**Назначение**

Программное средство «One Time Pad» предназначено для многопоточного шифрования открытого текста одноразовым блокнотом ([шифр Вернама](https://ru.wikipedia.org/wiki/Шифр_Вернама)), генерируемым с помощью линейного конгруэнтого ГПСЧ.

**Аппарат**

Реализовать ЛКГ — линейный конгруэнтный генератор псевдослучайных чисел (ПСЧ), производящий вычисление псевдослучайной последовательности (ПСП) по реккурентной формуле:



См. [https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9\_%D0%BA%D0%BE%D0%BD%D0%B3%D1%80%D1%83%D1%8D%D0%BD%D1%82%D0%BD%D1%8B%D0%B9\_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4](https://ru.wikipedia.org/wiki/Линейный_конгруэнтный_метод)

Несколько слов о шифре Вернама:

<https://www.cryptomuseum.com/crypto/otp/index.htm>

**Требования**

* платформа запуска — любая ОС с поддержкой POSIX API (GNU\Linux, QNX Neutrino 6.x, Windows с mingw);
* платформа разработки — IDE QNX Momentics (в случае использования QNX Neutrino);
* для работы с файлами и потоками использовать POSIX API;
* ЛКГ реализуется в отдельном потоке;
* для работы с аргументами терминала использовать функцию [getopt()](http://www.qnx.com/developers/docs/7.0.0/#com.qnx.doc.neutrino.lib_ref/topic/g/getopt.html) системного API;
* реализовать обработку базовых ошибок и исключительных ситуаций;
* шифрование осуществляется в N рабочих потоках (воркерах);
* число воркеров N = количеству ядер ЦП;
* воркеры и главный поток синхронизируются с помощью [барьера](http://www.qnx.com/developers/docs/7.0.0/index.html#com.qnx.doc.neutrino.sys_arch/topic/kernel_Barriers.html) POSIX;
* количество ядер ЦП получать программно, с помощью специальной функции.

**Входные данные**

* путь к файлу с открытым текстом;
* параметры ЛКГ;
* путь к выходному файлу.

**Вид строки запуска**

otp  *-i /path/to/text.txt* -o -*/path/to/cypher.txt -x 4212 -a 84589 -c 45989 -m 217728*

**Выходные данные**

* шифротекст, сохранённый в указанном файле.

**Алгоритм**

1. Прочитать параметры командной строки, распаковать их в структуру.
2. Прочитать файл c открытым текстом в бинарном виде, отобразить в оперативную память. Получить размер файла. Предусмотреть ограничение на размер файла.
3. Создать ПСП по прочитанным параметрам с помощью ЛКГ в отдельном потоке pthread API.
4. Синхронизировать основной поток с рабочим с помощью [присоединения](http://www.qnx.com/developers/docs/6.5.0/index.jsp?topic=%2Fcom.qnx.doc.neutrino_lib_ref%2Fp%2Fpthread_join.html).
5. [Создать барьер](http://www.qnx.com/developers/docs/7.0.0/index.html#com.qnx.doc.neutrino.lib_ref/topic/p/pthread_barrier_init.html).
6. Задекларировать структуру — контекст, содержащую барьер, входные данные для каждого воркера (фрагменты блокнота и открытого текста), а также предусматривающую получение выходных данных от воркера.
7. Создать N воркеров с помощью функции pthread\_create(), передав каждому экземпляр контекста.
8. Главный поток [блокируется по ожиданию барьера](http://www.qnx.com/developers/docs/7.0.0/index.html#com.qnx.doc.neutrino.lib_ref/topic/p/pthread_barrier_wait.html).
9. Каждый воркер производит побитовое сложение по модулю 2 своих фрагментов блокнота и текста и [блокируется по ожиданию барьера](http://www.qnx.com/developers/docs/7.0.0/index.html#com.qnx.doc.neutrino.lib_ref/topic/p/pthread_barrier_wait.html).
10. Когда счётчик n барьера становится равен N + 1 (n = N + 1), главный поток разблокируется и продолжает работу.
11. Главный поток объединяет и сохраняет данные в выходной файл.
12. [Барьер уничтожается](http://www.qnx.com/developers/docs/7.0.0/index.html#com.qnx.doc.neutrino.lib_ref/topic/p/pthread_barrier_destroy.html).

**Тест**

* запустить OTP, зашифровать текст, получить шифротекст;
* запустить OTP ещё раз, на вход в качестве текста подать шифротекст, получить открытый текст;
* проверить равенство файлов с помощью [diff](https://www.qnx.com/developers/docs/6.3.0SP3/neutrino/utilities/d/diff.html) (или контрольных сумм CRC32).

**Критерии оценки**

1. Проверка корректности работы программы с помощью автотеста.
2. Проверка соответствия исходного кода программы Требованиям даного задания.
3. Реализация опционального пункта «Дополнительно».

**Дополнительно**

Измерить продолжительность каждого этапа работы программы otp. Оценить порядок издержек на инициализацию механизмов многопоточности. Оценить предел величины входной последовательности, после которой эффективность от многопоточного выполнения превосходит эти издержки.