Добрый день! И добро пожаловать в блок «Работа с системой go», в котором мы рассмотрим, package os – это стандартная библиотека, она входит в комплект поставки go, и позволяет нам делать достаточно много вещей в системе, какие мы узнаем чуть позже. Как работать с переменными окружения в системе, что это такое. И как работать или как приделать флаги запуска к вашему конкретному приложению. Поехали.

Так, разбираться, что делает package os. Там много всего, но, по сути, говоря, там есть работа с системой, с процессами и прочими системными штуками. Это абстракция над реальными функциями. То есть у библиотеки есть platform specific имплементация. То есть для window это будет один package, для Linux-а будет другой package, для разных версий Linux-а тоже будет разный package. То есть физический код там будет разный несмотря на то, что он обернут одни и те же функции. И заточено в плане интерфейсов и прочего на все-таки Linux-систему, то есть на структуру файлов, папок, на структуру permission-ов, которые приняты в Linux-системах. Естественно, window при этом поддерживается, но держать в голове модель Linux-а будет проще, при работе с package os.

По своей сути, package os состоит из, собственно говоря, классов и функций, предназначенных для запусков приложений. Из классов и функций для работы с файлами и папками. И соответственно. Не классов, а, простите, структур, конечно, и функций для отслеживания и управления процессами.

И давайте сейчас посмотрим, как это работает на практике. Для запуска процессов из нашей системы, будем использовать функцию os.exec из package os. И соответственно, она нужна для запуска команд из go. В чем фишка? Она оборачивает системные вызовы, ну как оборачивает, она, собственно говоря, позволяет делать эти системные вызовы. Потому что в норме из среды разработки вы не можете делать систему вызова. Вам нужно как-то прокинуть систему и,

соответственно, с этой системой как-то взаимодействовать. Она оборачивает эти системные вызовы, дает вам полный доступ как к вводу, так и к выводу приложения, можете писать приложение, можете читать из приложения.

В отличии от (НРЗ 02:20), например, не использует shell. То есть не вызывает shell, и в нем не запускает приложения. Таким образом, ваши файлы, ваш (НРЗ 02:28), ваши там переменные, которые под своим пользователем в shell настраиваются, они не влияют на запуск приложения. И к сожалению, может не работать под window, вернее оно работает под window, вы можете вызывать приложения и команды, но не всегда, если у вас там есть проблемы с правами доступа, если пользователь не видит папку и так далее. Ну в общем, сами понимаете, что я уже говорил, заточено под Linux, поэтому есть некоторые проблемы с запуском exec из-под window.

При первом приближении, у любого разработчика возникает вопрос, чаще ли мне надо запускать приложения другие-то, я же могу там библиотеки использовать, какие-то вещи подключать, что-то скомбинировать, сделать приложения, сделать модуль, скачать, блин, столько всего есть.

SysOps-ы, DevOps-ы знают ответ на этот вопрос. Естественно, во-первых, вы можете переиспользовать функциональность других приложений, вам не нужно включать helm, например, если вы работаете с каким-то deploy-ем, полностью свое приложение, если только вам нужно обработать маленький файл. Вам не нужно тащить все эти огромные зависимости, все (НРЗ 03:33). Опять же поддерживать версии и так далее.

Если в системе есть helm, это будет работать. Если нет, значит, helm надо установить. Это гораздо проще менеджить по отдельности. Опять же, если у вас Linux, и вы делаете какие-то системные вещи, то вы скорее всего знаете про

подход, что, значит, вы делите приложение тоже, как и у нас до этого в лекции было, по их использованию. То есть вы в одно приложение включаете одну какую-то функцию. У вас команда ls просматривает листик директорий, просматривает файлы и папки в директориях. Она не копирует файлы и папки, она только их смотрит. Команда cp только копирует, она не может посмотреть файлы и папки. Ну и так далее, и так далее.

Поэтому, если ваше приложение написано в таком же стиле, скорее всего оно может использовать какие-то другие вещи. Вы можете обрабатывать вывод этих приложений и его хранить. Например, вы можете логировать выводы helm-а и как-то их там в файлы складывать. И уже обрабатывать, какие-то метрики из них строить, ну, например.

И опять же, даже если вам все это не надо, вы разработчик, не всегда есть нужные библиотеки, не всегда есть библиотека, которая легко вам позволяет реализовать какую-то функциональность уже существующего…, просто консольной утилиты, или существующего решения. Теперь давайте перейдем уже, непосредственно, к практике.