# Платформа речевого вывода в дистрибутивах ALT Linux на базе сервера “VoiceMan”

Описание архитектуры и руководство по настройке

Михаил Пожидаев
msp@altlinux.org

ALT Linux Team 2008

# Содержание

# 1 Введение

Механизм вывода речи — это один из базовых компонентов среды, позволяющей организовать работу на персональном компьютере с выводом информации только через аудио-подсистему. Пользователь получает текстовые данные в речевом виде. Передаваемая информация должна как можно полнее воспроизводить визуальное содержание экрана. Подобный подход к работе связан с большим числом трудностей. Так, например, объём выводимой информации должен быть оптимально подобран. Её недостаток не позволяет однозначно определить состояние приложения, с которым работает пользователь, а с другой стороны, слишком полный вывод замедляет работу и приводит к тому, что пользователь быстро устаёт. На практике объём выводимой информации подбирается экспериментально. Большое внимание нужно уделять и скорости произношения текста. Слишком быстрый вывод заставляет человека вслушиваться в речь, а очень медленный — замедляет работу на компьютере.

При разработке приложений, оснащённых выводом речи, обычно уделяется большое внимание как можно более полному использованию различных параметров текста. К параметрам речи относят три характеристики:

- высота голоса;
- скорость речи;
- громкость.

Стала традиционной практика выделения заглавных букв несколько более высоким голосом, чем при произношении обычных строчных символов. Также высотой голоса можно различать некоторое форматирование, которое видно визуально на экране. Например, другим тембром голоса можно прочитывать ссылки на веб-страницах или выделять папки в списках файлов при просмотре содержимого директории. Очень качественные речевые синтезаторы имеют возможность менять интонацию конца предложения, если прочитывается строка с восклицательным или вопросительным знаков в конце. Часто работу приложений по выводу речи дополняют возможностью воспроизведения коротких звуковых сигналов, так называемых “звуковых иконок”. Ими могут быть как просто синтезируемые звуки постоянной частоты, так и предварительно записанные короткие фрагменты музыки. Такими сигналами оповещают

пользователя о каких-либо событиях в приложении. Например, перемещения курсора на пустую строку при редактировании текста, закрытие окна, оповещение об отсутствии следующего элемента списка. Иногда бывает удобной функция воспроизведения некоторых знаков препинания через короткие звуковые отрывки.

К платформе речевого вывода предъявляются некоторые требования, не связанные напрямую с её главной задачей. Система должна организовывать вывод речи от нескольких приложений, не допуская одновременного произношения. Кроме того, существует ряд требований по конфигурированию речевых компонентов, т. к. очень важно сделать платформу легко устанавливаемой и настраиваемой неподготовленным пользователем.

Предлагаемая к рассмотрению платформа предназначена только для организации работы на компьютере в реальном времени, в то время как существует ряд других задач, связанных с обработкой речи. К ним относится, например, прослушивание больших книг из текстовых файлов. Это задача имеет иную природу, предпочтения и акценты в ней расставлены по-другому. При непосредственной работе на компьютере очень важны скорость отклика речевой системы, максимальная чёткость произношения и темп работы, так подобранный, чтобы пользователь не уставал от выполнения длительных задач. При прочтении книг требуется прежде всего ровность произношения, полное отсутствие знаков препинания и качественная обработка восклицательных и вопросительных предложений интонацией синтезатора.

# 2 Архитектура платформы вывода речи

## 2.1 Форма и содержание входных данных

Входная информация, которая должна быть обработана, может быть разделена на три группы:

- непосредственно текст в виде последовательности символов;
- команды, управляющие обработкой текста;
- команды, управляющие воспроизведением дополнительных сигналов (“звуковых иконок”).

Входной текст может содержать любые символы, включая знаки препинания. Обычно отсутствие какой-либо предварительной обработки оказывается предпочтительнее, поскольку определение правил пропуска и

замены знаков препинания внутри речевого сервера даёт возможность унифицированно подготовить текста для произношения. Тем не менее, бывают ситуации, когда специфика приложения требует явной обработки посылаемого текста, поэтому здесь сложно сформулировать строгую единую политику и приходится искать оптимальное компромиссное решение.

В настоящий момент предпочтительная кодировка для входного текста — *UTF–8*. Существующие речевые приложения не придерживаются каких-либо определённых правил выбора используемой кодировки. Иногда принимается решение на основе значения переменной окружения *LANG*. Также часто используется подход явного указания кодировки.В наше время предпочтительнее всегда пользоваться возможностями *UNICODE*, чтобы исключить несогласованное общение приложений и уменьшить количество параметров, требуемых для настройки платформы.

Частным случаем текстовой последовательности является строка, содержащая только один символ. Обычно существует специальная пометка, обозначающая, что нужно воспроизвести только один символ. Одиночные символы обрабатываются особым образом. Например, при обработке символа “ъ” для русского языка должна явно воспроизводиться строка “твёрдый знак”, в то время как при обычном воспроизведении этот символ явно не обрабатывается, и если он встречается вне слова, то просто пропускается. При отдельной обработке символов включается механизм различения заглавных и строчных букв. Заглавные буквы читаются более высоким голосом, чем строчные. При воспроизведении строки символов заглавные и строчные буквы не различаются.

При обработке текста воспринимаются следующие команды:

- команда немедленно прекратить воспроизведение текстового отрывка, полученного при предыдущем обращении;
- команды изменения характеристик произношения, таких как высота речи, скорость и громкость.

В приведённом списке указаны только некоторые часто используемые команды. Различные протоколы передачи текста могут содержать любые свои расширения. Например, возможны дополнительные команды переключения и выборов голосов, если они контролируются пользовательским приложением.

Команда изменения характеристик речи меняет параметры обработки только для того клиента, через подключение которого она была по-

лучена. При установлении нового сеанса все характеристики устанавливаются в некоторое начальное значение, указанное пользователем.

Наиболее распространённый вариант "звуковых иконок" — звуковые отрывки одной постоянной частоты. При посылке команды воспроизведения такого отрывка указывается длительность и частота сигнала. Речевой сервер генерирует нужный звуковой фрагмент и посылает его в аудио-устройство.

В настоящий момент поток входной информации имеет строго односторонний характер. В перспективе возможны задачи, требующие наличия обратной связи от сервера к клиенту, например, для получения оповещений об окончании обработки посланной части текста.

## 2.2 Инструменты для передачи входной информации

Речевая платформа строится по архитектуре клиент-сервер. Демон обработки текста получает входную информацию через сетевые подключения по протоколу /textitTCP/IP или через *UNIX domain sockets*. Несмотря на то, что у пользовательских приложений всегда есть возможность самостоятельно установить соединение с сервером и передать данные, лучше избегать такого прямого общения, т. к. должен существовать единый механизм определения положения сервера, используемого в данный момент (см. гл. 3.1). Кроме этого, реализация протокола общения должна быть изолирована от пользовательских приложений. Речевая среда предусматривает набор инструментов, которыми рекомендуется пользоваться в разных ситуациях. Предлагаются три метода передачи текстовых команд:

- консольная утилита, при помощи которой можно передать текст вручную и из *shehll*-скриптов;
- *API* для разработчиков пользовательских приложений (в настоящий момент доступна реализация только для языков *C/C++*);
- конвертеры для обработки текстовых команд существующих распространённых протоколов (в настоящий момент доступна реализация протокола *emacspeak*).

Самый простой клиент должен представлять собой небольшое консольное приложение, которое может работать как в режиме диалога, так и в режиме обработки параметров командной строки с немедленным выходом после передачи данных. Подобный клиент можно сравнить с хорошо известными консольными утилитами, которыми снабжены СУБД,

за исключением того, что это не инструмент администрирования, а в большей степени инструмент отладки и проверки. Возможность вызова консольной утилиты из *shell*-программ позволяет добавить оповещения о различных системных событиях в уже существующие скрипты. Удобные возможности открывает реализация обработчиков для демона *cron*. С его помощью можно добавить регулярные речевые уведомления о текущем времени, приходе личных писем и т. п.

*API* для разработчиков приложений — это базовый компонент связи с сервером. В настоящий момент *API* представляет собой несколько функций на языке *C*, при помощи которых можно установить соединение с сервером и передать необходимые команды. Функция установления соединения возвращает непрозрачный дескриптор текущего подключения, который необходимо указывать при посылке команд. Эти функции полностью изолируют реализацию протокола общения. В настоящей документации не приводится детальное описание интерфейса этих функций. За дополнительной информации обращайтесь к страницам *man*.

Конвертеры существующих протоколов необходимы для совместимости платформы с существующими приложениями и облегчения задач внедрения. Один из самых распространённых протоколов — протокол, по которому передаются команды от *emacspeak*. Речевой сервер не имеет прямой реализации этого протокола, поскольку emacspeak передаёт информацию, специфичную для аппаратных синтезаторов речи. Существующий конвертер удаляет из потока ненужные команды и преобразует данные к универсальной форме. К недостаткам протокола *emacspeak* можно отнести отсутствие каких-либо договорённостей об используемой кодировке для национальных символов.

## 2.3 Организация и функции речевого сервера

Речевой сервер — это центральный компонент обработки текстовой информации. Его главная задача — максимальная унификация всех процессов, задействованных при преобразовании текста в речь. Такой подход предъявляет к его реализации особые требования, т. к. речевой сервер не должен стать узким местом в работе системы. Кроме этого, поскольку он обрабатывает соединения по сети *TCP/IP*, данные, получаемые им через подключения клиентов, не должны создавать угрозы безопасности.

Речевой сервер “проговаривает” только одну фразу в каждый момент времени. Если было получено несколько команд на произношение текста, то они выстраиваются в очередь в порядке поступления и прогова-

риваются одна за другой. Все клиенты имеют одинаковые приоритеты при посылке команд. Клиент может запросить остановить произношение. При получении такой команды сервер немедленно останавливает воспроизведение текущей фразы и полностью очищает очередь. Если фраза состоит из отрывков текста на разных языках, то она разбивается на несколько фрагментов, каждый из которых добавляется в очередь отдельно.

С каждым элементом в очереди ассоциирован свой набор значений характеристик речи. При его создании эти значения устанавливаются на основе информации, хранимой в атрибутах клиента. При получении команды изменения характеристик, меняются только атрибуты клиента, но это не имеет никакого влияния на элементы текста, уже хранимые в очереди.

Важное значение в конфигурации сервера имеет понятие “вывода” (“output”) и настройка сервера должна иметь не менее одного вывода. Ниже приведены некоторые атрибуты, с которыми ассоциируется каждый вывод:

- имя вывода;
- подготовленный к работе речевой синтезатор;
- один из поддерживаемых национальных языков.

Имя вывода — это текстовая строка, используемая для его идентификации. В настоящий момент речевой синтезатор указывается командой, интерпретируемой *bash*, при помощи которой вызывается утилита преобразования. При её вызове текст для обработки будет послан на поток стандартного ввода. Более подробно механизм вызова синтезатора будет описан в гл. 3.2. Национальный язык указывается для определения некоторых дополнительных преобразований, которым подвергается текст перед произношением. Примером таких преобразований может служить представление чисел в их словесном виде.

На сервере создаются несколько выводов, и настраивается отдельный механизм выбора для каждого элемента в очереди фрагментов для произношения. Различные выводы используются прежде всего для автоматического переключения поддержки национальных языков. В настоящее время поддерживаются только английский и русский языки.

Сервер хранит специальную таблицу отображения различных символов во множество сконфигурированных выводов. На основе этой таблицы ведётся выбор подходящего вывода при обработке фрагментов тек-

ста. Некоторые символы, такие как знаки препинания и цифры, ассоциированы с выводом по умолчанию. Указание такого вывода обозначает, что они должны быть обработаны в том же потоке, что и предшествующий им текст. Это позволяет организовать прочтение знаков препинания и чисел тем же языком, что и текст, в котором они были встречены.

Речевой сервер запускается при старте системы как служба и сразу переключается в режим принятия подключений. В конфигурационном файле есть возможность указать строку приветствия, которая будет озвучена при старте сервера. Кроме того, сервер имеет возможность обработать строку текста в автономном режиме, в котором не будет инициализирован код обработки подключений, и сервер завершит свою работу сразу после окончания воспроизведения. Такая возможность может оказаться полезной в утилите конфигурирования, в которой пользователю будет предложена возможность попробовать выбранный им синтезатор.

## 2.4 Использование речевых синтезаторов

В подавляющем большинстве случаев речевой синтезатор — это утилита, воспринимающая текст через поток стандартного ввода и выдающая звуковые *PCM*-данные на поток стандартного вывода. Некоторые синтезаторы имеют функции посылки сгенерированных данных на аудиоустройство. Если нет такой возможности, то его вывод пересылается в утилиту *aplay* или *sox*.

Речевые синтезаторы не имеют в настоящий момент каких-либо единых стандартов на кодировку обрабатываемого текста. Поэтому иногда запуск синтезатора снабжается вызовом утилиты *iconv*.

Атрибуты речи, такие как высота и скорость указываются через параметры командной строки. Каких-либо единых стандартов на их форму не существует. В работе речевого сервера приходится явно поддерживать возможность указания формата, в котором будет передаваться информация о характеристиках речи в речевой синтезатор.

Иногда речевой синтезатор разбит на несколько компонентов. Например, речевой синтезатор *mbrola* способен выполнять только преобразование информации о звуках речи в аудио-данные. Подготовку текста и разбиение его на звуки выполняет отдельная утилита *freephone*.

Часто в работе синтезатора используются дополнительные библиотеки и словари. Таким словарём при обработки русской речи может быть словарь ударений.

# 3 Настройка речевого сервера VoiceMan и его клиентов

## 3.1 Использование клиентов речевого сервера "VoiceMan"

Ключевым моментом в использовании клиентских приложений является политика указания положения сервера, через который должна вестись обработка текста. Задача в том, что это должен быть единый метод, который все клиенты анализируют одинаковым алгоритмом. Механизм определения положения сервера встраивается в клиентское *API*, для которого варианты, основанные на чтении единого конфигурационного файла или на передаче дополнительных параметров из вызывающего приложения, нежелательны.

В данном сервере предлагается указывать эту информацию через переменную окружения *VOICEMAN*. Алгоритм анализа этой переменной следующий:

1. проверяется переменная окружения *VOICEMAN*. Если она отсутствует или пуста, то подключение производится через *UNIX domain socket*, путь к которому определяется параметром, указанным при компиляции сервера (в дистрибутивах *ALT Linux* — */var/run/voiceman.socket*);

2. если значение переменной не пусто, то анализируются её начальные символы:

   (a) если значение переменной начинается с символов *"TCP/IP:"*[1], то последующее содержимое переменной обрабатывается как имя хоста или *IP*-адрес с необязательным указанием порта через двоеточие;

   (b) в противном случае значение переменной рассматривается как путь к *UNIX domain socket*, через который нужно установить соединение.

Переменную окружения *VOICEMAN* можно устанавливать при входе в систему через скрипты в */etc/profile.d*. Значение порта по умолчанию при подключениях через *TCP/IP* указывается при компиляции. В настоящий момент он равен 5511.

[1] В будущем возможно эта строка будет изменена на *"inet:"*.

*API*-библиотека и консольный клиент имеют возможности явного указания положения сервера при подключении, но пользоваться ими следует обосновано. Рекомендуемая функция для подключении в языках *C/C++* — *vm_ connect()*. Она не имеет параметров и всегда самостоятельно определяет правильный вариант подключения.

Ниже будут описаны некоторые моменты использования консольного клиента и конвертера протокола *emacspeak*. За дополнительной информацией об использовании *API* для разработчиков обращайтесь к документации, поставляемой вместе с библиотекой в пакете *libvmclient-devel*.

### 3.1.1 Описание использования консольного клиента

Простейшая утилита для подключения к серверу вызывается командой *"voiceman"* в командной строке. Эта утилита поставляется в пакете *voiceman* (в настоящий момент клиенты и сервер поставляются в одном пакете из-за небольшого размера серверной части). Если в командной строке не указаны никакие параметры, утилита пытается произвести подключение в соответствии с алгоритмом, приведённом выше, и переходит в режим ввода команд. Если соединение с сервером было успешно установлено, то проговаривается любой текст, набранный в приглашении. Попытка ввода пустой строки посылает на сервер команду остановить воспроизведение. Если введённый текст содержит только один символ, то он будет передан специальной командой, обозначающей обработку одной буквы. В этом случае должно работать звуковое различение заглавных и строчных букв. Следующие команды имеют особое значение при вводе в консольный клиент:

- *pitch=N* — установить новое значение высоты голоса для текущего подключения;
- *rate=N* — установить новоге значение скорости произношения для текущего подключения;
- *volume=N* — установить новое значение громкости для текущего подключения.

Во всех приведённых командах число $N$ должно быть целым значением от 0 до 100. В случае установления громкости значение 0 означает полную тишину.

Полный список ключей командной строки и их назначения вызываются командой *"voiceman -h"*. Среди них нужно отметить ключ *"-S"*.

Этот ключ имеет один обязательный аргумент, а именно текстовую строку, которая будет отправлена на сервер. При использовании этого ключа приглашение для диалога ввода команд выдано не будет, и программа завершит свою работу сразу после передачи данных на сервер. Например, выполнив команду ‘*“voiceman -S "Now $(date +%H:%M)„*‘, можно услышать текущее время.

### 3.1.2 Конвертер протокола *emacspeak*

Утилита *voiceman-emacspeak*[2] — это приложение, выполняющее преобразование протокола *emacspeak* и пересылающая полученные данные на сервер. Эта утилита выполняет подключение к серверу по правилам, описанным выше, и не имеет механизмов указания специального положения сервера, т. к. она никогда не вызывается напрямую, а её запуском управляет внешнее приложение. *Emacspeak* — это не единственное приложение, которое можно использовать с *voiceman-emacspeak*. По этому протоколу работает и консольная утилита *yasr*.

Клиент *voiceman-emacspeak* не выдаёт ошибочных сообщений, если ему не удалось подключиться к серверу. Если подключение отсутствует, при получении каждой следующей команды будет выполняться новая попытка подсоединения. Все команды, обработанные при отсутствии связи с сервером, удаляются из обработки. Такое поведение было реализовано, чтобы добиться стабильной работы *emacspeak*, т. к. эта программа работает некорректно, если ей не удалось запустить обработчик вывода или он внезапно завершил свою работу.

Единственный ключ, который может быть использован при работе с *voiceman-emacspeak* — это *“-t”*. Указание этого ключа приводит к выполнению пробного подключения к серверу, посылке текстовой фразы и нескольких звуковых сигналов, после чего утилита немедленно завершает свою работу. Такая возможность часто оказывается полезной при выполнении отладочных действий.

## 3.2 Конфигурирование и настройка сервера *VoiceMan*

В репозиториях *ALT Linux* сервер *VoiceMan* подготовлен к использованию с минимальной рабочей конфигурацией. В настоящий момент синтезатор, используемый по умолчанию, — *espeak*. Для запуска сервера необходимо выполнить следующие команды с правами пользователя *root*:

```
# apt-get install voiceman espeak
```

[2] В будущих версиях возможно будет изменено на *vmespeak* или на *vm-espeak*.

```
# chkconfig voiceman on
# service voiceman start
```
После выполнения последней команды должно прозвучать приветствие.

Сервер запускается с правами *root*. Его построение допускает одновременный запуск нескольких экземпляров, как с правами *root*, так и в локальной пользовательской среде, но описание подобных конфигураций выходит за рамки этого документа.

### 3.2.1 Конфигурационный файл *voiceman.conf*

*VoiceMan* имеет единый конфигурационный файл, расположенный в */etc/voiceman.conf*. Он состоит из нескольких секций. Каждая секция начинается с заголовка в формате*"[имя]"*. Внутри секции может быть несколько параметров в форме *"ключ=значение"*. Имена ключей параметров и заголовки секций не чувствительны к регистру. Значения параметров должны заключаться в кавычки, если они содержат пробелы, знаки препинания и национальные символы. Если в значении необходимо вставить символ кавычек, то он должен быть продублирован. Все национальные символы должны быть в кодировке *UTF-8*. Все символы, которые следуют за символом "#" считаются комментарием и не обрабатываются программой. Пустые строки и строки, содержащие только пробелы и знаки табуляции, игнорируются. Если параметр предполагает логическое значение, то оно может быть указано в формах *true/false*, *yes/no* или *1/0*.

Две главные секции *"global"* и *"default"* присутствуют только один раз, в отличии от секции *"output"*, которая может встречаться несколько раз. Каждое её использование будет обозначать один допустимый вывод (см. гл. 2.3). Секция *"global"* содержит общие настройки сервера, среди которых параметры обработки сетевых подключений, некоторые численные настройки и пр. Секция *"default"* содержит конфигурацию обработки символов, которые будут обрабатываться выводом по умолчанию (см. гл. 2.3).

Ниже приводится список всех возможных параметров секции *"global"*:

- **capitalization** (boolean) — произносить слова, состоящие только из согласных букв, по буквам (только для анг. языка);
- **data dir** (string) — имя директории с дополнительными файлами настроек;
- **default pitch** (unsigned integer) — высота голоса по умолчанию для нового подключения (число от 0 до 100);

- **default rate** (unsigned integer) — скорость речи по умолчанию для нового подключения (число от 0 до 100);
- **default volume** (unsigned integer) — громкость голоса по умолчанию для нового подключения (число от 0 до 100);
- **digits** (enum) — режим обработки чисел (*none* — не обрабатывать, *single* — по цифрам, *normal* — обычная словесная форма);
- **log file name** (string) — имя файла лога (лог не ведётся, если этот параметр не указан);
- **log level** (unsigned integer) — уровень детальности лога;
- **log to stderr** (boolean) — посылать сообщения лога на поток ошибок (всегда выключено в режиме демона);
- **loop delay** (unsigned integer) — задержка цикла обработки элементов очереди;
- **maxc lients** (unsigned integer) — максимальное допустимое число клиентов (0 — не ограничено);
- **max input line** (unsigned integer) — максимальная длина строки в протоколе соединения (0 — не ограничена);
- **max queue** (unsigned integer) — максимальная длина очереди;
- **port** (unsigned integer) — номер порта *TCP/IP* для входящих соединений (если не указано, сокет *TCP/IP* не инициализируется);
- **separation** (boolean) — включить эвристический метод разделения слов, записанных без пробелов (полезно при наборе кода в программировании);
- **socket** (string) — путь к *UNIX domain socket* (если не указано, то сокет не инициализируется);
- **startup message** (string) — текст приветствия;
- **tones** (boolean) — воспроизводить “звуковые иконки”;
- **tones in queue** boolean() — помещать команды воспроизведения “звуковых иконок” в очередь, или обрабатывать сразу по получении.

Параметр *data dir* обозначает путь к директории, где располагаются некоторые вспомогательные данные, используемые при обработке полученного текста. К ним относятся, например, таблицы текстовых подстановок для знаков препинания. В следующей версии сервера этот параметр предполагается удалить из конфигурационного файла и задавать его значение переменными при компиляции.

При работе сервера в отдельном потоке выполняется бесконечный цикл, который управляет элементами очереди. При появлении нового текста в этом цикле производится обращение к нужному речевому синтезатору, а затем ожидается окончание произношения фразы. Параметр *loop delay* указывает задержку работы этого цикла. Отсутствие такой задержки привело бы к очень неэффективному использованию ресурсов процессора. Значение этого параметра сказывается на скорости реакции речевого сервера при поступлении нового текста, и слишком большое значение приведёт к заметному отставанию обработки и произношения текста.

Параметры *socket* и *port* отвечают за механизм принятия подключений. Если параметр *socket* не указан, то *UNIX domain socket* не будет инициализироваться при старте системы. Аналогично влияние параметра *port* на инициализацию *TCP/IP* сокета. Если не один из сокетов не указан в конфигурационном файле, то сервер не запускается, т. к. в этом случае отсутствует способ получения входных данных для сервера.

Секция *default* содержит только два параметра:

- **chars** (string) — список символов, которые должны обрабатываться тем же выводом, что и предшествующий им текст;
- **output** (string) — имя вывода, используемого в случае отсутствия предшествующего текста.

Как описывалось выше, существует возможность обрабатывать некоторые символы тем же выводом, что и предшествующий текст. Такая возможность полезна при обработке цифр и знаков препинания. Параметр *chars* определяет строку, состоящую из тех символов, которые должны быть обработаны таким образом. Если какой-либо из этих символов был использован в начале текстовой фразы и нет предшествующего текста, то возникает ситуация, в которой невозможно принять решение о выводе для обработки этого символа. Эта ситуация разрешается путём указания параметра *output*, В нём хранится имя вывода, используемого при обработки символов, для которых не удалось принять решение.

В Секции *output* допустимы следующие параметры:

- **cap list** (string) — список подстановок для произношения отдельных букв;
- **command** (string) — команда для вызова речевого синтезатора;
- **lang** (enum) — язык вывода. В настоящий момент допускаются только *eng* и *rus*;
- **name** (string) — имя вывода;
- **pitch** (string) — параметры преобразования высоты речи при вызове синтезатора;
- **rate** (string) — параметры преобразования скорости речи при вызове синтезатора;
- **type** (enum) — тип вывода;
- **volume** (string) — параметры преобразования громкости речи при вызове синтезатора.

В настоящий момент поддерживаются выводы только типа *command*. При использовании такого вывода синтезатор вызывается через команду, обрабатываемую *bash*. Текст для воспроизведения всегда посылается на поток стандартного ввода и всегда в кодировке *UTF-8*. Текст команды указывается в параметре *command*. Зарезервированы три специальные текстовые последовательности, на месте которых будут подставлены числовые параметры воспроизведения при вызове команды. Это *"%p"* — высота речи, *"%r"* — скорость речи и *"%v"* — громкость речи. Сервер определяет значения этих параметров в виде долей единицы, т. е. как дробное число от 0 до 1. Значение, подставляемое в команду, проходит дополнительную обработку и выражается формулой $n = k(max - min) + min$, где $n$ — значение, подставляемое в команду, $k$ — значение параметра в долях единицы, $min$ и $max$ — минимальное и максимальное значение этого параметра. После вычисления, значение $n$ округляется так, чтобы осталось только нужное количество десятичных знаков после запятой. Параметры $min$, $max$ и количество десятичных знаков определяются строкой вида *"количество знаков:min:max"*. Такая строка задаётся отдельно для высоты, скорости и громкости речи параметрами *pitch*, *rate* и *volume* соответственно.

Параметр *cap list* содержит строку пар, разделённых пробелами, где первый элемент пары — буква, а второй — текстовая строка, которая подставляется вместо буквы, если сервер указал, что некоторый символ должен быть прочитан отдельно от остального текста. Это бывает нужно для обработки сокращений.

# 4 Заключение

В рамках этого документа изложены многие вопросы, связанные с работой речевого сервера *VoiceMan*. Тем не менее, остался не освещённым важный вопрос о соглашениях конфигурирования речевого вывода в операционной системе. Для гибкого использования речевых приложений предстоит выработать набор правил, позволяющих организовать работу утилиты конфигурирования речевой подсистемы. Должен существовать единый способ получения списка установленных синтезаторов и выбора из него нужных для работы. В настоящий момент, возможно, организация конфигурационного файла для *VoiceMan* является неудобной для решения этих задач. Настройка каждого вывода должна производиться отдельным файлом. Все файлы с настройками выводов должны быть собраны в одной директории, и их список должен определять множество подготовленных выводов для работы.