floatncommon.h.34.7e1840d7dfb19e9bdb51aeb077d76637  
0000000000000000 n wm4.floatn.h.20.a55feb25f1f7464b830caad4873a8713  
0000000000000000 n wm4.libcheaderstart.h.31.045646cfd09d1c615866e08d91c4f364  
0000000000000000 n wm4.libcheaderstart.h.37.e7d4b6f4649b40d3e0dce357ae78234f  
0000000000000000 n wm4.stdarg.h.34.3a23a216c0c293b3d2ea2e89281481e6  
0000000000000000 n wm4.stdcpredef.h.19.88fdbfd5cf6f83ed579effc3e425f09b  
0000000000000000 n wm4.stddef.h.185.cbb642e1ccd385e8aa504b15cb7fb086  
0000000000000000 n wm4.stddef.h.237.2a7f4947d4b7296e7e393bf9a618c3c1  
0000000000000000 n wm4.stdio.h.147.dbd603e98db8f3e1583090fab2abd54e  
0000000000000000 n wm4.stdio.h.24.5c1b97eef3c86b7a2549420f69f4f128  
0000000000000000 n wm4.stdio.h.31.e39a94e203ad4e1d978c0fc68ce016ee  
0000000000000000 n wm4.stdio.h.93.0122ffbd02ddfe34dfaf44a2e3561c5b  
0000000000000000 n wm4.stdio\_lim.h.19.de8a40c165be4f8437982ec2cd6fd8b4  
0000000000000000 n wm4.stdlib.h.29.dde59e751a3b6c4506ba901b60a85c87  
0000000000000000 n wm4.stdlib.h.36.d43beea9355ea645b197955d416b700d  
0000000000000000 n wm4.struct\_FILE.h.19.0888ac70396abe1031c03d393554032f  
0000000000000000 n wm4.stubs64.h.10.7865f4f7062bab1c535c1f73f43aa9b9  
0000000000000000 n wm4.time64.h.24.a8166ae916ec910dab0d8987098d42ee  
0000000000000000 n wm4.types.h.109.56eb9ae966b255288cc544f18746a7ff  
0000000000000000 n wm4.typesizes.h.24.ccf5919b8e01b553263cf8f4ab1d5fde  
0000000000000000 n wm4.wordsize.h.4.baf119258a1e53d8dba67ceac44ab6bc

Секции переменных и функций:

$ objdump -t main\_debug.o  
  
main\_debug.o:     формат файла elf64-x86-64  
  
SYMBOL TABLE:  
0000000000000000 l    df \*ABS\*    0000000000000000 main.c  
0000000000000000 l    d  .text    0000000000000000 .text  
0000000000000000 l    d  .rodata    0000000000000000 .rodata  
0000000000000000 l    d  .debug\_info    0000000000000000 .debug\_info  
0000000000000000 l    d  .debug\_abbrev    0000000000000000 .debug\_abbrev  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_line    0000000000000000 .debug\_line  
0000000000000000 l    d  .debug\_str    0000000000000000 .debug\_str  
0000000000000000 l    d  .debug\_line\_str    0000000000000000 .debug\_line\_str  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l    d  .debug\_macro    0000000000000000 .debug\_macro  
0000000000000000 l       .group    0000000000000000  
wm4.0.8a34639624ab4113341b5db51650960f  
0000000000000000 l       .group    0000000000000000  
wm4.stdcpredef.h.19.88fdbfd5cf6f83ed579effc3e425f09b  
0000000000000000 l       .group    0000000000000000  
wm4.stdio.h.24.5c1b97eef3c86b7a2549420f69f4f128  
0000000000000000 l       .group    0000000000000000  
wm4.features.h.19.be13bb4b33b2be4d5fdeac670166e1a8  
0000000000000000 l       .group    0000000000000000  
wm4.wordsize.h.4.baf119258a1e53d8dba67ceac44ab6bc  
0000000000000000 l       .group    0000000000000000  
wm4.features.h.432.600d23269265a0a96d6ec6df7a9f596a  
0000000000000000 l       .group    0000000000000000  
wm4.cdefs.h.20.d078fe069b8c68005efa8ff3a158391b  
0000000000000000 l       .group    0000000000000000  
wm4.cdefs.h.616.8d7ca1b9d01e52f5b2c040c19a111f7b  
0000000000000000 l       .group    0000000000000000  
wm4.stubs64.h.10.7865f4f7062bab1c535c1f73f43aa9b9  
0000000000000000 l       .group    0000000000000000  
wm4.libcheaderstart.h.37.e7d4b6f4649b40d3e0dce357ae78234f  
0000000000000000 l       .group    0000000000000000  
wm4.stdio.h.31.e39a94e203ad4e1d978c0fc68ce016ee  
0000000000000000 l       .group    0000000000000000  
wm4.stddef.h.185.cbb642e1ccd385e8aa504b15cb7fb086  
0000000000000000 l       .group    0000000000000000  
wm4.stdarg.h.34.3a23a216c0c293b3d2ea2e89281481e6  
0000000000000000 l       .group    0000000000000000  
wm4.types.h.109.56eb9ae966b255288cc544f18746a7ff  
0000000000000000 l       .group    0000000000000000  
wm4.typesizes.h.24.ccf5919b8e01b553263cf8f4ab1d5fde  
0000000000000000 l       .group    0000000000000000  
wm4.time64.h.24.a8166ae916ec910dab0d8987098d42ee  
0000000000000000 l       .group    0000000000000000  
wm4.struct\_FILE.h.19.0888ac70396abe1031c03d393554032f  
0000000000000000 l       .group    0000000000000000  
wm4.stdio.h.93.0122ffbd02ddfe34dfaf44a2e3561c5b  
0000000000000000 l       .group    0000000000000000  
wm4.stdio\_lim.h.19.de8a40c165be4f8437982ec2cd6fd8b4  
0000000000000000 l       .group    0000000000000000  
wm4.stdio.h.147.dbd603e98db8f3e1583090fab2abd54e  
0000000000000000 l       .group    0000000000000000  
wm4.floatn.h.20.a55feb25f1f7464b830caad4873a8713  
0000000000000000 l       .group    0000000000000000  
wm4.floatncommon.h.34.7e1840d7dfb19e9bdb51aeb077d76637  
0000000000000000 l       .group    0000000000000000  
wm4.libcheaderstart.h.31.045646cfd09d1c615866e08d91c4f364  
0000000000000000 l       .group    0000000000000000  
wm4.stdlib.h.29.dde59e751a3b6c4506ba901b60a85c87  
0000000000000000 l       .group    0000000000000000  
wm4.stddef.h.237.2a7f4947d4b7296e7e393bf9a618c3c1  
0000000000000000 l       .group    0000000000000000  
wm4.stdlib.h.36.d43beea9355ea645b197955d416b700d  
0000000000000000 g     F .text    00000000000000a4 input\_arr  
0000000000000000         \*UND\*    0000000000000000 \_\_isoc99\_scanf  
00000000000000a4 g     F .text    00000000000000f3 bubble\_sort  
0000000000000197 g     F .text    0000000000000053 print\_arr  
0000000000000000         \*UND\*    0000000000000000 printf  
0000000000000000 g     O .data    0000000000000008 global\_var\_init  
0000000000000000 g     O .bss    0000000000000008 global\_var  
00000000000001ea g     F .text    0000000000000099 main  
0000000000000000         \*UND\*    0000000000000000 puts  
0000000000000000         \*UND\*    0000000000000000 putchar

В сравнении с предыдущим выводом секций, в новом добавились отладочные символы и отладочные секции.

## **Получение исполняемого файла**

$ gcc -o main.exe main.c –lm

## **Вопросы**

1. Объектный и исполняемый файлы с отладочной информацией больше по размеру по сравнению с ими же, но без информации.

-rwxr-xr-x. 1 Natalia Natalia 55408 мар 19 13:48 main\_debug.exe

-rwxr-xr-x. 1 Natalia Natalia 25288 мар 19 13:42 main.exe  
-rw-r--r--. 1 Natalia Natalia 62216 мар 19 13:31 main\_debug.o  
-rw-r--r--. 1 Natalia Natalia  3096 мар 19 12:23 main.o

1. Объектный и исполняемый файлы с отладочной информацией имеют большее количество секций по сравнению с ими же, но без информации.
2. Расположение функций, глобальных и локальных переменных в объектных и исполняемых файлах с отладочной информацией и без неё не отличается.

## **Используемые динамические библиотеки**

Команда ldd позволяет узнать какие динамические библиотеки использует исполняемый файл.

$ ldd main.exe  
    linux-vdso.so.1 (0x00007ffe6cae0000)  
    libm.so.6 => /lib64/libm.so.6 (0x00007f31ebab6000)  
    libc.so.6 => /lib64/libc.so.6 (0x00007f31eb8d9000)  
    /lib64/ld-linux-x86-64.so.2 (0x00007f31ebbac000)

Исполняемый файл main.exe использует следующие динамические библиотеки:

1. linux-vdso.so.1
2. libm.so.6
3. libc.so.6
4. ld-linux-x86-64.so.2