*Архитектура CLI*

Общая архитектура

CLI состоит из трех частей (компонентов): терминального клиента, frontend- и backend- сервисов. Терминальный клиент принимает ввод пользователя (подключенного как локально, так и удаленно, например, через ssh), производит первичную обработку, отправляет обработанный ввод пользователя на frontend-сервис, получает с него результаты дальнейшей обработки ввода пользователя и отображает их либо на стандартном выводе, либо на стандартном выводе ошибок (это определяет frontend-сервис). Взаимодействие между терминальным клиентом и frontend-сервисом осуществляется через сокетное соединение. Для каждого подключающегося пользователя создается свой терминальный клиент. frontend-сервис существует в единственном экземпляре и ведет учет всех подключенных к нему терминальных клиентов (пользовательских сессий). frontend-сервис принимает введенные пользовательские данные от терминального клиента и разбирает (парсит) эти данные в цепочку команд. Команды, которые известны frontend-сервису, выполняются на нем; команды, которые известны backend-сервису, передаются на backend-сервис (все остальное считается ошибкой, которая возвращается на терминальный клиент). frontend-сервис хранит состояние, связанное с каждым подключенным клиентом (клиентскую сессию); при передаче команды на backend-сервис, вместе с командой передается и это состояние. backend-сервис также существует в единственном экземпляре и занимается выполнением команд для взаимодействия с реальным железом (backend-команд). Backend-сервис ничего не знает о пользователях и не хранит никакого состояния, связанного с ними; однако это состояние передается на backend-команду во время ее выполнения.

Функциональность и поведение компонентов CLI:

1) Терминальный клиент:

* Принимает ввод пользователя, позволяет его редактировать.
* Поддерживает историю в том же виде, что и командная оболочка Bash.
* Поддерживает автодополнение (при нажатии на клавишу «Tab»); за данными об автодополнении обращается на frontend-сервис.
* Поддерживает жестко заданный набор команд, выполнение которых происходит на самом терминальном клиенте (на данный момент в такой набор команд входит команда **exit**).
* При нажатии на клавишу «Enter» производит первичную обработку ввода пользователя (очистку от пробельных символов с начала и конца). Если в результате получается пустая строка, то ничего не происходит. Если в результате получается одна из терминальных команд, то происходит ее выполнение на терминале. Иначе, ввод пользователя передается на frontend-сервис.
* После передачи ввода пользователя на frontend-сервис терминальный клиент ожидает ответ с сервера. Этот ответ может состоять как из одного, так и из нескольких сообщений. В зависимости от типа сообщения, терминальный клиент либо выводит данные на стандартный вывод, либо выводит данные на стандартный вывод ошибок, либо получает уведомление о завершении выполнения команды. После получения уведомления о завершении команды терминальный клиент готов к приему дальнейшего ввода пользователя.
* Поведение терминального клиента во время готовности к приему ввода и во время выполнения команды подобно поведению командной оболочки Bash.

2) frontend - сервис:

* Поддержка динамической конфигурации сервиса при запуске: ...

3) backend - сервис

* Поддержка динамической конфигурации сервиса при запуске: ...

Протоколы взаимодействия

Протокол взаимодействия frontend и backend сервисов.

Протокол взаимодействия терминального клиента и frontend-сервиса

Терминальный клиент взаимодействует с frontend - частью сервиса. Сообщения, которыми обмениваются терминальный клиент и frontend-часть являются обычными объектами языка Erlang. На стороне frontend-части для сериализации используется функция **term\_to\_binary/1**, а для десериализации - функция **binary\_to\_term/1**. На стороне терминального клиента для сериализации используется функция **erl\_encode** (из библиотеки **erl\_marshal**), а для десериализации — функция **erl\_decode** (из библиотеки **erl\_marshal**). Протокол взаимодействия выглядит следующим образом (в терминах объектов языка Erlang):

1) Выполнение некоторой команды на сервере:

Запрос на сервер: **{command, CommandLine}**, где **CommandLine** - данные, введенные пользователем, очищенные от пробельных символов с начала и с конца.

Ответ с сервера (один из следующих):

* Данные для стандартного вывода **{command\_out, CommandOutput}**, где **CommandOutput** - данные с сервера, которые терминал выводит на стандартный вывод.
* Данные для стандартного вывода ошибок **{command\_err, CommandError}**, где **CommandError** - данные с сервера, которые терминал выводит на стандартный вывод ошибок.
* Окончание работы команды **{end, PromptStr}**, где **PromptStr** - строка подсказки (prompt).

2) Прерывание выполнения текущей выполняющейся команды:

Запрос на сервер: **{interrupt}**.

Ответ с сервера (один из следующих):

* Если на сервере для данного клиента какая-либо команда выполняется: окончание работы команды **{end, PromptStr}**, где **PromptStr** - строка подсказки (prompt).
* Если на сервере для данного клиента никакой команды не выполняется: ничего.

3) Запрос текущего состояния:

Запрос на сервер: **{current\_state\_request}**.

Ответ с сервера: **{current\_state\_response, Prompt}**, где **Prompt** - текущая строка подсказки (prompt).

4) Запрос списка расширений:

Запрос на сервер: **{****extension\_request, CommandLine}**, где **CommandLine** - данные, введенные пользователем, очищенные от пробельных символов с начала.

Ответ с сервера: **{extension\_response, ExtensionList}**, где **ExtensionList** - список расширений для данных, введенных пользователем.

5) Завершение работы:

Запрос на сервер: **{exit}**.

Ответ с сервера: ничего.

Зависимости

На данный момент для сборки и работы системы существует ряд внешних зависимостей:

* Среда выполнения Erlang версии R16 и выше.
* Библиотека readline версии 6.2 и выше.
* Пакет build-essential.