

Задание 1

Необходимые знания

1. Аргументы командной строки
2. Сборка с помощью gcc (clang)

Написать функцию GetMinMax в find_max_min.c, которая ищет минимальный и максимальный элементы массива, на заданном промежутке. Разобраться, что делает программа в sequential_min_max.c, скомпилировать, проверить, что написанный вами GetMinMax работает правильно.

1. Аргументы командной строки

Ответ на этот вопрос есть в прошлой лабе

2. Сборка с помощью GCC/Clang

Основные команды:

Команда	Описание
gcc main.c -o program	Сборка в исполняемый файл program
gcc -c file.c	Компиляция в объектный файл (file.o)
gcc file1.o file2.o -o program	Линковка объектных файлов
gcc -Wall -Wextra main.c	Включение всех предупреждений
clang	Аналогично GCC (более строгий анализ кода)

```
lab3 > src > C find_min_max.c
1  #include "find_min_max.h"
2  #include <limits.h>
3
4  struct MinMax GetMinMax(int *array, unsigned int begin, unsigned int end) {
5      struct MinMax min_max;
6      min_max.min = INT_MAX;
7      min_max.max = INT_MIN;
8
9      // Поиск минимального и максимального значений в заданном диапазоне
10     for (unsigned int i = begin; i < end; i++) {
11         if (array[i] < min_max.min) {
12             min_max.min = array[i];
13         }
14         if (array[i] > min_max.max) {
15             min_max.max = array[i];
16         }
17     }
18
19     return min_max;
20 }
```

С файлом разобрался, написал подробные комментарии.

```
● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ gcc find_min_max.c utils.c sequential_min_max.c -o min_max_program
● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ ./min_max_program
Usage: ./min_max_program seed arraysize
● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ ./min_max_program 42 20
min: 53523743
max: 1854614940
○ @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $
```

Задание 2 - 3

Необходимые знания

1. Аргументы командной строки
2. Системный вызов `fork`
3. Системный вызов `pipe`
4. Работа с файлами в Си

Завершить программу `parallel_min_max.c`, так, чтобы задача нахождения минимума и максимума в массиве решалась параллельно. Если выставлен аргумент `by_files` для синхронизации процессов использовать файлы (задание 2), в противном случае использовать `pipe` (задание 3).

1. Аргументы командной строки

Этот вопрос тоже был

2. Системный вызов `fork()`

Что делает:

Создает копию процесса (дочерний процесс).

```
#include <unistd.h>
#include <stdio.h>

int main() {
    pid_t    = fork(); // Делим процесс на два

    if (    == 0) {
        printf("Дочерний процесс (PID: %d)\n", getpid());
    } else {
        printf("Родительский процесс (PID: %d)\n", getpid());
    }
}
```

Вывод

Родительский процесс (PID: 123)

Дочерний процесс (PID: 124)

3. Системный вызов `pipe()`

Что делает:

Создает канал для обмена данными между процессами.

```
#include <unistd.h>
#include <stdio.h>

int main() {
    int  [2];
    pipe( ); // fd[0] — чтение, fd[1] — запись
```

```

if (fork() == 0) {
    close( [0]); // Закрываем чтение
    write( [1], "Hello", 6); // Пишем в канал
} else {
    close( [1]);
    char [6];
    read( [0], , 6); // Читаем из канала
    printf("Получено: %s\n", ); // "Hello"
}
}

```

4. Работа с файлами в С

Основные функции:

- `fopen()` — открытие файла.
- `fclose()` — закрытие.
- `fread()/fwrite()` — чтение/запись.

```

● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ touch parallel_min_max.c
○ @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $

```

Код написал

```

● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ gcc parallel_min_max.c find_min_max.c utils.c -o parallel_min_max
● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ ./parallel_min_max --seed 20 --array_size 20 --pnum 20
Min: 175990765
Max: 2132438409
Elapsed time: 1.553000ms
○ @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $

```

Задание 4

Необходимые знания

1. Как работают Makefile'ы

Изучить все targets в makefile, будьте готовы объяснить, за что они отвечают. Используя `makefile`, собрать получившиеся решения. Добавьте target `all`, отвечающий за сборку всех программ.

```

● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ make all
gcc -Wall -Wextra -c sequential_min_max.c -o sequential_min_max.o
gcc -Wall -Wextra -c find_min_max.c -o find_min_max.o
gcc -Wall -Wextra -c utils.c -o utils.o
utils.c: In function 'GenerateArray':
utils.c:7:21: warning: comparison of integer expressions of different signedness: 'int' and 'unsigned int' [-Wsign-compare]
  7 |     for (int i = 0; i < array_size; i++) {
    |                     ^
gcc -Wall -Wextra sequential_min_max.o find_min_max.o utils.o -o sequential_min_max
gcc -Wall -Wextra -c parallel_min_max.c -o parallel_min_max.o
parallel_min_max.c: In function 'main':
parallel_min_max.c:24:9: warning: unused variable 'current_optind' [-Wunused-variable]
  24 |     int current_optind = optind ? optind : 1;
    |         ^~~~~~
gcc -Wall -Wextra parallel_min_max.o find_min_max.o utils.o -o parallel_min_max
○ @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ make clean
bash: make: command not found
● @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ make clean
rm -f sequential_min_max parallel_min_max *.o
○ @ccurecc →/workspaces/os_lab_2019/lab3/src (master) $

```

1. Как работают Makefile'ы

Makefile — это скрипт для автоматизации сборки проектов, который определяет:

- **Зависимости** между файлами.
- **Команды** для компиляции и линковки.
- **Правила** для очистки, тестирования и других задач.

Зачем это нужно?

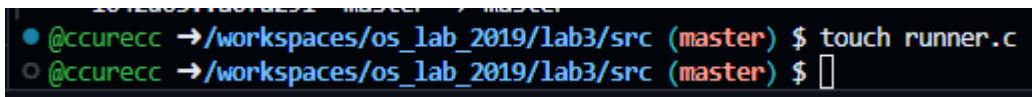
Автоматизация — не нужно вручную компилировать каждый файл.

Инкрементная сборка — пересобираются только измененные файлы.

Управление зависимостями — например, пересборка при изменении .h-файлов.

Автоматические переменные

- `$@` — имя цели.
- `$^` — все зависимости.
- `$<` — первая зависимость.



```
@ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ touch runner.c
@ccurecc →/workspaces/os_lab_2019/lab3/src (master) $
```

Написал код для ранера

Отредактировал мейк файл

Сто раз его чинил

Задание 5

Необходимые знания

1. Системный вызов `exec`

Написать программу, которая запускает в отдельном процессе ваше приложение `sequential_min_max`. Добавить его сборку в ваш `makefile`.

1. [Мануал для exec](#)
2. [Небольшой пример, как комбинировать fork и exec](#)

1. Семейство функций exec

Включает несколько вариантов:

Функция	Описание
execl	Принимает список аргументов (завершается NULL)
execv	Принимает массив аргументов (char *argv[])
execlp, execvp	Ищут программу в PATH (не нужно указывать полный путь)
execle, execvpe	Передают переменные окружения (дополнительный аргумент char *envp[])

```
@ccurecc →/workspaces/os_lab_2019/lab3/src (master) $ ./runner 42 10
Executing: ./sequential_min_max 42 10
min: 71876166
max: 1854614940
Child process exited with status 0
@ccurecc →/workspaces/os_lab_2019/lab3/src (master) $
```