Министерство науки и образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
"Московский институт электронной техники"
(МИЭТ)

Отчет по лабораторной работе № 2

Операционные системы

Выполнил: студент ПМ - 31

Мартынова Мария Олеговна

2023 г.

Задание 1

В папке swap лежит 3 файла: swap.c, swap.h и main.c. Ваша задача закончить метод Swap в swap.c, так, чтобы он менял местами два символа. Скомпилировать программу. Если вы все сделали верно, то программа, которую вы собрали выведет "b a".

1) • Если функция не использует указатели или ссылки, **она** не может изменить значение параметра.

• Для изменения значения параметра функция должна знать адрес параметра в памяти.

• Оператор адреса C (&) позволяет вашей программе определить адрес переменной в памяти.

• Когда ваша программа узнает адрес памяти, она сможет использовать операцию разыменования C (*) для определения значения, хранимого по данному адресу.

• Если программе нужно изменить значение параметров функции, программа передает в функцию адрес параметра.

2) GCC - это свободно доступный оптимизирующий компилятор для языков C, C++.

Чтобы откомпилировать исходный код C++, находящийся в файле F.c, и создать объектный файл F.o, выполните команду:

gcc -c <compile-options> F.c

Здесь строка compile-options указывает возможные дополнительные опции компиляции.

-c – только компиляция. Из исходных файлов программы создаются объектные файлы в виде name.o. Компоновка не производится.

-Wall – вывод сообщений о всех предупреждениях или ошибках, возникающих во время трансляции программы.

```
~/oslab2019$ cd lab2/src
~/.../lab2/src$
~/.../lab2/src$ cd swap
~/.../src/swap$
```

// изменение местами содержимого через временную переменную

lab2/src/swap/swap.c

```c
#include "swap.h"

void Swap(char *left, char *right)
{
  char t=*left;
  *left=*right;
  *right=t;

}
```

```
~/.../src/swap$ gcc -Wall -c swap.c
~/.../src/swap$ gcc -Wall -c main.c
~/.../src/swap$ gcc main.o swap.o -o swap
~/.../src/swap$ ./swap
b a
~/.../src/swap$
```

Задание 2

В папке revert_string содержатся файлы main.c, revert_string.h, revert_string.c. Вам необходимо реализовать метод RevertString в revert_string.c, который должен переворачивать данную пользователем строку. Изучить код main.c, скомпилировать программу, рассказать, как она работает и, что делает.

1) Для выделения памяти на куче в си используется функция ***malloc*** (memory allocation) из библиотеки stdlib.h.

Освобождение памяти с помощью ***free.*** Функция free освобождает память, но при этом не изменяет значение указателя.

2) Основное различие между стеком и кучей заключается в том, что стек включает в себя линейное и последовательное выделение памяти, которая используется для статического выделения памяти, тогда как куча выступает в качестве пула области хранения, которая выделяет память случайным образом (динамическое выделение памяти).

Стек — это область оперативной памяти, которая создаётся для каждого потока. Он работает в порядке LIFO (Last In, First Out).

Куча — это хранилище памяти, также расположенное в ОЗУ, которое допускает динамическое выделение памяти и не работает по принципу стека: это просто склад для ваших переменных.

3) **Аргумент командной строки** – это информация, которая вводится в командной строке операционной системы вслед за именем программы

```
~/oslab2019$ cd lab2/src/revert_string
~/.../src/revert_string$ gcc -Wall -c main.c
~/.../src/revert_string$ gcc -Wall -c revert_string.c
~/.../src/revert_string$ gcc main.o revert_string.o -
o revert_string
~/.../src/revert_string$ ./revert_string Green
Reverted: neerG
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "revert_string.h"

int main(int argc, char *argv[]) // argc -количесвто аргументов, передаваемых через командную строку, argv- адреса переданных аргументов
{
  if (argc != 2)
  {
    printf("Usage: %s string_to_revert\n", argv[0]);
    return -1;
  }

  char *reverted_str = malloc(sizeof(char) * (strlen(argv[1]) + 1)); // выделение памяти под строку
  strcpy(reverted_str, argv[1]); // копирование введенной строки

  RevertString(reverted_str); // функция, которая переворачивает строку

  printf("Reverted: %s\n", reverted_str);
  free(reverted_str);
  return 0;
}
```

```c
#include "revert_string.h"
#include <string.h>

void RevertString(char *str)
{
  int i,j;
  for (i=0,j=strlen(str)-1;i<j;i++,j--)
    {
      char t=str[i];
      str[i]=str[j];
      str[j]=t;
    }
}
```

Задание 3

В **задании 2**, вы написали маленькую библиотеку с одной функцией, переворота строки. Тем не менее этот код уже можно переиспользовать, а чтобы это было удобнее делать, его необходимо вынести в библиотеку. Ваше задание скомпилировать статическую и динамическую библиотеки с RevertString и залинковать их в приложения с main.c.

1)

Состоит из 3 этапов:

1. Обработка исходного кода препроцессором (preprocessing)
2. Компиляция, то есть перевод подготовленного исходного кода в инструкции процессора (объектный файл) (compiling)
3. Компоновка - сборка одного или нескольких объектных файлов в один исполняемый файл (linking)

3)

-I – используется для добавления ваших собственных каталогов для поиска заголовочных файлов в процессе сборки.

-L – передается компоновщику. Используется для добавления ваших собственных каталогов для поиска библиотек в процессе сборки.
-l – передается компоновщику. Используется для добавления ваших собственных библиотек для поиска в процессе сборки.

-c – только компиляция. Из исходных файлов программы создаются объектные файлы в виде name.o. Компоновка не производится.

-o – поместить вывод в файл 'файл'

-fPIC - порождает позиционно независимый код, подходящий для динамической линковки и не имеющий никаких ограничений на размер глобальной таблицы смещений.

-shared – производит разделяемый объект, который может затем быть слинкован с другими объектами, чтобы сформироваться исполнимый файл.

4) Команда **ar** позволяет использовать утилиту для архивации данных с соответствующим именем.

5) LD_LIBRARY_PATH

Для какого-то конкретного запуска приложения, можно подменить библиотеки. В Linux переменная окружения LD_LIBRARY_PATH - это список отделённых двоеточиями имён каталогов, где библиотеки следует искать перед тем, как их будут искать по стандартным путям.

Статическая

```
~/.../src/revert_string$ ar rcs librevert.a revert_st
ring.o
~/.../src/revert_string$ gcc main.o -L. -lrevert -o s
tatic
~/.../src/revert_string$ ./static neerG
Reverted: Green
```

Динамическая

```
~/.../src/revert_string$ gcc -fPIC -c revert_string.c
~/.../src/revert_string$ gcc -shared revert_string.o -o librevert_shared.so
~/.../src/revert_string$ LD_LIBRARY_PATH=.
~/.../src/revert_string$ gcc main.o -L. -lrevert_shared -o revert_shared
~/.../src/revert_string$ ./revert_shared neerG
Reverted: Green
```