

Лабораторная работа #4

# Автоматическая сборка многофайловых проектов

*[bash, GNU make, Makefiles, rand, srand, time]*



# ЛАБОРАТОРНАЯ РАБОТА #4

## Автоматическая сборка многофайловых проектов

### Цель работы

Научиться эффективно использовать специальные средства для автоматизации процесса компиляции, сборки и запуска многофайловых проектов.

### Требования

- 1) Для автоматизации сборки проекта необходимо использовать стандартную оболочку сценариев Linux **bash** (`/bin/sh`), а в качестве утилиты автосборщика – **GNU make**.
- 2) Для автосборки проекта с использованием **GNU make** необходимо описать два типа *make-файлов*: простой и параметризованный.
- 3) При выполнении задания **запрещается** использовать *IDE*. Рекомендуется задействовать любой текстовый редактор (к примеру, **gedit**) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования C/C++ **gcc/g++**, а для отладки – **gdb**-отладчик.

### Основное задание

Необходимо переработать основное задание из предыдущей лабораторной работы таким образом, чтобы пользователь (игрок) загадывал число, а компьютер, используя оптимальный и эффективный алгоритм, его отгадывал.

### Индивидуальное задание

Для программ, которые были разработаны в двух предыдущих лабораторных работах №2 и №3 необходимо добавить автоматизацию сборки многофайлового проекта с использованием сценарных оболочек и автосборщиков.

*Best of LUCK with it, and remember to HAVE FUN while you're learning :)*



## Контрольные вопросы

1. Какие существуют способы и средства для автоматической сборки многофайловых проектов? Опишите категории утилит автосборок в Linux, а также их преимущества и недостатки?
2. Как осуществить сборку и запуск проекта с использованием исполняемого файла стандартной скриптовой оболочки ОС Linux **bash**?
3. Зачем и где используется утилита **GNU make** (автосборщик)?
4. Зачем нужен **make-файл**? Что он содержит?
5. Опишите общий алгоритм процесса автоматической сборки программы.
6. Опишите базовый синтаксис make-файла и общие правила определения целевых связей в make-файле: целей (*targets*), зависимостей (*dependencies*) и инструкций (*instructions*)? Какая цель будет обрабатываться самой первой при запуске утилиты make? Что такое «основная цель сборки» и что в неё входит?
7. Какое необходимо дать стандартное имя make-файлу, чтобы его автоматически распознал автосборщик? А если в проекте используется нестандартное имя make-файла, как нужно вызывать утилиту make?
8. Параметризация процесса сборки и переменные make-файла?
9. Зачем нужен рекурсивный вызов make-утилиты? Как его осуществить?
10. Как избежать перекомпиляции проекта и при этом сделать все файлы исходного кода программы последними или «свежими» (*up to date*), т.е. не нуждающимися в перекомпиляции?
11. Как генерировать случайные значения с использованием стандартных функций **rand()** и **srand(...)** из стандартной C-библиотеки **stdlib**? Приведите пример. Как искусственно с помощью данных функций задать диапазон генерирования случайных значений?