Korni

Status: proof of concept; WIP; usable for fun ; unstable

Русский RU	English EN
Проект korni (рабочее название) .... ("не забывай свои корни") - синхронизируемое файловое хранилище с идеологией *nix & git Может быть использовано для построения не цензурируемых распределенных приложений типа ZeroNet. Не блокчейн (пока). Интернет не обязателен, возможна длительная офлайн работа.	rus "Korni" = en "roots" \ "Don't forget you roots". File storage and database like git but other purpose with *nix ideoms. Can be used for make **private group** distributed not moderated application and platforms like ZeroNet Not blockchain (at this moment) Can work offline without internet
Утилита строит из шифрованных файлов базу данных для приложений. Задача синхронизации файлов решается другими инструментами. Команды дописывают файлы шифрованными данными, файлы синхронизируются с другими участниками сети - данные в БД таким образом синхронизируются. Есть контроль доступа.	Utility build from crypted syncable files local database for some application. File syncronization task must to perform by external indendendent tools. "commands" append files in work folder by crypted blocks and changed files sync with other users - this update data in user databases. System has the permition control and private secret conmmunication zones with separeted secret keys

Для кого

Такое решение очень хорошо для IoT т.к.

1. может работать даже без интернета - оффлайн,
2. файлы данные можно синхронизировать по разным видам связи, по радио в том числе, по wifi , провода, модемы, ftp, флеш карты ... да что угодно ...
3. от устройства не требуется ничего сверхестесственного - нужно умение писать файлы, ядро linux (желательно)
4. масштаб системы может быть очень большим - на сколько я знаю, ограничения на размер файловой системы - самые условные.
5. легко организовать поэтапную синхронизацию с несколькими центрами - мульти-датацентр
6. безопасность. синхронизацию файлов могут выполнять разные люди , но прочитать зашифрованные данные может только тот у кого есть secret.
7. двойное назначение. что именно содержится среди нераспознанных блоков данных тоже не ясно - это могут быть "не бесполезные данные".

8. git версионизирование. для многих промышленных и учетных применений версионирование просто необходимо. например чтобы сбор и обработка данных шли своим чередом, а их бухгалтерский учет пересчитывался одномоментно по команде

Такое решение хорошо подходит для обмена знаниями, учебными материалами, документами, обсуждения (чаты, форумы, переписка), материалы информационных сайтов. т.к.

1. децентрализация и управляемый доступ
2. оффлайн доступ - даже в глухих деревнях и на трассах решение будет работать, синхронизируясь при наличии сети.
3. Доступность уже полученных материалов - вечная (достаточно сделать fork и даже если потом доступ заберут - будет ветка в которой он есть том состоянии в котором был получен когда-то)

Такое решение хорошо подходит для торговых решений т.к.

1. без остановки работы системы есть возможность анализа всех материалов на "своей машине" любыми индексаторами и анализаторами
2. парсинг как таковой не нужен - есть доступ в БД. представьте, что подключаясь к, например, avito, вся бд авито у вас на компе есть и вы там можете фильтровать своими скриптами как вам нужно и строить такие графики, которые сам авито не позволяет строить.

Такое решение хорошо подходит для социальных взаимодействий т.к.

1. возможно построение честных голосований
    1. с официальным ключем - открытые, с юридически значимых результатов,например для коллективных судов по защите ваших прав
    2. тайных, в том числе референдумов.
2. можно создать "анонимного абонента" и работать от его имени
3. можно зайти в систему "по паспорту" и говорить от своего имени.
4. прозрачность алгоритмов рекомендаций
5. анализ хронологии социального топика.
6. полный и развиваемый поиск по всем материалам.

Начать пользоваться

- [x] инициализация системы, настройка,
- [x] создаем ключи (openssl)
- [x] добавляем ключи в систему (аналог login)
- [ ] управляем данными

- [x] создание объектов
- [x] изменение объектов
- [x] получение объектов, чтение
- [ ] управление правами на объекты (не реализовано)
- [ ] git логика конфликты (не реализовано)

- [ ] рекомендации по синхронизации файлов между пользователями (будет отдельная статья)
  - [ ] известные способы (не описано)
  - [ ] новые перспективные способы (не описано)

Tutorial

Executable file `korni` are small binary less than 50 KB.

```
$ ll korni
-rwxrwxr-x 1 mp mp 37520 map 12 19:45 korni*
$ ./korni help
```

```
mp@mp-strela:~/MY/Korni$ ./korni help
ERROR when parse params
usage: korni <options> command [commandParams ...]

options:
---------

commands:
---------

key add mycert.pem mykey.pem     - add new THIS user key
secret add <secret-password>     - add secret to known
select topic <topic id>          - select known topic for data
                                 work info saved in ./.korniTopic
J new                            - create new journal object .
                                 return jid of new created object
J show asList <jid>              - show object as list
J show asMap [<jid>]             - show object as map
dbconsole|dbConsole|db_console|db - interactive for db stdin
J set <jid> <key> <bin value> - write value to journal
J set <jid> <key>                - write value to journal
                                 but read std input bin
J permGet <jid>                  - read permition of object.
J permAdd <jid> <to> [S|R|D|A]   - grant permition to other user <to>
                                 S-set|R-Replace|D-delete|A - admin
J permRemove <jid> [G|S|D|A]     - remove permition for SELFT for jid
file|File|FILE <PATH>            - scan file for operation blocks

installDepends
        prepare system for working . install some soft from repos
        create .config folder and structure
```

gen YOU keys and cert (self signed by openssl) - standart step for

Как сгенерировать самоподписной сертификат?

Создать файл mycert.pem, в котором будет и секретный ключ и открытый сертификат, основанный на нём. Сертификат будет действителен в течение 365 дней; ключ (благодаря опции `-nodes`) будет нешифрованным.

```
# -days 365
openssl req \
    -x509 -nodes  \
    -newkey rsa:1024 -keyout mykey.pem -out mycert.pem
```

После вызова команды надо будет ответить на несколько вопросов: Country Name, State, City и так далее. На вопрос “Common Name” нужно отвечать именем сервера, по которому будут обращаться люди.

Можно автоматизировать ввод ответов с помощью опции `-subj`. **ЛУЧШЕ** сгенерируйте себе несколько сертификатов для разных целей

```
openssl req \
  -x509 -nodes -days 365 \
  -subj '/C=RU/ST=Yar/L=Ryb/CN=mapavlov.ru' \
  -newkey rsa:1024 -keyout mykey.pem -out mycert.pem
```

out Файл может быть один и тот же (допишется, а не затрется)

Как сгенерировать хороший секретный пароль secret

Как сгенерировать хэш в стиле crypt?

Сгенерировать хэш очень просто:

```
$ openssl passwd MySecret
8E4vqBR4UOYF.
```

Если salt для пароля известен, можно воссоздать хэш.

```
$ openssl passwd -salt 8E MySecret
8E4vqBR4UOYF.
```

Как сгенерировать хэш пароля в стиле shadow?

В новых UNIX/Linux-системах вместо старого crypt-хэша используется новый, более стойкий хэш MD5. Его генерирование выполняется с помощью ключа -1:

```
$ openssl passwd -1 MySecret
$1$sXiKzkus$haDZ9JpVrRHBznY5OxB82.
```

В этом формате salt состоит из 8 символов; он находится между вторым и третьим знаком $, в данном случае это sXiKzkus.

```
$ openssl passwd -1 -salt sXiKzkus MySecret
$1$sXiKzkus$haDZ9JpVrRHBznY5OxB82.

# сохранить новый пароль
$ openssl passwd -1 MySecret > test.secret
```

В общем, генерируете пары ключей и Вы можете генерировать секреты по разному или придумать их из головы для вашего приложения.

Если вам надо подключиться к существоующему комюнити - вам нужен их пароль. его надо где раздобыть "по знакомству" и прочим каналам связи.

Настройка korni для работы

Настройка системы , установка пакетов и настройка структуры БД

```
$ ./korni installDepends
```

> приётся вводить пароль для sudo т.к. выполняется `apt-get install`
>
> если не хотите , то установите всё вручную :
>
> ```
> sudo apt-get install -y awk pcregrep grep openssl jq sqlite3 # потом это будет легче
> cat schema.db.sqlite.sql | sqlite3 ~/.config/korni/db.sqlite
> mkdir -p ~/.config/korni/  ~/.config/korni/work/
> ```

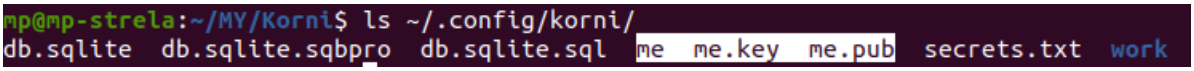
тут создаётся папка для файлов - в ней буду постеперннo появлятся файлы с шифрованными блоками

добавляем ключи

```
$ ll my*
-rw-rw-r-- 1 mp mp 875 map  4 12:20 mycert.pem
-rw------- 1 mp mp 916 map  4 12:20 mykey.pem
$ ./korni key add mycert.pem mykey.pem
Error loading file key
ERROR: cert are bad
```

Выполнится выделение публичного ключа из сертификата и проверка подписи (самоподписанный) .

> корни сейчас отвергает ключи подписанные удостоверяющими центрами и поддерживает только ассиметричное шифрование на самоподписанных сертификатах, т.к. **я не доверяю никому кроме себя и вам не советую.** инструмент создан для улучшения свободы и большей скрытности поэтому пока так. (позже возможно я найду хороший способ повышения доверия). Основа доверия в сети - личное знакомство , ответственность, наработанная репутация и проверка ключей и паролей по бумажке в темноте под одеялом :-D.



Далее добавляем секреты. у меня секрет записан в `test.secret`

```
$ ./korni secret add $(cat test.secret)
$ ll ~/.config/korni/secrets.txt
-rw------- 1 mp mp 0 map 12 19:57 /home/mp/.config/korni/secrets.txt
$ ./korni select topic 'testTopic' $(cat test.secret)
$ ll .korniTopic
-r-------- 1 mp mp 45 map 12 20:00 .korniTopic
$ head -n 1  .korniTopic
testTopic
# то там дальше нельзя смотреть - там секрет
```

Итого "Что мы сделали ?"

1. сгенерировали для авторизации пару ключей и самоподписанный сертификат
2. придумали какой - секрет,
3. выбрали и сопоставили имя топика для работу и секрет для него (значение по умолчанию, в будущем эти опции могут забираться не из файла а из окружения)
4. есть готовность выполнять команды утилиты.

Журналы

Базовая сущность - журнал. он может читаться как список и как map объект.

Это J сущность в sqlite таблице, имеет два представления (view) aslist asmap.

её можно читать, пистаь в неё нельзя напрямую - делаем это специальными командами.

команды создают шифрованные блоки в файлах, потом парсят их и добавляют инфо в базу.

Создать журнал

команда придумывает новый журнал с новым случайным идентификатором длиной 48 символов.

```
$
```

Вставка записи с ключём

журнал используется как объект asmap

```
$
```

Отображение журнала как MAP

```
$ ./korni J show asmap DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1','100500','1647069271','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647088608','{"test":','1647088608','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647088617','{"test":','1647088617','S','ccdaff8c28710175ef559f64d12b3e14'
```

```
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089057','{"test":
123}','1647089057','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089076','{"test":
123}','1647089076','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089105','{"test":
123}','1647089106','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089154','{"test":
123}','1647089154','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089180','{"test":
123}','1647089180','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089214','{"test":
123}','1647089214','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089285','{"test":
123}','1647089285','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089306','{"test":
123}','1647089306','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','1647089334','{"test":
123}','1647089334','S','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','2','579','1647085579','S','ccdaff8c28710175ef
559f64d12b3e14'
```

Порядок столбцов следующий:

1. `jid` journal id
2. `key`
3. `value`
4. `time` = system `time %s` секунд с 1970 года...
5. `command` = S|R Set|Replace
6. `author`

Выдаются для мапы не все данные, а только значение и момент времени для каждого ключа с последней датой (если их было для ключа несколько т.е. если ключ перезаписывали)

Вставка записи в журнал (key=current time)

```
$ ./korni J  insert DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V  '{"test": 123}'
```

Журнал здесь используется в смысле списка \ aslist.

Внутрь файла дописывается блок примерно такого вида (одна "почти" base64 (в стиле base64, реально - нет) строка без разрывов и переносов):

```
MPU2FsdGVkX1/JzHuuLYeSDAswJEEQAqL9OInOO2WT3UOEtabXq+P1urqvSL1aUPr+JjPnLA+IT7QLk4s
+i/1oe03L7Yoq6ttVHVvisE9tuYPdIwHO1wU8B0rW1TS5zp8wMKYU8/pAHUR3BvTL4nO2OFDzL1eWdoyQ
J/dm8WDHwRJ89TWlCFq4kUT9x+4peKy2DACROHTumgR3Wh2EIwXZ+bA+tNgaz+k/4E1qEIp6ikpQPxo1A
Sv4TgDC2lPC4zc+QUX9lUWtX/T78/iODhakySW0jDqD/DPmtlHDjSE8kG782ZEenQQdZ6/ozMBtAlCzHW
vIvLCwsawcKhDcKdfjk8UPXIIL2NLRDAWdG0Lc9n82jILSmzlHcKJfGxVins5O/JHbLb9FkxKUv3YWCR5
oZFFUZKfp2cH9dlUxFrxus6L8Zb+zlCPj0r3avZcYZGJBi/E1ZdCWe26Lw9dW2fZnHgJMR9pozyytOOzK
qIBMdYrHYQTfQzgq8nXU0br9hXOpAeyEwWEnH/15fEyQackmHPTJTXNuOdt3EMnmBO1OSdAV5hAMYLB2p
A0+FnyefsO+NS
```

Сканировать файл, мониторить изменения

можно сканировать выборочно один файл и это и правильно в принципе, если изменяемые файлы получать через "inotify" ведь меняются они при синхронизации постепенно.

файл передаётся ЛИБО параметром

```
$ ./korni  file
'/home/mp/.config/korni/work/99/f0/cb/d5/7d3/1428f8ac4ab6c6bed9c57'
```

ЛИБО утилита может из stdin их читать

```
$ ./korni file
<WAIT STDIN>
^Ctrl+C

$ sudo apt install -y inotify-tools
<PASS>
# wait file changes in folder and print filenames that changed
$ inotifywait -m -q -r --format '%T % e %w%f'  --excludei '/trash/' --timefmt
'%s' -e MOVED_TO -e modify . | awk '{print $3}'

# handle it by korni
$ inotifywait -m -q -r --format '%T % e %w%f'  --excludei '/trash/' --timefmt
'%s' -e MOVED_TO -e modify '~/.config/korni/work/' | awk '{print $3}' | ./korni
file
```

В результате последней команды запускается конвейер который следит за рабочей папкой korni и если в ней меняются файлы - пересканирует их. Можно запустить конвейер в фоне `... &`. и перезапускать в цикле если упал - это уже задача администрирования. Обычно утилита синхронизации "пишет" обновленные файлы и скачанные вновь файлы - эта олпция предназначена чтобы эту информацию воспринимать и использовать.

Если в файле удалось найти зашифрованные блоки (т.е. для этих блоков вам ключ известен), то они расшифровываются, проверяются подписи. если структура в порядке - они добавляются в БД.

Далее БД переиндексруется - среди блоков находим такие которе еще не применены, но в соответствии с системой прав могут быть применены. Из блоков создаются журналы и модцифицирующие их команды (если пройдена проверка прав).

В рузультате БД приобретает такой вид, что представления aslist и asmap показывают актуальную структуру.

> утилита пытается Все найденные корректные блоки из файла проиндексировать через БД. Это может не удаваться с существующими уже в БД блоками из органичений первичных ключей (блок повторно найден, возможно в других файлах). попытка заранее определить, что блок уже в бд имеется не предпринимается. просто пытамся вставить, если уже есть - ок, работаем дальше. Анализ состава и допустимости блока происходит на следующих этапах. данная процедура просто пытается блоки найти , расшифровать, проверить подписи, проверить структуру, автора и разегистрировать блок в БД.

Отображение журнала как списка, `raw sql` работа напрямую с базой

Со списками приложению (бекенду приложения) удобнее работать напрямую отправляя запросы в БД. т.к. данные приложения могут иметь сложную структуру и все ситуации трудно предусмотреть, то имеется интефейс sql

```
$ ./korni db # posible: dbconsole | dbConsole | db_console | db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> select * from aslist limit 5;
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','S','100500','1647069271','ccdaff8c28710175ef5
59f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','S','{"test":','1647088608','ccdaff8c28710175e
f559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','S','{"test":','1647088617','ccdaff8c28710175e
f559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','S','{"test":
123}','1647089057','ccdaff8c28710175ef559f64d12b3e14'
'DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V','S','{"test":
123}','1647089076','ccdaff8c28710175ef559f64d12b3e14'

sqlite> .mode line
sqlite> select * from aslist limit 3;
   jid = DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
     C = S
 value = 100500
  time = 1647069271
author = ccdaff8c28710175ef559f64d12b3e14

   jid = DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
     C = S
 value = {"test":
  time = 1647088608
author = ccdaff8c28710175ef559f64d12b3e14

   jid = DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
     C = S
 value = {"test":
  time = 1647088617
author = ccdaff8c28710175ef559f64d12b3e14
```

чтобы получить больше информации о полной структуре данных в списке, если считать значения не bin json то можно использоватьтакой "трюк" с `json_tree` :

```
$ ./korni db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> .mode line
sqlite> select *
   ...> from aslist , json_tree(aslist.value) as V
   ...> where json_valid(aslist.value) AND "$.test"=fullkey ;
    jid = DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
      C = S
  value = {"test": 123}
   time = 1647089057
 author = ccdaff8c28710175ef559f64d12b3e14
```

```
    key = test
  value = 123
   type = integer
   atom = 123
     id = 2
 parent = 0
fullkey = $.test
   path = $

    jid = DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
      C = S
  value = {"test": 123}
   time = 1647089076
 author = ccdaff8c28710175ef559f64d12b3e14
    key = test
  value = 123
   type = integer
   atom = 123
     id = 2
 parent = 0
fullkey = $.test
   path = $

    jid = DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V
      C = S
  value = {"test": 123}
   time = 1647089106
 author = ccdaff8c28710175ef559f64d12b3e14
    key = test
  value = 123
   type = integer
   atom = 123
     id = 2
 parent = 0
fullkey = $.test
   path = $
..... etc
```

- Таким образом можно отфильтровать любой путь внутри json. Конечно интерактивный вариант не обязателен.
- sql script можно отправить утилите на стандартный вход.
- удобно "интроспектировать" БД ,
- можно создать индексы для ускорения приложения, свои представления и таблицы + триггеры, которые их пополняют данными по мере изменения журналов.

"технические таблицы" легко распознать, они уже "обложены индексами" и инкрементальное переиндексирование новых данных занимает десятки миллисекунд (надеюсь в будущем оно вырастет не значительно).

Модули расширения sqlite

sqlite не так слаб как кажется. SQLITE может тесно интегрироваться с "shared lib", которые расширят его новыми функциями, так , например , используя его из python можно "нативно, а не через сетевой протокол" траивать в БД функции на питоне и это будет работать очень и очень быство т.к. это связывание внутри одного процесса.

- python create function
- perl create function link
- php create function
- lua sqlite link

Он интегрирован со "всеми языками" , есть гибкие ORM ...

узнать с чем собран в комплекте ваш sqlite: `sqlite> PRAGMA compile_options;`

- встроенное расширения для полнотекстового поиска
- даже классикаторы и train встраивают
- https://github.com/planetopendata/awesome-sqlite
- https://github.com/nalgeon/sqlean/
    - тут описаны очень мощные расширения, которые легкие и легко могут расширить возможности вашего приложения и бекенда. комбайны просто о которых в "больших бд" и мечтать никак.

> **Идея** чисто технически у меня "бродят мысли" сделать всю эту систему расширением для sqlite - на плечах гиганта легче будет ее распространять, но это не главное. важно, что такая тесная интеграция позволит мнигие процессы спрятать и упростить работу юзеру.
>
> > Под приложение можно будет спроектировать любую БД и выгрузить в виде крипто файлов и обновить из папки с такими файлами - очень удобно. - предложите use case with commandline examples!

Идеи для next iteration

1. прописать варианты для `sqlite extention` как именно это будет в реальности?
2. переделать конвейер обработки блоков из файлов и сканирования файлов (сейчас там есть "технический долг" и не слишком то оптимально получилось + возможны зависания т.е. надо за процессом присматривать).
3. journal as list , journal as map не слишком удобная концепция. надо что-то типа redis data model
    1. **???** модель должна быть relation table ? не нахожу ничего лучше. список записей с несколькими полями.
    2. вариант api, PK всегда поле id, например:
        1. table.insert(record)
        2. table.update(id, recordPatch)
        3. table.delete(id)
4. подумать как сделать для "любой базы данных" (разные схемы) аналогичное решение
5. сделать proof of concept with "git logic".
6. **Идея на подумать, обсудить.** Может стоит убрать шифрование из самого korni и перенести его на момент синхронизации файлов а файлы в папке будут открытые ?
7. Идея

Прогресс "Готово"

- [x] кодировать по другому команды `j new` - чтобы в файл писались только binB данные,
  - [x] [jitFomUser]
- [x] добавлять в колменды время
- [x] парсинг данных, id блока = sha256 а не bin
- [x] переделать формат общения между awk и bash на FS="\n" RS="" когда расшифровываем
- [x] передавать данные о блоке : file , ~~topic * secret~~ когда индексируем , передавать время когда индексируем блоки
- [x] {"bin", "sign"} from block in file TODO - **just bin**
- [x] encode
  - [x] base64(command json with sorted keys) as blockB64 + base64(sign(blockB64)) as signBase64 -> (blockB64signBase64) as packet
  - [x] base64(crypt(secret(topic), packet)) as binBase64 -> **bin**
  - [x] append to file
- [x] decode
  - [x] bin .... try decript with secrets -> topic|secret, blockB64, signB64
  - [x] вывести в sql виде для импорта в БД `.mode line
  - [x] apply objects - enable objects in DB , set time

Готовность 08032022

- [x] parse params
- [x] `korni installDepends`
  - [x] db schema
- [x] `korni key add mycert.pem mykey.pem`
- [x] `korni secret add <big-secret-password>`
- [x] `korni select topic <secret> <topic id>`
  - [x] save tmp to local file
  - [ ] ~~[save tmp to export env var]~~ отказался - родительскому процессу не установить окружение
  - [ ] ~~[make it as option]~~
- [x] **JOB parse file , find blocks, decrypt, check sign,**
  - [x] read file , get json from it
  - [x] handle line by line fast
    - [x] redirect json(binBase64, signBase64) -> awk
    - [x] awk for each line print(bin, sign, decriptedBin)
      - [x] awk make sub process for bin -> korni tool1 -> decripted bin + success flag
      - [x] awk communicate with it
      - [x] print only if decripted !
    - [x] pipe to new awk (decriptedBin, bin, sign) - сделано по другому

- [x] coomunicate for check sign Internal Json -> korni tool2 -> ok | error
- [x] if verified - that print(decriptedBin, bin, sign) , nothing if not
- [ ] ~~pipe to new awk that tooling with korni~~ - сделано по другому
- [x] korni 3 save JSON to DB for indexing
- [x] [relative path to DB] ~/.config not work VS /hpme/mp/.config - are working

- [x] index it in DB

- [ ] check and ~~verify~~ command, apply command to database
  - [x] decript not decripted blocks - set decripted flag to block
  - [x] check sign if posible - set sign verifed flag to block
  - [x] if sign verifed: extract objects from BLOCK - index J , P, C objects
- [x] ~ `korni J new`
  - [ ] print `jid` - journal id of created object
- [x] (simple create record) `korni J set <jid> <key> <bin|str|value>`
  - `key` important for view `asMap` . it show only last value for each key
- [x] (simple create record) `korni J insert <jid> value`
  - key here exists too, but it eq time of operation
- [x] (simple read DB sql) `korni J show asList <jid>` - show all actual (not deleted) records of journal

```
/* получить только ключи json значений по пути .test
 * для всех jid
 */
select *
from aslist , json_tree(aslist.value) as V
where json_valid(aslist.value) AND "$.test"=fullkey
;
```

	jid	c	value	time	author	key	value	type	atom	id	parent	fullkey	path
1	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089057	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
2	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089076	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
3	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089106	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
4	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089154	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
5	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089180	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
6	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089214	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
7	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089285	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
8	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089306	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$
9	DZo2ev2zBQBpbgCOb2ymW/3KdDH31/0V	S	{"test": 123}	1647089334	ccdaff8c28710175ef559f64d12b3e14	test	123	integer	123	2	0	$.test	$

- [x] (simple read DB sql) `korni J show asMap <jid>`
- [x] embed sql create db to script
- [x] `korni dbconsole|dbConsole|db_console|db` interactive for db stdin
- [x] внутри блока есть jid + uJid , в таблице есть id блока. это проблема? **нет**
- [x] выбираем и дописываем один и тот же файл , по возможности, пока его размер не превысил разумный порог
- [x] файлы не растут более 39МБ иначе - придумывается новое имя файла и пишем в него

~~Создаём ключи для своего топика~~ пока не надо, не уверен что нужно (аргументы?)

Генерируем пару ключей, как пользователя, сертификат и ключ.

Запускам выбор топика.

> Топик выбирается для каждой текущей директории и секрета (внутри секрета есть топики, внутри топика - журналы = объекты с правами доступа, объекты и команды изменения подписываются автором). Поэтому чтобы получить доступ к данным нужно иметь правильный секрет, и для топиков унутри иметь пары ключей , чтобы шифровать и расшифровывать в нем данные (все "заговорщики" имеют эти пары ключей).
>
> Все авторы это самоподписанные сертификаты by design. все анонимно и свободно от проверок.

```
$ openssl req \
>   -x509 -nodes -days 365 \
>   -subj '/C=RU/ST=Yar/L=Ryb/CN=mapavlov.ru' \
>   -newkey rsa:1024 -keyout testTopicKey.pem -out testTopicCert.pem
```

сделали ключ, будем работать в рамках секрета который в test.secret :

```
$ korni select topic testTopic $(cat test.secret) testTopicCert.pem
testTopicKey.pem
$ cat ./.korniTopic
.... данные
```

Прогресс "НЕ готово"

- [ ] ~~сохранять токены и пароли так, чтобы оба они хранились~~
- [ ] разбор блоков через git ? **или применение журналов через git ?**
  - [ ] создать файл для каждого журнала
  - [ ] создать ветку korni для топика topic
  - [ ] **применение операции в БД**
  - [ ] экспорт состояния журнала в файл - перезаписать
  - [ ] git add; git commit to topic ;
- [ ] [сделать difficult] - подбор сложного хеша чтобы не так то просто было добавлять данные если установлены требования сложности - защита от спама
- [ ] [opts] екоорые опции для команд - вариативность поведения
- [ ] [add topic to _blocks] optional waiting чтобы приложениям легче был оориентироваться среди данных нескольких приложений
- [ ] [add file to _blocks]

- [ ] check permitions apply objects - права проверять не только [S,R] но и ... вообще надо написать тесты для покрытия прав - ответственно протестировать
- [ ] Операции с правами
  - [ ] (simple create record) `korni J permAdd <jid> <target user> [S|R|D|A]` (**S**et new key, **R**eplace exist key, **D**elete key, **A**dmin can change access perm)
  - [ ] (simple create record) `korni J permRemove <jid> [G|S|D|A]` user can change only SELF Permition
- [ ] (simple read DB sql) `korni J permGet <jid>`

Критические Ошибки

- [ ] `j new` перестал работать т.к. **parseFile перестал работать - tool1 problem**. т.е. parse файл придется запустить отдельно
- [ ] `J new` не печатает id нового журнала, но в бд он появляется с текущим time и author==me
- [ ] `key add` не сработал
- [ ] `secret add` не сработал возможно я сам это заменил на `korni select topic` тогда и документе это надо заменить
- [ ]