

Омуралиев Искендер

Система:

Linux pupsik-IdeaPad-3-15ADA6 6.8.0-31-generic #31-Ubuntu SMP
PREEMPT_DYNAMIC Sat Apr 20 00:40:06 UTC 2024 x86_64 x86_64 x86_64
GNU/Linux

Версия дистрибутива :

Ubuntu 24.04

Пояснительная записка по заданию к стажировке

Задание № 1: *Разобраться, как настроить окружение для сборки Yocto (дистрибутив Poky). Собрать простейший образ (цель core-image-minimal). Убедиться, что он запускается в QEMU. Используйте версию Yocto: Kirkstone.*

На сайте <https://www.yoctoproject.org/> по документации было настроено окружение для сборки дистрибутива .

1) Установлены необходимые пакеты для сборки и генерация локали

```
> sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debiанutils iputils-ping python3-git python3-jinja2 python3-subunit zstd liblz4-tool file locales libacl1
```

```
> sudo locale-gen en_US.UTF-8
```

2) Было произведено клонирование репозитория Yocto с дистрибутивом Poky и последующее переключение на нужную нам ветку kirkstone

```
> git clone git://git.yoctoproject.org/poky
```

```
> cd poky
```

```
> git branch -a
```

```
> git checkout kirkstone
```

```
> git pull
```

3)Инициализация среды сборки .Запустили сценарий настройки среды oe-init-build-env, чтобы определить среду сборки Yocto Project на вашем хосте сборки.

```
> cd poky  
> source oe-init-build-env
```

Была создана директория build в которой и происходит сборка Позже, когда сборка завершится, каталог сборки будет содержать все файлы, созданные во время сборки.

4) Сборка образа

```
> bitbake core-image-minimal
```

5) Инициализация и запуск в QEMU

После создания этого конкретного образа вы можете запустить QEMU, который представляет собой быстрый эмулятор, входящий в состав проекта Yocto:

```
> runqemu qemu86-64
```

Был написан скрипт script.py для автоматической сборки на языке python (см.в приложении)

Выводы: В ходе выполнения данного пункта были получены знания и понятия о сборке собственного образа и запуск его в эмуляторе QEMU

Задание № 2: Реализовать автоматизацию сборки через Docker. В контейнере необходимо реализовать сборку образа и запуск образа в QEMU. Выбор между сборкой и запуском необходимо реализовать через аргументы, передаваемые контейнеру при запуске. Лучшие всего использовать `docker volume`, чтобы собирать все исходные файлы в директории на хост-машине (то есть, загрузка и сборка образа осуществляется в `docker volume`). В качестве базового Docker образа используйте: `ubuntu:20.04`.

В начале , был установлен docker и необходимый для выполнения задания контейнер ubuntu 20.04 с сайта <https://hub.docker.com/>

В докер файле были написаны инструкции для автоматической сборки по примеру из первого задания .А так же добавлена логика выбора аргумента запуска или сборки . И сборка осуществляется через `docker volume` в директорию `volume` на основной машине.

См. Dockerfile файл в приложении

1) После написания Dockerfile был он собран при помощи :

```
> sudo docker build --build-arg "USER_NAME=user" -t yocto-image .
```

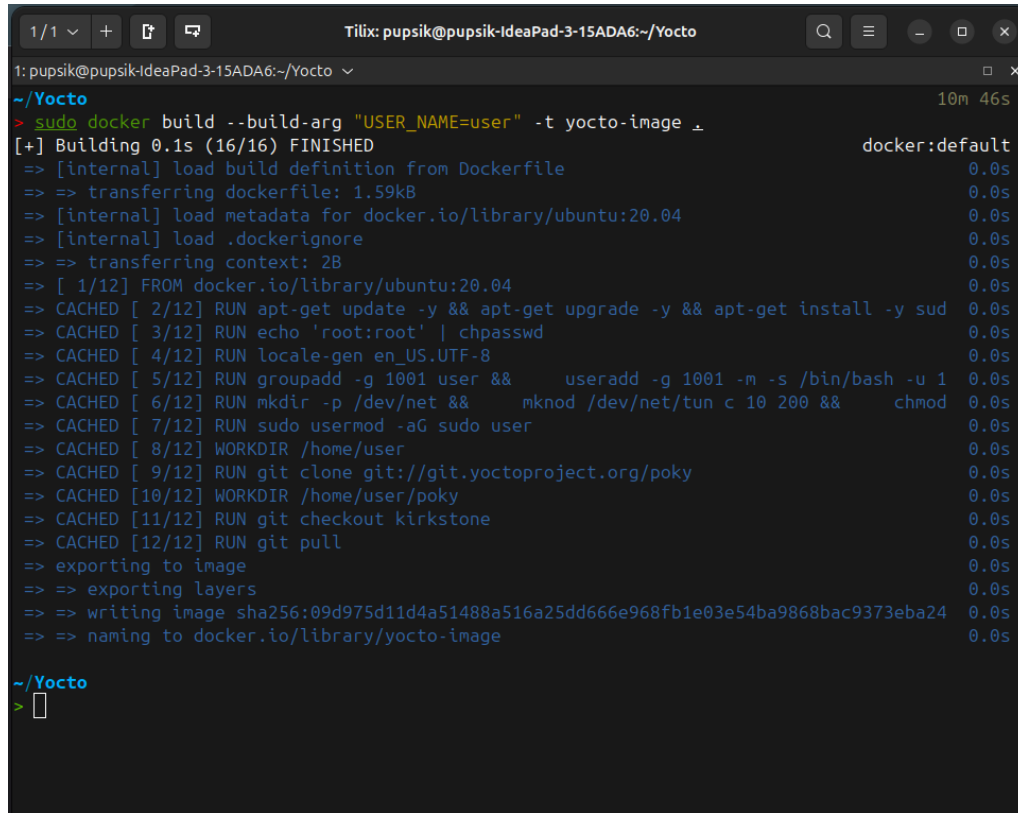
1. `--build-arg "USER_NAME=user"`: Этот флаг используется для передачи аргументов сборки (build arguments) в процесс сборки. Аргументы сборки могут быть определены в Dockerfile с использованием директивы ARG, а затем использованы внутри Dockerfile или в инструкциях RUN, CMD, ENTRYPOINT и т.д. В данном случае, `"USER_NAME=user"` передает значение user в аргумент сборки USER_NAME, который должен быть определен в Dockerfile.

2. `-t yocto-image`: Флаг `-t` используется для задания имени (и тега) образа, который будет создан после успешной сборки. В данном случае, имя образа будет `yocto-image`.

3. `.`(точка): Этот символ представляет собой текущую директорию в командной строке. Docker использует эту директорию в качестве контекста

сборки, то есть всех файлов и каталогов, доступных во время сборки, будут относиться к этому местоположению. Dockerfile должен находиться в корне этого контекста.

2)



```
1/1 v + [T] [Q] Tilix: pupsik@pupsik-IdeaPad-3-15ADA6:~/Yocto
1: pupsik@pupsik-IdeaPad-3-15ADA6:~/Yocto
~/Yocto 10m 46s
> sudo docker build --build-arg "USER_NAME=user" -t yocto-image .
[+] Building 0.1s (16/16) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 1.59kB 0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [ 1/12] FROM docker.io/library/ubuntu:20.04 0.0s
=> CACHED [ 2/12] RUN apt-get update -y && apt-get upgrade -y && apt-get install -y sudo 0.0s
=> CACHED [ 3/12] RUN echo 'root:root' | chpasswd 0.0s
=> CACHED [ 4/12] RUN locale-gen en_US.UTF-8 0.0s
=> CACHED [ 5/12] RUN groupadd -g 1001 user && useradd -g 1001 -m -s /bin/bash -u 1 0.0s
=> CACHED [ 6/12] RUN mkdir -p /dev/net && mknod /dev/net/tun c 10 200 && chmod 0.0s
=> CACHED [ 7/12] RUN sudo usermod -aG sudo user 0.0s
=> CACHED [ 8/12] WORKDIR /home/user 0.0s
=> CACHED [ 9/12] RUN git clone git://git.yoctoproject.org/poky 0.0s
=> CACHED [10/12] WORKDIR /home/user/poky 0.0s
=> CACHED [11/12] RUN git checkout kirkstone 0.0s
=> CACHED [12/12] RUN git pull 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:09d975d11d4a51488a516a25dd666e968fb1e03e54ba9868bac9373eba24 0.0s
=> => naming to docker.io/library/yocto-image 0.0s

~/Yocto
> 
```

Dockerfile был собран теперь , мы должны запустить Dockerfile с режимом build чтобы началось выполнение инструкций и сама сборка образа внутри контейнера

```
> sudo docker run -v yocto-dir:/home/user -e mode=build yocto-image
```

mode = build как раз и есть тот режим отвечает за сборку проекта

```

~/yocto
> sudo docker run -v yocto-dir:/home/user -e mode=build yocto-image

### Shell environment set up for builds. ###

You can now run 'bitbake <target>'

Common targets are:
  core-image-minimal
  core-image-full-cmdline
  core-image-sato
  core-image-weston
  meta-toolchain
  meta-ide-support

You can also run generated qemu images with a command like 'runqemu qemux86'

Other commonly useful commands are:
- 'devtool' and 'recipetool' handle common recipe tasks
- 'bitbake-layers' handles common layer tasks
- 'oe-pkgdata-util' handles common target package tasks
Loading cache...done.
Loaded 1644 entries from dependency cache.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "2.0.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "universal"
TARGET_SYS           = "x86_64-poky-linux"
MACHINE              = "qemux86-64"
DISTRO               = "poky"
DISTRO_VERSION        = "4.0.18"
TUNE_FEATURES        = "m64 core2"
TARGET_FPU            = ""
meta
meta-poky
meta-yocto-bsp        = "kirkstone:e575d02196b0013c25f8064e4dbe3fc2a0ef72d0"

Initialising tasks...done.
Sstate summary: Wanted 0 Local 0 Mirrors 0 Missed 0 Current 1397 (0% match, 100% complete)
NOTE: Executing Tasks
NOTE: Tasks Summary: Attempted 3595 tasks of which 3595 didn't need to be rerun and all succeeded.
Build Yocto end's successful!\nYou can run docker with mode=run

```

3) Далее мы запускаем докер с режимом run

```
> sudo docker run -v yocto-dir:/home/user -e mode=run -it yocto-image
```

Действительно, что и требовалось доказать образ запускается в qemu как и было задумано

```
Sat Jun  1 18:17:32 UTC 2024
INIT: Entering runlevel: 5
Configuring network interfaces... ip: RTNETLINK answers: File exists
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 4.0.18 qemu86-64 /dev/ttyS0
qemu86-64 login: root

root@qemu86-64:~# ls
root@qemu86-64:~#
```

Выводы : В ходе выполнения задания был получен навык работы с Docker и знания работы с контейнером и создания своей директории docker volume на хост машине .

Задание № 3: Добавить в образ программу yadro_hello. Программа должна выводить в поток стандартного вывода строку “Hello from my own program!”. Программа должна быть написана на языке C. Добавление программы необходимо осуществить посредством создания слоя Yocto.

Для начала был создан новый слой hello

> bitbake-layers create-layer meta-hello

Вырезка из терминального окна контейнера :

```
1 / 1  +  ... 0a4: ~/poky/meta-hello/recipes-example/hello/files  🔍  ☰  -  □  ×

1: user@d97e64a640a4: ~/poky/meta-hello/recipes-example/hello/files  ~
user@d97e64a640a4:~/poky$ sl
bash: sl: command not found
user@d97e64a640a4:~/poky$ ls
LICENSE                README.OE-Core.md      bitbake                meta-poky
LICENSE.GPL-2.0-only  README.hardware.md    build                  meta-selftest
LICENSE.MIT            README.md              contrib                meta-skeleton
MAINTAINERS.md         README.poky.md         documentation          meta-yocto-bsp
MEMORIAM               README.qemu.md         meta                   oe-init-build-env
Makefile               SECURITY.md             meta-hello             scripts
user@d97e64a640a4:~/poky$ cd meta-hello/
user@d97e64a640a4:~/poky/meta-hello$ ls
COPYING.MIT  README  conf  recipes-example
user@d97e64a640a4:~/poky/meta-hello$ cd recipes-example/
user@d97e64a640a4:~/poky/meta-hello/recipes-example$ ls
example  hello
user@d97e64a640a4:~/poky/meta-hello/recipes-example$ cd hello/
user@d97e64a640a4:~/poky/meta-hello/recipes-example/hello$ ls
files  hello.bb
user@d97e64a640a4:~/poky/meta-hello/recipes-example/hello$ cd files/
user@d97e64a640a4:~/poky/meta-hello/recipes-example/hello/files$ ls
hello.c
user@d97e64a640a4:~/poky/meta-hello/recipes-example/hello/files$
```

В паке build/conf/locale.conf в самый конец мы добавили

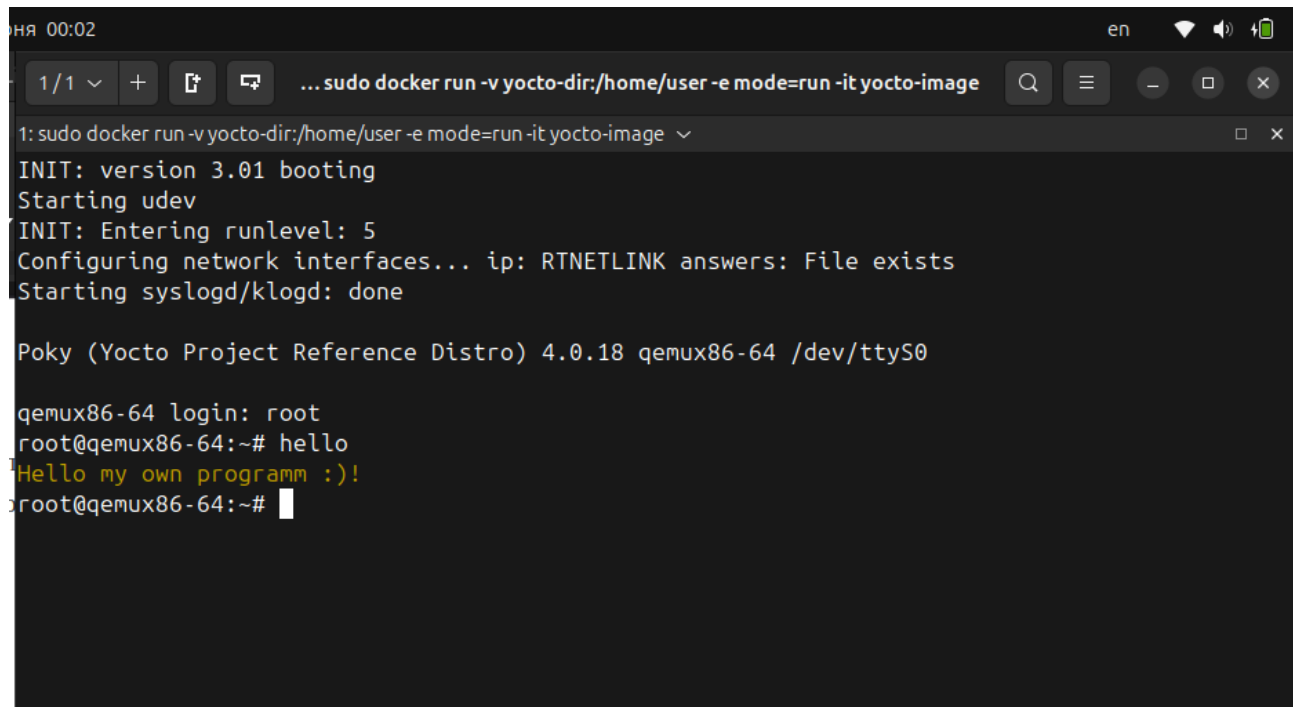
IMAGE_INSTALL:append = "hello"

Дали знать о новом слое конфигурационному файлу

Далее мы через битбэйк собрали данный слой

```
user@d97e64a640a4:~/poky$ bitbake hello
```

Выходим из контейнера через quit и пересобираем докер файл , запускаем с mode=run и попадаем в qemu пишем команду которую мы сделали на языке Си .Наблюдаем вывод



```
1: sudo docker run -v yocto-dir:/home/user -e mode=run -it yocto-image
INIT: version 3.01 booting
Starting udev
INIT: Entering runlevel: 5
Configuring network interfaces... ip: RTNETLINK answers: File exists
Starting syslogd/klogd: done

Poky (Yocto Project Reference Distro) 4.0.18 qemu86-64 /dev/ttyS0

qemu86-64 login: root
root@qemu86-64:~# hello
Hello my own programm :)!
root@qemu86-64:~#
```

См. Файлы в приложении (hello.c, hello.bb)

