Ну что же, мы рассмотрели системные переменные, environment-ы. Давайте теперь посмотрим на флаги запуска. Что такое флаги запуска все знают – это, соответственно, те аргументы, которые передаете приложению, после его запуска. Например, LS минус L. Минус L – это факт запуска. Опять же, их там может быть целая куча. Golang, естественно, их поддерживает. Любой системный язык поддерживает какую-то оболочку для работы с флагами запуска. И нужны они, в первую очередь, для установки специфичных настоек для запуска приложений.

Здесь несколько неоднозначно, то есть специфичные настройки у всех бывают разные. Для консольных системных утилит, это, в принципе, может быть какая-то система-полагающая вещь. То есть, что приложению делать для, например, каких-то веб сервисов. Если разработчики туда поставили флаги запуска, они, например, могут отправлять состоянием какой-то системной штуки низкоуровневой, которую не надо контролировать переменными окружения. Например, в приложениях на Java-е, это количество памяти у Java машины. Оно становиться не переменным окружение, а флагом запуска. Потому что это атрибут, непосредственно, самой машины, а не вашего бизнес-приложение. Опять же, позволяет управлять приложением извне. Разработчик предусматривал флаги запуска, вы как-то их используйте.

И принадлежит конкретному запуску. То есть здесь нет никаких внешних переменных, которые где-то там тусуются. Это конкретно ваш запуск. Конкретно ваши флаги. Они принадлежат каждый раз каждому новому запуску приложения. Могут, естественно, в рамках сессии одного пользователя меняться.

Давайте посмотрим о флаге на практике. Я даже не стал здесь ничего усложнять. Просто взял стандартный пример из golana, из стандартной библиотеки обучающих, соответственно, примеров. И посмотрим, что здесь есть.

Соответственно, управление флагами осуществляется package-ом flag, который является отдельным от package os. Это не принадлежит os.

Соответственно, флаги бывают разных типов. То есть существует еще какая-то внутренняя валидация. Если у вас тип не совпадают, ваше приложение упадет. Соответственно, у флага существует имя, которое вам передается. То есть при вызове программы, мы будем писать минус, минус word. Значит, стандартное значение, которое будет установлено, если нет никакого флага, и соответственно, описание этого флага. То есть, если создается флаг string, флаг int, флаг bool и специальный флаг, то есть, как вы видите, они возвращают какие-то значения, и специальный флаг, который записывается в переменных. То есть, если вам что-то нужно обработать в переменной, вы можете ее сразу передать в флаги. И соответственно, после вызова метода parse, это все будет обработано.

Соответственно, помимо явного указания флага, библиотека флаг имеет побочную функцию, при вызове help-а, у вас все-таки вызовется справка по флагам, и будет выведено их описание, и будет выведено default-ное значение, если таковые имеются. Таким образом, вы как бы делаете встроенный (НРЗ 03:16).

Давайте теперь посмотрим на установку флагов. Итак, я здесь поставлю два флага, соответственно, word в test, и numb тоже поставлю в test. Потому что я хочу посмотреть, как работает валидация, numb у меня должен быть номер. Да, вот у нас неверное значение test для флага numb. Соответственно, если я туда поставлю нормальное значение, то сейчас все пройдет. Переменная svar, превратиться в slurm. И в tail, в flag.args, в аргументы, которые переданы после флагов, то есть в уже, непосредственно, какой-то там строчке, которую мы обрабатываем или чем-то еще. У меня появятся мои аргументы. То есть смотрит. Да, действительно, вот у нас svar, превратилась в slurm. Numb превратилась в четыре единички. Word превратился в test. И соответственно, в args у нас

храниться переменная test. В принципе, реально, про флаги больше рассказывать нечего. Если вы пишите какие-то консольные приложения, то это будет тот функционал, который вы используете.

Подчеркну, что очень важное использовать, очень важно понимать, когда мы используем флаги, а когда мы используем переменное окружение. Флаги, как я уже говорил, это либо консольные утилиты, которыми вы управляете, поведением консольной утилиты, так называемые, кнопочки контроля управления, либо это какие-то системные настройки, например, память для Java машины, или память для (HP3 04:45) в Go и так далее.

В go, во внесистемном программировании, в веб, например, приложениях, флаги практически не используются. Используется env. Env подходит для установки паролей. Или туда вы можете запихнуть большие реально данные, потому что, если вы хотите передать большие данные во флагах, это вот будут большие кавычки, супер-длинные строки. Это ну не очень удобно. Соответственно, можно, естественно, их использовать совместно, можно использовать, как флаги вместе с environment-ом, так и чистый environment, так и чистые флаги. Это уже зависит от ваших целей.

Единственное, что хотелось бы добавить, что, если у вас приложение реально маленькое и какая-то консоль утилита для DevOps-а, лучше не используйте environment variable, потому что вы не знаете, вы не контролируете, как правило, environment variable, который на машинах происходит. Их, как правило, контролируют разработчики. И что-то они так и могут поломать. Если у вас, естественно, какие-то утилиты, которые требуют там каких-то портов или требуют чтение чего-то, ну это уже, непосредственно, лежит в зоне вашей ответственности. Универсального рецепта здесь нет.

И поздравляю, мы дошли до конца. Мы осилили этот довольно большой блок. Поняли, как использовать флаги env, как запускаются сторонние приложения, из Go. Я надеюсь, здесь было интересно. И увидимся в следующих шагах.