Руководство пользователя kas

По материалам https://kas.readthedocs.io/en/latest/.

Этот инструмент предназначен для упрощения организации проектов на основе bitbake.

Поддержка инструментария OpenEmbedded начинается с bitbake на этапе 2. Загрузка исходных кодов и настройка обычно выполняется вручную, как описано в файлах README. Вместо этого kas использует файл конфигурации проекта и самостоятельно выполняет загрузку и настройку конфигурации.

Основные свойства инструмента сборки включают:

- клонирование и извлечение уровней bitbake;
- задание принятых по умолчанию настроек bitbake (machine, arch и т. п.);
- запуск минимальной среды сборки, снижающий риск загрязнения сборочного хоста;
- запуск процесса сборки с помощью bitbake.

Оглавление

Установка

Для работы с kas должны быть установлены пакеты:

- Python 3

- distro Python 3;
- jsonschema Python 3;
- PyYAML Python 3 (необязательно, служит для поддержки файлов yaml);
- kconfiglib Python 3 (необязательно, служит для подключаемого модуля menu);
- NEWT Python 3 distro (необязательно, служит для подключаемого модуля menu).

Для установки kas в репозиторий python site-package служит команда

```
$ sudo pip3 install .
```

Работа с пакетом

Имеется по меньшей мере 4 варианта работы с пакетом kas.

- Локальная установка с помощью pip (обеспечивает поддержку команды kas).
- Использование локального образа контейнера. Для этого загружается сценарий kas-container из репозитория kas, применяемый вместо команды kas. Версия сценария соответствует версии kas и образа kas.
- Использование образа контейнера в сценариях интерпретатора команд. Для этого в сценарий включается строка ghcr.io/siemens/kas/kas[-isar][:<x.y>], запрашивающая образ контейнера в качестве рабочей среды. Доступные версии образов можно получить по ссылкам https://github.com/orgs/siemens/packages/container/kas%2Fkas/versions и https://github.com/orgs/siemens/packages/container/kas%2Fkas-isar/versions.
- Использование оболочки **run-kas**. В этом случае в приведённых ниже примерах следует заменять kas на path/to/run-kas.

Для запуска сборки служит команда

```
$ kas build /path/to/kas-project.yml
```

Опытные пользователи bitbake могут использовать задаваемые вручную действия bitbake, например,

```
$ kas shell /path/to/kas-project.yml -c 'bitbake dosfsutils-native'
```

Пакет kas будет помещать загружаемые файлы и артефакты сборки в текущий каталог (откуда вызывается kas). Можно указать иной каталог с помощью переменной окружения KAS_WORK_DIR.

Примеры использования

1. Начальная сборка (установка)

```
$ mkdir $PROJECT_DIR
$ cd $PROJECT_DIR
$ git clone $PROJECT_URL meta-project
$ kas build meta-project/kas-project.yml
```

2. Обновление или новая сборка

```
$ cd $PROJECT_DIR/meta-project
$ git pull
$ kas build kas-project.yml
```

3. Интерактивная настройка конфигурации

```
$ cd $PROJECT_DIR/meta-project
$ kas menu
$ kas build  # необязательно, если не вызывается через меню kas
```

Подключаемые модули

Субкоманды kas реализованы в форме подключаемых модулей (plugin), каждый из которых обычно поддерживает 1 команду.

build

Этот модуль реализует команду kas build. При выполнении команды kas извлекает (checkout) репозитории, организует среду сборки, а затем вызывает bitbake для сборки целей (target), указанных в файле конфигурации. Например, для сборки с конфигурацией из файла kas-project.yml служит команда

```
kas build kas-project.yml
```

checkout

Этот модуль реализует команду kas checkout. При выполнении команды kas извлекает (checkout) репозитории и организует каталог сборки в соответствии с файлом конфигурации. Эта команда полезна в случаях, когда нужно проверить конфигурацию или изменить какой-либо из выбранных уровней до начала сборки. Например, для установки конфигурации, заданной в файле kas-project.yml, служит команда

```
kas checkout kas-project.yml
```

dump

Этот модуль реализует команду kas dump. При выполнении этой команды в принятом по умолчанию режиме kas будет анализировать все указанные файлы конфигурации со включёнными файлами и выводить на стандартное устройство краткую yaml-версию конфигурации. Эта конфигурация семантически идентична входной, но не содержит ссылок на другие конфигурационные файлы. Вывод команды может применяться для анализа конфигурации системы сборки.

При запуске с опцией --lock создаётся спецификация блокировки, содержащая лишь точные указания каждого репозитория. Это может быть полезно для фиксации плавающих веток при сохранении простого пути обновления. При задании вместе с опцией --inplace создаётся файл блокировки вместе (см. kas.includehandler.IncludeHandler).

Следует отметить, что

- выгруженная конфигурация идентична семантически, но побитово;
- все указанные репозитории извлекаются для устранения конфликтов;
- все ссылки (refspec) преобразуются (resolv) до применения правок (patch).

Например, для вывода конфигурации, представляющей окончательную конфигурацию сборки kas-project.yml:target-override.yml можно ввести команду

```
kas dump kas-project.yml:target-override.yml > kas-project-expanded.yml
```

Созданный файл конфигурации можно использовать для передачи в kas

```
kas build kas-project-expanded.yml
```

Ниже представлен пример использования механизма блокировки (повторный вызов для воссоздания файла блокировки) для создания файла блокировки kas-project.lock.yml

```
kas dump --lock --inplace --update kas-project.yml
```

Созданный файл блокировки будет автоматически применяться для закрепления версии (ревизии)

```
kas build kas-project.yml
```

Отметим, что файлы блокировки следует регистрировать в системе контроля версий (VCS).

for-all-repos

Этот модуль реализует команду kas for-all-repos, при выполнении которой kas просматривает (checkout) указанные в выбранном файле конфигурации репозитории и выполняет для каждого заданную команду. Это может служить для запроса состояний репозиториев, автоматизации операций (таких как архивирование использованных при сборке уровней) или выполнения иных команд.

Например, для вывода хэш-значений представлений (commit), использованных в каждом репозитории из файла kas-project.yml (в предположении, что это репозитории git), можно воспользоваться командой

```
kas for-all-repos kas-project.yml 'git rev-parse HEAD'
```

В рабочую среду для выполнения команды в каждом репозитории включаются указанные ниже переменные.

- KAS_REPO_NAME - имя текущего репозитория, определяемое свойством name или ключом этого репозитория в файле конфигурации.
- KAS_REPO_PATH - путь к локальному каталогу, в котором хранится репозиторий, относительно каталога, откуда запускается kas.
- KAS_REPO_URL - URL, откуда был извлечён репозиторий, или пустая строка, если в файле конфигурации нет удалённого URL.
- KAS_REPO_REFSPEC - ссылка на спецификацию (refspec) для этого репозитория или пустая строка, если в файле конфигурации нет refspec.

menu

Этот модуль реализует команду kas menu, открывая меню конфигурации, как описано в файле Kconfig. Обрабатываются все имеющиеся файлы конфигурации с сохранёнными настройками, записывается окончательный выбор и вызывается модуль сборки (build plugin), если это запрошено пользователем. Для использования этого модуля нужен файл Kconfig. Меня может задавать указанные ниже типы переменных конфигурации, которые модуль будет транслировать в настройки kas.

- Файлы конфигурации kas, которые будут включаться в создаваемую конфигурацию. Файлы берутся из строковых переменных kconfig с префиксом KAS_INCLUDE_.
- Цели bitbake, которые нужно собрать с помощью создаваемой конфигурации. Цели берутся из строковых переменных kconfig с префиксом KAS_TARGET_.
- Используемая система сборки build_system. Значение определяется статической переменной KAS_BUILD_SYSTEM (openembedded, oe или isar).
- Переменные конфигурации bitbake, добавляемые к раздел local_conf_header создаваемой конфигурации. Остальные переменные kconfig (string, integer, hex) трактуются обычным способом.

Полное описание языка Kconfig доступно по ссылке https://www.kernel.org/doc/html/latest/kbuild/kconfig-language.html.

Модуль menu записывает выбранную конфигурацию в файл .config.yaml в рабочем каталоге kas, а также считывает прежний выбор, если такой файл уже имеется. Файл .config.yaml содержит выбранную конфигурацию (ключ menu_configuration), а также действующие настройки, которые могут использоваться при вызове kas build или иных команд kas.

shell

Этот модуль реализует команду kas shell, просматривая (checkout) репозитории, организуя среду сборки и запуская в ней интерпретатор команд (shell). Это может служить для запуска bitbake вручную со своими опциями или выполнения таких команд, как runqemu. Например, для запуска оболочки в среде сборки проекта kas-project.yml можно использовать команду

```
kas shell kas-project.yml
```

Для вызова меню или проверки собираемого образа можно воспользоваться командой

```
kas shell kas-project.yml -c 'runqemu'
```

Конфигурация проекта

В настоящее время поддерживаются базовые форматы файлов JSON и YAML. Поскольку формат YAML проще для чтения, в этом документе применяются примеры именно в этом формате.

```
# Каждый файл должен иметь заголовок, предоставляющий kas сведения
# о контексте данного файла.
header:
  # Запись version в заголовке указывает, для какой версии формата
  # конфигурации создан файл. Это позволяет kas проверить совместимость.
  # Версия указывается целым числом, которое увеличивается при каждой
  # смене формата.
  version: x
# Машина указывается как в файле local.conf для bitbake.
machine: qemux86-64
# Имя дистрибутива (distro) как в файле local.conf для bitbake.
distro: poky
repos:
  # Эта запись включает репозиторий, где находится файл конфигурации
  # для bblayers.conf
  meta-custom:
  # Здесь указывается список уровней из репозитория poky для
  # bblayers.conf
  poky:
    url: "https://git.yoctoproject.org/git/poky"
    refspec: 89e6c98d92887913cadf06b2adb97f26cde4849b
    layers:
      meta:
      meta-poky:
      meta-yocto-bsp:
```

Минимальный входной файл содержит разделы header, machine, distro и repos. Можно также включить в файл разделы bblayers_conf_header и local_conf_header, содержащие строки, добавляемые к заголовкам соответствующих файлов (bblayers.conf и local.conf).

```
bblayers_conf_header:
  meta-custom: |
    POKY_BBLAYERS_CONF_VERSION = "2"
    BBPATH = "${TOPDIR}"
    BBFILES ?= ""
local_conf_header:
  meta-custom: |
    PATCHRESOLVE = "noop"
    CONF_VERSION = "1"
    IMAGE_FSTYPES = "tar"
```

В этих примерах значению meta-custom следует быть уникальным для конфигурационной записи. Рекомендуется делать это уникальное имя совпадающим с именем содержащего репозитория или уровня, чтобы сделать более понятными ссылки между проектами.

В примерах предполагается, что файл конфигурации является частью репозитория или уровня meta-custom. Это позволяет переопределять или добавлять записи в файлах, включающих данную конфигурацию за счёт повтора имён (переопределение) или использования новых (добавление в конец).

Включение файлов конфигурации из основного дерева

Возможно включение конфигурационных файлов kas из того же репозитория или уровня, как показано ниже.

```
header:
  version: x
  includes:
    - base.yml
    - bsp.yml
    - product.yml
```

Пути к файлам в списке includes могут быть относительными или абсолютными (начинаются с */*). Если задан относительный путь и файл конфигурации находится в репозитории, путь определяется относительно базового каталога репозитория. Если файл конфигурации находится вне репозитория, путь определяется от родительского каталога этого файла.

Включение файлов из других репозиториев

Можно также включать конфигурационные файлы из других репозиториев, как показано ниже.

```
header:
  version: x
  includes:
    - repo: poky
      file: kas-poky.yml
    - repo: meta-bsp-collection
      file: hw1/kas-hw-bsp1.yml
    - repo: meta-custom
      file: products/product.yml
repos:
  meta-custom:
  meta-bsp-collection:
    url: "https://www.example.com/git/meta-bsp-collection"
    refspec: 3f786850e387550fdab836ed7e6dc881de23001b
    layers:
```

```
      # Кроме уровней, добавленных из этого репозитория в
      # hw1/kas-hw-bsp1.yml, добавляется метауровень bsp.
      meta-custom-bsp:
  poky:
    url: "https://git.yoctoproject.org/git/poky"
    refspec: 89e6c98d92887913cadf06b2adb97f26cde4849b
    layers:
      # Если kas-poky.yml уже добавляет уровень meta-yocto-bsp и он
      # не нужен в bblayer этого проекта, его можно переопределить
      meta-yocto-bsp: excluded
```

Местоположение файлов определяется относительно репозитория git.

Механизм включения собирает и объединяет содержимое сверху вниз (по файлу) и в глубину (структуры). Это означает, что установки из одного включённого файла могут быть переопределены последующими файлами, а установки из последнего включаемого файла - текущим файлом. При слиянии все словари объединяются рекурсивно с сохранением порядка, в котором записи добавляются в словарь. Это означает, что записи local_conf_header добавляются в файл local.conf в том же порядке, как они были указаны в других файлах конфигурации. Отметим, что порядок записей не сохраняется внутри одного включаемого файла, поскольку синтаксический анализатор создаёт обычные неупорядоченные словари.

Включение файлов конфигурации из команды

При указании файла конфигурации kas в строке команды можно включить дополнительные файлы, как показано ниже.

```
$ kas build kas-base.yml:debug-image.yml:board.yml
```

Это эквивалентно статическому включению, например, некого файла kas-combined.yml, показанного ниже.

```
header:
  version: x
  includes:
    - kas-base.yml
    - debug.image.yml
    - board.yml
```

Включение из команды позволяет создавать конфигурации по потребности без необходимости писать файлы конфигурации kas для всех возможных комбинаций.

Отметим, что файлы конфигурации, собранные из команды, должны приходить из одного репозитория, либо не использовать контроля версий. Все прочие комбинации kas будет отвергать во избежание сложностей и ошибок конфигураций, которые могут возникать.

Работа с файлами блокировки

В kas поддерживается применение файлов блокировки для привязывания репозиториев к точному указанию ссылок (refspec, например, ссылок SHA-1 для git). Таким образом, файл блокировки переопределяет лишь refspec, заданные в файле kas. При извлечении (checkout) kas проверяет наличие файла <filename>.lock.<ext> рядом с первым файлом, указанным в строке команды kas. Если такой файл имеется, kas добавляет имя этого файла в свою строку команды и выполняет запрошенную операцию.

Ниже приведён пример для файла kas/kas-isar.yml и соответствующего файла блокировки kas/kas-isar.lock.yml.

```
kas/kas-isar.yml:
# [...]
repos:
  isar:
    url: https://github.com/ilbers/isar.git
    refspec: next

kas/kas-isar.lock.yml:
header:
  version: 14
overrides:
  repos:
    isar:
      refspec: 0336610df8bb0adce76ef8c5a921c758efed9f45
```

Модуль dump предоставляет вспомогательные функции для упрощения создания и обновления фалов блокировки.

Содержимое файла конфигурации

header: dict [обязательно]

Заголовок конфигурационного файла kas, содержащий сведения о контексте файла конфигурации.

version: integer [обязательно]

Позволяет kas проверить совместимость с файлом. Описание версий приведено в разделе Версии формата конфигурации.

includes: list [необязательно]

Список файлов конфигурации, на которых основан текущий файл. Эти файлы объединяются в порядке их указания и каждый следующий файл может переопределять установки любого из предшествующих. Текущий файл может переопределять установки включённых файлов. Элементы списка могут иметь 1 из двух типов:

- ***item: string*** - указывает путь к файлу конфигурации kas от корня репозитория с текущим файлом;
- ***item: dict*** - применяется для включения файлов из других репозиториев
  - ***repo: string [обязательно]*** - идентификатор репозитория, где размещается файл (репозиторий должен быть указан в словаре репозиториев как <repo-id>
  - ***file: string [обязательно]*** - путь к файлу от корня указанного репозитория.

build_system: string [необязательно]

Указывает систему сборки на основе bitbake (openembedded или oe, isar). При установке этой переменной поиск kas сценария инициализации ограничивается ограничивается репозиториями, заданными для oe-init-build-env и isar-init-build-env, соответственно. Если kas-container находит это свойство в файле конфигурации kas верхнего уровня, автоматически выбирается требуемый образ контейнера и режим вызова.

defaults: dict [необязательно]

Может применяться для установки принятых по умолчанию значений различных свойств. Это может помочь избежать многократного назначения одного свойства, если, например, нужно применять одно значение refspec для всех репозиториев.

repos: dict [необязательно]

Этот ключ может содержать принятые по умолчанию значения для некоторых свойств репозитория. Такие значения могут переопределяться установкой для свойства иного значения в данном репозитории.

- ***refspec: string [необязательно]*** - задаёт принятое по умолчанию свойство refspec для всех репозиториев, где оно не переопределено.
- ***patches: dict [необязательно]*** - этот ключ может содержать заданные по умолчанию значения некоторых свойств исправлений (patch) для репозитория. Эти значения можно переопределить установкой иного значения в данном исправлении.
  - ***repo: string [необязательно]*** - устанавливает применение принятого по умолчанию свойства repo ко всем исправлениям в репозитории, где оно не переопределено.

machine: string [необязательно]

Содержит значение переменной MACHINE, записываемое в файл local.conf. Значение можно переопределить через переменную KAS_MACHINE, по умолчанию задано значение qemux86-64.

distro: string [необязательно]

Содержит значение переменной DISTRO, записываемое в файл local.conf. Значение можно переопределить через переменную KAS_DISTRO, по умолчанию задано значение poky.

target: string [необязательно] or list [необязательно]

Указывает цель (или список целей) сборки с помощью bitbake. Значение можно переопределить через переменную KAS_TARGET, по умолчанию задано значение core-image-minimal. При указании нескольких целей в переменной разделителями целей служат символы пробела.

env: dict [необязательно]

Содержит имена переменных окружения с принятыми по умолчанию значениями или None. Эти переменные доступны bitbake через BB_ENV_EXTRAWHITE и могут быть переопределены переменными окружения, в котором запускается kas. В качестве значения может использоваться строка (string) или ничего (None). Строка задаёт принятое по умолчанию значение, а None ведёт лишь к добавлению переменной в BB_ENV_EXTRAWHITE и не меняет окружение запуска kas.

task: string [необязательно]

Указывает задачу для сборки с помощью bitbake. Может быть переопределена переменной KAS_TASK, по умолчанию принято значение build.

repos: dict [необязательно]

Содержит определения всех доступных репозиториев и уровней.

<repo-id>: dict [необязательно]

Указывает репозиторий и уровни, которые следует включать в сборку. При значении None репозиторий, где размещён текущий файл конфигурации, определяется как <repo-id> и добавляется в качестве уровня сборки. Рекомендуется связывать <repo-id> с содержащим его репозиторием или уровнем для упрощения ссылок между проектами.

- ***name: string [необязательно]*** - задаёт имя, с которым хранится репозиторий. При отсутствии параметра применяется значение <repo-id>.
- ***url: string [необязательно]*** - url репозитория. Отсутствие параметра отменяет контроль версий.
- ***type: string [необязательно]*** - тип репозитория для контроля версий (по умолчанию git, поддерживается также hg).
- ***refspec: string [необязательно]*** - спецификация выпуска, который следует использовать. При наличии url без указания refspec получаемый выпуск (revision) зависит от принятых по умолчанию настроек контроля версий.
- ***path: string [необязательно]*** - путь к сохранённому репозиторию. При отсутствии url и path применяется репозиторий, где размещён текущий файл конфигурации. При отсутствии url и наличии path запись ссылается на каталог, который указывает path. При наличии url и path значение path применяется для переопределения каталога checkout (по умолчанию kas_work_dir + repo.name). При указании в path относительного значения в начало добавляется kas_work_dir.
- ***layers: dict [необязательно]*** - указывает уровни из данного репозитория, которые следует включить в bblayers.conf. При отсутствии параметра, значении None или пустом словаре в качестве уровня добавляется путь к самому репозиторию. Можно указать точку (.), если в качестве уровня следует добавить сам репозиторий. Это позволяет создавать комбинации вида

```
repos:
  meta-foo:
    url: https://github.com/bar/meta-foo.git
    path: layers/meta-foo
    refspec: master
    layers:
      .:
      contrib:
```

Приведённый фрагмент добавляет layers/meta-foo и layers/meta-foo/contrib из репозитория meta-foo в файл bblayers.conf.

- ***<layer-path>: enum [необязательно]*** - добавляет уровень с <layer-path> от корневого каталога репозитория в файл bblayers.conf, если значение этого параметра не является одним из списка [disabled, excluded, n, no, 0, false]. Это позволяет переопределять включение уровня в загружаемых позднее файлах.

patches: dict [необязательно]

Указывает правки (patch), которые следует применить в репозитории до его использования.

<patches-id>: dict [необязательно]
Одна из записей для правок с уникальным идентификатором, определяющим порядок применения правок.
- ***repo: string [обязательно]*** - идентификатор репозитория, от которого идёт путь в данной записи.
- ***path: string [обязательно]*** - путь к одному из patch-файлов или каталогу patchset в формате quilt.

overrides: dict [необязательно]
Этот объект обеспечивает механизм для переопределения элементов конфигурации kas без их определения. Таким образом, переопределяются лишь имеющиеся элементы. Отметим, что все записи под этим ключом резервируются для автоматической генерации с использованием подключаемых модулей kas и не следует добавлять их вручную.

repos: dict [необязательно]
Задаёт сопоставление с записью репозитория верхнего уровня.
- ***<repo-id>: dict [необязательно]*** - отображение на запись <repo-id>.
  - ***refspec: string [необязательно]*** - задаёт refspec для переопределения refspec соответствующего репозитория. Значение refspec должно быть распознаваемым (не включать branch или tag).

bblayers_conf_header: dict [необязательно]
Строки, которые следует добавлять в bblayers.conf перед включением какого-либо уровня.

<bblayers-conf-id>: string [необязательно]
Строка, добавляемая в bblayers.conf. Идентификатор записи (<bblayers-conf-id>) следует делать уникальным, если нужно добавлять строки, и можно применять строки, совпадающие со строками из других включаемых файлов, если запись следует переопределять. Строки добавляются в bblayers.conf по алфавитному порядку <bblayers-conf-id> для обеспечения детерминированной генерации конфигурационных файлов.

local_conf_header: dict [необязательно]
Строки, которые следует добавлять в local.conf.

<local-conf-id>: string [необязательно]
Строка, добавляемая в local.conf. Обработка выполняется так же, как для записей bblayers_conf_header.

menu_configuration:: dict [необязательно]
Указывает выбор пользователя для меню Kconfig в проекте. Каждая переменная соответствует переменной конфигурации Kconfig и может иметь тип string, boolean или integer. Содержимое этого ключа обычно поддерживается подключаемым модулем kas menu в файле .config.yaml.

Использование команд

Пакет kas служит инструментом для проектов на основе bitbake.

```
kas [-h] [--version] [-d] [-l {debug,info,warning,error,critical}]
          {build,checkout,dump,for-all-repos,shell,menu} …
```

Позиционные аргументы

build, checkout, dump, for-all-repos, shell, menu, а также субкоманда help.

Именованные аргументы

--version
Выводит номер версии программы и завершает работу.

-d, --debug
Включает запись отладочных сведений в системный журнал. Параметр устарел и заменён --log-level debug.

-l, --log-level
Возможные значения: debug, info (принято по умолчанию), warning, error, critical.

Субкоманды

build

Проверка всех требуемых репозиториев и сборка с помощью bitbake в соответствии с файлом конфигурации.

```
kas build [-h] [--skip SKIP] [--force-checkout] [--update] [--target TARGET]
          [-c TASK]
          [config] [extra_bitbake_args …]
```

Позиционные аргументы

config
Файл конфигурации. По умолчанию применяется .config.yaml из каталога KAS_WORK_DIR.

extra_bitbake_args
Дополнительные аргументы для передачи bitbake (обычно требуется разделять с помощью --)

Именованные аргументы

--skip
Пропустить этапы сборки SKIP (по умолчанию список пуст).

--force-checkout
Всегда проверять желаемое значение refspec каждого репозитория, отбрасывая любые локальные изменения. По умолчанию отключено (False).

--update
Вносить (pull) новые изменения (upstream) для желаемого refspec, даже если они внесены (checked out) локально. По умолчанию отключено (False).

--target
Задаёт цель для сборки.

-c, --cmd, --task
Задаёт выполняемую задачу.

checkout

Извлекает все требуемые репозитории и организует каталог сборки в соответствии с файлом конфигурации.

```
kas checkout [-h] [--skip SKIP] [--force-checkout] [--update] [config]
```

Позиционные аргументы

config
Файл конфигурации. По умолчанию применяется .config.yaml из каталога KAS_WORK_DIR.

Именованные аргументы

--skip
Пропустить этапы сборки SKIP (по умолчанию список пуст).

--force-checkout
Всегда проверять желаемое значение refspec каждого репозитория, отбрасывая любые локальные изменения. По умолчанию отключено (False).

--update
Вносить (pull) новые изменения (upstream) для желаемого refspec, даже если они внесены (checked out) локально. По умолчанию отключено (False).

dump

Извлекает окончательную конфигурацию и выводит её на stdout. При обработке refspec она происходит до внесения правок (patch).

```
kas dump [-h] [--skip SKIP] [--force-checkout] [--update]
         [--format {yaml,json}] [--indent INDENT] [--resolve-refs]
         [--resolve-env | --lock] [-i]
         [config]
```

Позиционные аргументы

config
Файл конфигурации. По умолчанию применяется .config.yaml из каталога KAS_WORK_DIR.

Именованные аргументы

--skip
Пропустить этапы сборки SKIP (по умолчанию список пуст).

--force-checkout
Всегда проверять желаемое значение refspec каждого репозитория, отбрасывая любые локальные изменения. По умолчанию отключено (False).

--update
Вносить (pull) новые изменения (upstream) для желаемого refspec, даже если они внесены (checked out) локально. По умолчанию отключено (False).

--format
Задаёт формат вывода и может принимать значение yaml (по умолчанию) или json.

--indent
Отступ строк (число пробелов в начале), по умолчанию 4.

--resolve-refs
Заменять плавающие ссылки точными SHA. По умолчанию отключено (False).

--resolve-env
Устанавливает вместо принятого по умолчанию env указанное значение. По умолчанию отключено (False).

--lock
Создает файл блокировки с точными SHA. По умолчанию отключено (False).

-i, --inplace
Обновить файл блокировки (требует --lock). По умолчанию отключено (False).

for-all-repos

Выполняет указанную команду для всех извлечённых репозиториев.

```
kas for-all-repos [-h] [--skip SKIP] [--force-checkout] [--update] [-E]
                  [config] command
```

Позиционные аргументы

config
Файл конфигурации. По умолчанию применяется .config.yaml из каталога KAS_WORK_DIR.

command
Строка, указывающая команду для выполнения

Именованные аргументы

--skip
Пропустить этапы сборки SKIP (по умолчанию список пуст).

--force-checkout
Всегда проверять желаемое значение refspec каждого репозитория, отбрасывая любые локальные изменения. По умолчанию отключено (False).

--update
Вносить (pull) новые изменения (upstream) для желаемого refspec, даже если они внесены (checked out) локально. По умолчанию отключено (False).

-E, --preserve-env
Сохранять текущий пользовательский блок окружения. По умолчанию отключено (False).

shell

Запускает интерпретатор команд (shell) в среде сборки.

```
kas shell [-h] [--skip SKIP] [--force-checkout] [--update] [-E] [-k]
          [-c COMMAND]
          [config]
```

Позиционные аргументы

config
Файл конфигурации. По умолчанию применяется .config.yaml из каталога KAS_WORK_DIR.

Именованные аргументы

--skip
Пропустить этапы сборки SKIP (по умолчанию список пуст).

--force-checkout
Всегда проверять желаемое значение refspec каждого репозитория, отбрасывая любые локальные изменения. По умолчанию отключено (False).

--update
Вносить (pull) новые изменения (upstream) для желаемого refspec, даже если они внесены (checked out) локально. По умолчанию отключено (False).

-E, --preserve-env
Сохранять текущий пользовательский блок окружения. По умолчанию отключено (False).

-k, --keep-config-unchanged
Пропустить этапы, изменяющие конфигурацию. По умолчанию отключено (False).

-c, --command
Выполнить указанную команду. По умолчанию команда не задана.

menu

Предоставляет меню настройки конфигурации и запускает сборку выбранных целей.

```
kas menu [-h] [kconfig]
```

Позиционные аргументы

kconfig
Файл Kconfig (по умолчанию Kconfig).

Переменные окружения

KAS_WORK_DIR
Путь к рабочему каталогу kas. По умолчанию текущий рабочий каталог.

KAS_BUILD_DIR
Путь к каталогу сборки, по умолчанию ${KAS_WORK_DIR}/build.

KAS_REPO_REF_DIR
Путь к каталогу образцовых репозиториев, применяемых для клонирования. Чтобы пакет kas находил эти репозитории, они должны именоваться определенным способом - URL репозиториев транслируются, например, https://github.com/siemens/meta-iot2000.git преобразуется в имя github.com.siemens.meta-iot2000.git. Не найденные репозитории будут закрыты. В одном каталоге может работать одновременно несколько экземпляров kas, если базовая файловая система совместима с POSIX.

KAS_DISTRO KAS_MACHINE KAS_TARGET KAS_TASK
Переопределяют соответствующие установки конфигурационного файла.

KAS_PREMIRRORS DISTRO_APT_PREMIRRORS
Задаёт варианты подстановки для URL репозиториев. Подобно bitbake PREMIRRORS, эта переменная состоит из записей, размещённых по одной в строке. Каждая запись задаёт регулярное выражение для сопоставления с URL и отделённую пробелом замену. Например, http://.*.someurl.io/ http://localmirror.net/

SSH_PRIVATE_KEY
Секретный ключ, который следует добавить во внутренний агент ssh (ключ не защищается паролем). Эта переменная полезна для серверов сборки CI, а на обычных машинах (desktop) лучше применять агент ssh, расположенный вне среды kas.

SSH_PRIVATE_KEY_FILE
Путь к файлу с секретным ключом, который следует добавить во внутренний агент ssh (ключ не защищается паролем). Эта переменная полезна для серверов сборки CI, а на обычных машинах (desktop) лучше применять агент ssh, расположенный вне среды kas.

SSH_AUTH_SOCK
Сокет аутентификации SSH, служащий для клонирования через SSH (альтернатива SSH_PRIVATE_KEY или SSH_PRIVATE_KEY_FILE).

DL_DIR SSTATE_DIR TMPDIR
Переменные окружения, передаваемые в среду bitbake.

http_proxy https_proxy ftp_proxy no_proxy
Определяют конфигурацию прокси для bitbake.

GIT_PROXY_COMMAND NO_PROXY
Задаёт прокси для естественной выборки git. Значение NO_PROXY преобразуется в сценарий OE oe-git-proxy.

SHELL
Командный интерпретатор для использования с подключаемым модулем shell.

TERM
Опции терминала для подключаемого модуля shell.

AWS_CONFIG_FILE AWS_SHARED_CREDENTIALS_FILE
Путь к файлу конфигурации и свидетельствам (credential), которые копируются в домашний каталог kas.

GIT_CREDENTIAL_HELPER GIT_CREDENTIAL_USEHTTPPATH
Задает и настраивает вспомогательную функцию (helper) git для свидетельств в *.gitconfig* пользователя kas.

NETRC_FILE

Путь к файлу .netrc, который копируется в домашний каталог kas как .netrc.

CI_SERVER_HOST CI_JOB_TOKEN

Переменные окружения из gitlab CI, если .netrc настроен на разрешение выборки из экземпляра gitlab. Запись добавляется, если указан также файл NETRC_FILE. Отметим, что при наличии в файле записи для данного хоста многие инструменты будут использовать её.

BB_NUMBER_THREADS PARALLEL_MAKE

Переменные окружения для управления одновременной работой.

Версии формата конфигурации

1 (0.10)

Добавлен механизм включения и проверка версии.

2

Версии файлов конфигурации задаются целым числом. Исправлено поведение для включаемых файлов из репозиториев, которые не заданы в текущем файле.

3

Добавлен ключ task, позволяющий указать задачу для выполнения (bitbake -c).

4

Добавлен ключ target, разрешающий список имён целей.

5

Использование целей multiconfig:* автоматически добавляет подобающие записи BBMULTICONFIG в файл local.conf.

6

Ключ env позволяет передать пользовательские переменные окружения для процесса сборки bitbake.

7

Свойство type для repos позволяет указать применяемую систему сборки.

8

Свойство patches для repos позволяет применить к репозиторию дополнительные изменения (patch).

9

Ключ defaults применяется для задания принятого по умолчанию значения свойство репозитория refspec и свойство исправлений repo. Эти значения применяются, если соответствующие свойства не заданы для репозитория или исправлений.

10

Добавлено свойство build_system для выбора системы сборки OE или Isar.

11

Строковый элемент includes теперь указывает путь относительно репозитория. Пути относительно файла поддерживаются с выдачей предупреждения. Файл bblayers.conf создаётся с принятыми по умолчанию значениями переменных BBPATH и BBFILES, которые можно переопределить через bblayers_conf_headers. Ключ menu_configuration сохраняет выбор, сделанный через kas menu, в файле конфигурации (ключ проверяется только этим модулем).

12

Переменные url и path для репозиториев можно переопределить пустыми значениями для переключения между репозиториями с контролем версий и локальными каталогами без контроля версий.

13

Переменные из раздела env могут иметь значение None, задающее их экспорт лишь в «белый список» bb env.

14

Запись overrides на верхнем уровне можно применять для фиксации плавающих refspec репозитория.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru