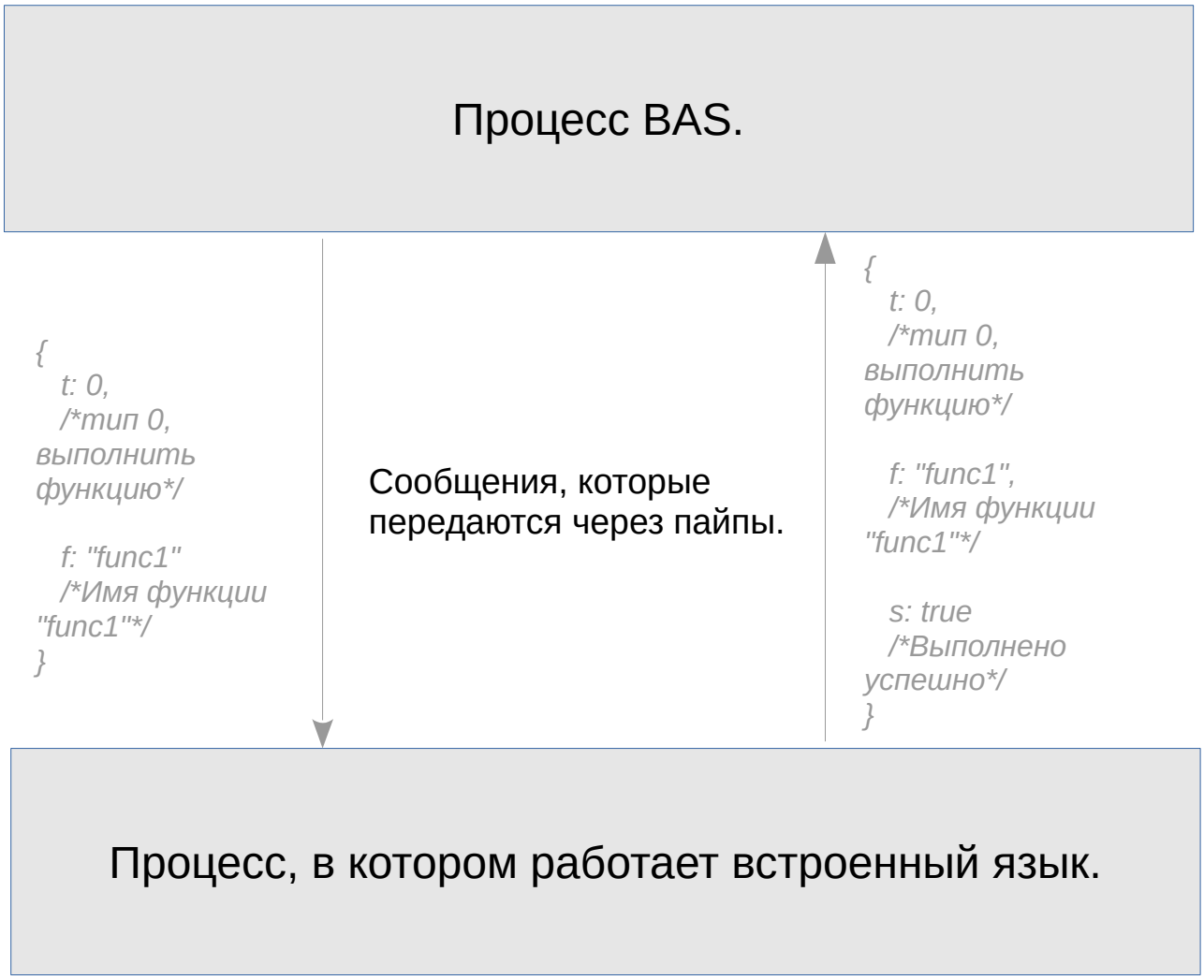
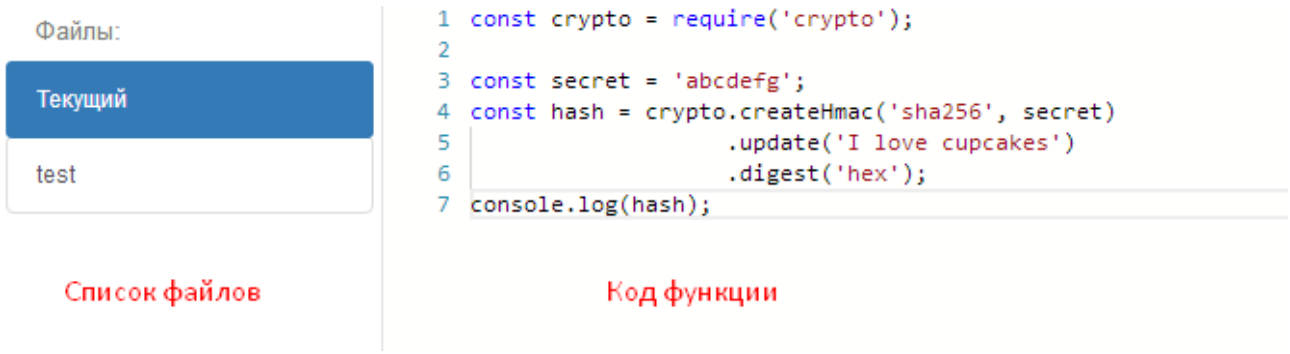
Механизм работы встроенных языков в BAS.

Код на встроенном языке никогда не выполняется в контексте BAS. Такая архитектура позволяет избавиться от проблем со стабильностью в случаях если внешний код работает некорректно. Вместо этого код на встроенном языке выполняется в отдельном процессе и обменивается сообщениями с процессом BAS. В качестве метода коммуникации выбраны windows pipes, в качестве формата сообщений - обычный json.



Любой вызов кода на встроенном языке называется функцией. BAS устанавливает название функций самостоятельно, а встроенный язык получает список функций перед стартом и должен уметь выполнить любую ранее установленную функцию по запросу BAS во время работы. Для интерпретируемых языков установка функций может состоять только из создания соответствующих файлов, тогда как для компилируемых нужна их компиляция.

Встроенные языки должны также поддерживать файлы. Файлы отличаются от функций только тем, что могут быть использованы сколько угодно раз в разных функциях.



Перед запуском BAS передает встроенному языку список модулей. Встроенный язык обязан их установить или выдать ошибку. Модули - это внешние библиотеки, который использует язык. Большинство языков обладают системами управления пакетами(npm для node.js, gems для ruby, и т. д.), необходимо использовать их для установки с модулей.

Для того, чтобы добавить новый встроенный язык в BAS необходимо указать:

1) Название языка(только латинские символы).
Пример: Node

2) Версию языка(строка, которая может содержать только цифры или точки).
Пример: 8.6.0

3) Метод, которым можно искать существующие модули по имени.
Пример: Сделать запрос к урл https://registry.npmjs.org/-/v1/search?text=<<имя модуля или его часть>>&size=20.

4) Метод, которым можно добавлять функции.
Пример: Создать файл /distr/lib/custom/<<имя фуннкции>>.js содержимое которого идентично коду функции.

5) Метод, которым можно добавлять файлы.
Пример: Создать файл /distr/lib/<<имя файла>>.js содержимое которого идентично коду файла одернутому в код module.exports = async function(){<<код файла>>}.

6) Последовательность действий необходимую для установки модулей.
Пример: Выполнить /distr/node_modules/npm/bin/npm-cli.js install

7) Последовательность действий необходимую для запуска процесса, в котором работает встроенный язык.
Пример: Выполнить /distr/node.exe /distr/lib/main.js

Пункты 4), 5), 6) выполняется каждый раз перед запуском процесса(пункт 7)

8) Также неодходимо создать 2 архива, один для 32 битных систем, один для 64 битных. Эти архивы должны содержать все необходимое для запуска процесса со встроенным языком. Сюда должен входить сам встроенный язык, компилятор для него если такой есть и сервер, который будет обмениваться сообщениями с BAS.
Пример: https://bablosoft.com/distr/Embedded/Node/8.6.0/distr.x86.zip https://bablosoft.com/distr/Embedded/Node/8.6.0/distr.x64.zip

9) Архивы из пункта 8 не должны иметь внешних зависимостей. Запуск процесса со встроенным языком должен быть возможен на виртуальной машине с только что установленной операционной системой. Если без установки внешних компонент нельзя обойтись, то это должно быть оговорено отдельно.

10) Описать способ, каким код BAS может выполнен из встроенного языка. В коде функции должен быть доступен для вызова метод, который вызывает код BAS.
Пример: await BAS_API("VAR_NEW_VARIABLE = \"aaa\"");

Возможная реализация:

В архив помещается дистрибутив языка вместе с приложением написанным на этом языке. Это приложение присоединяется к пайп серверу открытому BAS и реализует протокол описанный ниже. Чтобы лучше понять, как работает протокол можно запустить BAS, выполнить функцию на встроенном языке и посмотреть содержимое файла C:\Users\ %USERNAME%\AppData\Roaming\BrowserAutomationStudio\apps\20.9.2\node_log.txt

Запуск, установка соединения:

1) BAS запускает пайп сервер по адресу \\.\pipe\basembeddedpipes<<название языка>><<версия языка>><<идентификатор процесса BAS>>
Пример: \\.\pipe\basembeddedpipesNode8.6.01234

2) BAS подготавливает функции, файлы и модули согласно пунктам 4), 5), 6) и запускает процесс со встроенным языком согласно пункту 7)

3) Процесс со встроенным языком должен получить идентификатор процесса BAS как аргумент коммандной строки.

4) Процесс со встроенным языком должен присоедениться к пайп серверу.

5) Если пайп сервер перестает работать, то процесс со встроенным кодом должен завершиться. Это нужно чтобы процесс не продолжал работать после завершения работы BAS.

Обмен сообщениями:

Все сообщения должны быть в формате json. К каждому входящему и исходящему сообщению должна добавляется специальная строка --BAS-BOUNDARY--. Это необходимо для того, чтобы идентифицировать сообщение в том случае, если оно было получено не полностью или было получено сразу несколько сообщений. Процесс со встроенным языком должен иметь специальный буфер, куда помещать все весь входящий текст и из которого получать все входящие сообщения. Пример реализации можно посмотреть в /distr/lib/internal/pipes.js
Пример: {"l":"124","id":"3755530799338930720"}--BAS-BOUNDARY--

Выполнение функций:

В случае необходимости выполнения функции, BAS отправляет сообщение с типом 0.
Пример: {
"f": "0mbno8od677pxnrx8z798dunmi",
"id": "3755530799338930720",

```
        "t": 0,
        "v": {
        }
}
```

Параметры сообщения: f, id, t, v. f - название функции, id - id сообщения, используется чтобы процесс смог вернуть ответ, и чтобы идентифицировать к какому именно сообщению возвращается ответ. t - тип сообщения, для вызовов функции всегда 0. v - список переменных и их значений. Будет описано позже.
После того, как функция будет выполнена, процесс должент отправить сообщение о завершении.

Пример:
```
{
        "f":"0mbno8od677pxnrx8z798dunmi",
        "id":"3755530799338930720",
        "t":0,
        "v":{
        },
        "e":"",
        "s":true
}
```

Параметры f, id, t, v. f должны быть точно такими же, как и во входящем сообщении, но в ответном сообщении должно быть еще 2 параметра: e и s. s - true в случае если функция была выполнена успешно, и false, если произошла ошибка. Текст ошибки необходимо поместить в параметр e.

Обмен данными(переменные):

Обмен данными между процессом и BAS осуществляется с помощью переменных. BAS передает переменные, которые использовались в вызываемой функции в параметре v. Вместе с названиями переменных, должны передаваться и их значения.

Пример:
```
{
        "f": "0mbno8od677pxnrx8z798dunmi",
        "id": "3755530799338930720",
        "t": 0,
        "v": {
                "VARIABLE": "value"
        }
}
```

После того, как функция будет выполнена, процесс должен передать в ответном сообщении те же самые переменные, но уже с актуальными значениями.

Пример:
```
{
        "f": "0mbno8od677pxnrx8z798dunmi",
        "id": "3755530799338930720",
        "t": 0,
        "v": {
                "VARIABLE": "changed value"
        }
}
```

На этапе добавления функции(пункт 4)) BAS должен каким-то образом преобразовать код функции так, чтобы в нем было возможно использовать переменные. Для обозначения

переменной BAS использует имя переменной помещенное в двойные квадратные скобки [[VARIABLE]]. Нужно преобразовать код так, чтобы он стал синтаксически правильным на данном языке.

Пример: console.log([[VARIABLE]]) преобразуется в
console.log(BAS_VARS["VARIABLE"])

Но нужно преобразорвать код так, чтобы он не только был синтаксически правильным, но и чтобы значения переменных можно было читать и записывать.

Пример: [[VARIABLE]] = "changed variable" // такой код должен менять значение переменной.

Тип переменной может быть любой который можно сериализировать в json: строка, целое число, булевое значение, массив, ассоциативный массив.

Пример: [[VARIABLE]] = [1,2,3] //создание массива в js, может выглядеть по другому в другом языке. Тогда по завершении функции ответное сообщение должно содержать

```
"v": {
        "VARIABLE": [1,2,3]
}
```

Преждевременное завершение функции:

BAS может запросить преждевременное завершение выполнение функции. Это может произойти в случае остановки скрипта, или если превышено максимальное время ожидания действия. При этом BAS отсылает сообщение с типом 2.

Пример:

```
{
        "id": "6934758713911839001",
         "t": 2
}
```

Процесс обязан завершить выполнение функции с заданным id, но не обязан отсылать ответное сообщение.

Вызов кода BAS из функции:

Согласно пункта 10), процесс должен предоставлять метод, которым можно будет вызвать код BAS.

Пример: Внутри кода сервера может быть определена функция BAS_API, которая отсылает соответствующее сообщение.

После вызова этого метода процесс должен отправить сообщение BAS и дождаться ответа. Метод должен продолжить выполнение только тогда, когда пришел ответ от BAS.

Пример сообщения от процесса к BAS:

```
{
        "f": "0mbno8od677pxnrx8z798dunmi",
        "id": "3755530799338930720",
        "t": 0,
        "v": {
                "VARIABLE": "value"
        },
        "a":"Код BAS"
}
```

Все параметры этого сообщения аналогичны сообщению о запуске функции за исключением параметра a, в нем передается код BAS.

Пример сообщения от BAS к процессу:

```
{
    "f": "0mbno8od677pxnrx8z798dunmi",
    "id": "3755530799338930720",
    "t": 1,
    "v": {
        "VARIABLE": "changed value"
    }
}
```

Тип такого сообщения всегда 1. Как и вызовы функций, вызовы кода BAS обмениваются данными через переменные BAS. Алгоритм отправки переменных точно такой же.

Вывод в лог BAS:

Процесс должен переопределить один или несколько методов, которые часто используются в данном языке для вывода сообщения в лог.

Пример: console.log([[VARIABLE]]) // вывод переменной в лог BAS на node.js

Чтобы сообщить BAS о такой необходимости нужно отправить сообщение с двумя параметрами l и id. l - текст сообщения, id - идентификатор текущей функции.

Пример:

```
{
    "l":"текст сообщения",
    "id":"5631366655624329623"
}
```

Что делать, когда код готов?

Для того, чтобы поделиться изменениями, вам нужно создать репозиторий на github или любом другом хостинге для исходного кода, в файле README указать ответы на 10 пунктов в начале документа. Чтобы ваш код можно было добавить в BAS, он должен иметь лицензию MIT или похожую. Если в вашем коде будут найдены баги, то о них будет сообщаться в разделе issues вашего репозитория на github.