# Лабораторная работа #2. Однопоточный администратор ресурсов “Генератор псевдослучайных чисел BBS”

## **Теоретические сведения**

Ранее, в лабораторной работе #1, для генерации псевдослучайной последовательности (ПСП) использовался линейный конгруэнтный генератор (ЛКГ). Данный генератор обладает хорошей производительностью, однако малым периодом повторения последовательности и другими аномалиями, что делает его не пригодным для использования в криптографических задачах.

[Алгоритм BBS (Blum-Blum-Shub)](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BB%D1%8E%D0%BC_%E2%80%94_%D0%91%D0%BB%D1%8E%D0%BC%D0%B0_%E2%80%94_%D0%A8%D1%83%D0%B1%D0%B0), напротив, позволяет получить в высокой степени криптографическостойкую последовательность, но характеризуется медленной работой. Вычисление производится по рекуррентной формуле (1).

xn+1 = xn2 mod M, (1)

где M = pq, p и q - простые числа, такие, что M является [числом Блюма](https://oeis.org/A016105). Выходом Yn+1 генератора будет являться [бит чётности](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D1%82_%D1%87%D1%91%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B8) элемента xn+1 (2):

Yn+1 = paritybit(xn+1). (2)

Параметрами генератора, таким образом, будут являться x0 (seed, начальное состояние), а также значения p и q.

*Вопрос. Почему данный алгоритм является более сложным, чем алгоритм ЛКГ?*

## **2. Описание протокола взаимодействия “Клиент-АР”**

Протокол должен быть описан в общем заголовочном файле bbs.h. Коды команд определяются с помощью макросов [*\_\_DIOT* и*\_\_DION*](https://www.qnx.com/developers/docs/6.5.0SP1.update/com.qnx.doc.neutrino_lib_ref/d/devctl.html)*.* Взаимодействие “Клиент-АР” осуществляется по интерфейсу POSIX с использованием функций [open](http://www.qnx.com/developers/docs/6.5.0/index.jsp?topic=%2Fcom.qnx.doc.neutrino_lib_ref%2Fo%2Fopen.html), [devctl](https://www.qnx.com/developers/docs/6.5.0SP1.update/com.qnx.doc.neutrino_lib_ref/d/devctl.html), [close](http://www.qnx.com/developers/docs/6.5.0/index.jsp?topic=%2Fcom.qnx.doc.neutrino_lib_ref%2Fo%2Fopen.html). Запросы, описанные ниже, передаются через функцию devctl по валидному дескриптору, полученному с помощью успешного вызова функции open. +

Запрос 1. Установка параметров генератора.

Код запроса: 1 // \_\_DION(class, cmd, data)

Входные данные: структура, содержащая параметры генератора:

struct BBSParams {

uint32\_t seed;

uint32\_t p;

uint32\_t q;

};

Выходные данные: нет

Запрос 2. Получение элемента ПСП

Код запроса: 2 \_\_DIOF(class, cmd, data)

Входные данные: нет

Выходные данные: uint32\_t

!Вопрос. Назовите недостатки предложенного Протокола

## **3. Разработка клиента**

Реализовать программу-клиент cryptobbs-client для платформы QNX Neutrino 6.6 (x86). Первичность разработки клиента можно понимать в контексте концепции TDD (test-driven-development), когда код верхнего уровня, использующий некий интерфейс, разрабатывается в первую очередь, перед реализацией самого интерфейса. Это позволяет лучше продумать заранее решение ряда проблем, в том числе, особенности сценариев использования интерфейса и обработку ошибок.

Клиент должен работать следующим образом.

1. Открывать соединение с АР с помощью функции open(“/dev/cryptobbs”).+
2. Устанавливать параметры генератора BBS (p = 3, q = 263, seed = 866) с помощью функции devctl(). \*
3. В цикле посылать запросы к АР с помощью функции devctl() с требованием получения очередного элемента ПСП.
4. Сохранять очередной элемент в вектор фиксированной длины 1024, работающий по принципу кольцевого буфера.
5. Цикл завершается при нажатии пользователем сочетания Ctrl + C (получения процессом SIGINT).
6. Содержимое вектора выводится в stdout.
7. Соединение с АР закрывается с помощью функции close().

*Вопрос. Нужен ли sleep между запросами?*

*Вопрос. До каких пор АР будет работать в реальном времени?*

*Вопрос. Как можно измерить максимальную нагрузку на АР, при котором он сможет обеспечивать работу в реальном времени?*

## **4. Разработка администратора ресурсов**

Реализовать администратор ресурсов cryptobbs для платформы QNX Neutrino 6.6 (x86), удовлетворяющий следующим требованиям:

* за основу использует [скелет однопоточного АР](http://www.qnx.com/developers/docs/7.0.0/#com.qnx.doc.neutrino.resmgr/topic/skeleton_SIMPLE_ST_EG.html);//+
* [регистрируется](http://www.qnx.com/developers/docs/7.0.0/#com.qnx.doc.neutrino.lib_ref/topic/r/resmgr_attach.html) и функционирует с использованием библиотеки resmgr\_library ОС QNX Neutrino, точка монтирования "/dev/cryptobbs”;// +
* реализует обработку клиентского запроса с помощью функции [io\_devctl](https://it.wikireading.ru/2461)(), передаваемую библиотеке io\_funcs через соответствующую структуру указателей в скелете; //+
* работает по описаному в п.2 Протоколу, использует файл bbs.h;//+
* извлекает структуру данных из сообщения (для запроса 1 Протокола), производит приведение типа void \* к типу структуры BBSParams с помощью [reinterpret\_cast](https://ru.cppreference.com/w/cpp/language/reinterpret_cast);
* валидирует данные, в случае ошибки передаёт её клиенту;
* по запросу 2 Протокола генерирует очередной элемент ПСП с помощью алгоритма BBS согласно параметрам, переданным в запросе 1 и отправляет клиенту.

*Вопрос. Какой механизм IPC лежит в основе функционирования АР?*

*Вопрос. Можно ли запустить два процесса АР?*

*Вопрос. Каков недостаток предложенного АР при работе с несколькими клиентами? Как его можно исправить?*

5. Тестирование

1) Запуск АР.

2) Запуск клиента.

3) Завершение клиента по SIGINT.

4) Проверка вывода клиента.