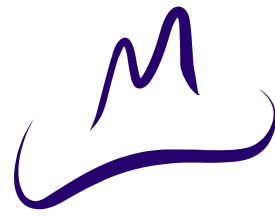


MSE 07

Iteration №2





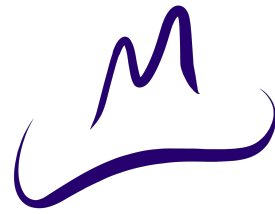
Состав команды

Магистр

- 1) Тиняков Сергей Алексеевич (9304)

Бакалавры

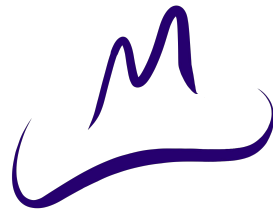
- 2) Кондратенко Константин Евгеньевич (1384)
- 3) Денисова Ольга Константиновна (1381)
- 4) Бутыло Егор Алексеевич (1303)
- 5) Андреева Елизавета Алексеевна (1303)



План на текущую итерацию

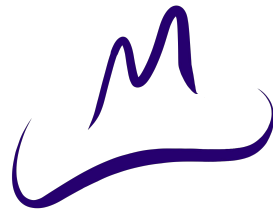
В итерации 1 было зафиксировано, что планами на текущую являются:

- 1) Оформить текущие наработки в виде документации
- 2) Сбор и подробный анализ полученных данных



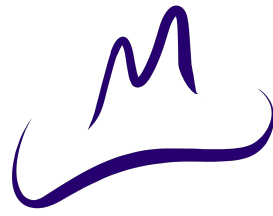
Промежуточный результат (1)

- Задокументированы следующие темы прошлой итерации:
 - сбор статистики сборщиком
 - механизмы формирования системных зависимостей сборки
 - описаны стандартные утилиты трассировки
 - описаны простые примеры использования библиотеки psutil
- Работа с docker оберткой
 - добавлены зависимости от библиотек трассировки
 - добавлен docker-compose
 - добавлена точка входа в проект
 - добавлен общая с хостом папка для сохранения результатов сборки + логов
 - добавлена инструкция сборки контейнеров



Промежуточный результат (2)

- Проведены исследования создания кэша сборки
 - рассмотрено, как создается кэш (концептуально)
 - рассмотрена возможность организации кэш серверов
 - рассмотрено, как осуществляется подгрузка кэше - через сверку хэш сумм
- Реализован парсер стандартный логов и итерация по ним
 - реализован парсер логов
 - исследована причина отсутствия фиксации логов некоторых задач
 - реализован класс-итератор по созданным директориям и сборщик данных



План на следующую итерацию

- Детально проанализировать процесс кэширования сборки образа (триггеры формирования, возможности конфигурации)
- Окончить обертку проекта в образ - оформление всех доступов для утилит профилировки
- Произвести первичный анализ логов на выявление явных симптомов потребления ресурсов

Screencast



Изучение объема занимаемого места на ПК после сборки образа

```
elizaveta@elizaveta-HP-ProBook-650-G8-Notebook-PC:~/poky$ cd /build
bash: cd: /build: Нет такого файла или каталога
elizaveta@elizaveta-HP-ProBook-650-G8-Notebook-PC:~/poky$ du -h --max-depth=1 build
53G      build/tmp
7,8G     build/downloads
3,0M     build/cache
36K      build/conf
4,2G     build/sstate-cache
65G      build
elizaveta@elizaveta-HP-ProBook-650-G8-Notebook-PC:~/poky$
```

Screencast



Изучение занимаемых объемов в рабочей директории

```
elizaveta@elizaveta-HP-ProBook-650-G8-Notebook-PC:~/poky$ du -h --max-depth=1 build/tmp
4,0K    build/tmp/hosttools
2,0G    build/tmp/deploy
638M    build/tmp/log
13M     build/tmp/sstate-control
29M     build/tmp/cache
48G     build/tmp/work
20M     build/tmp/buildstats
56M     build/tmp/stamps
2,5G    build/tmp/work-shared
16M     build/tmp/sysroots-components
7,1M    build/tmp/pkgdata
114M    build/tmp/sysroots-uninative
53G     build/tmp
elizaveta@elizaveta-HP-ProBook-650-G8-Notebook-PC:~/poky$
```




Screenecast

Пошаговое описание процесса сборки образов

Yocto system image build

В этом файле рассмотрены следующие аспекты сборки Yocto:

- Как в итоговый образ добавляются файлы/библиотеки/тд
- Где это происходит в коде
- В какой момент времени это происходит
- Используется ли кэширование
- Отличия слоев и классов

Screencast



Описана механика кэширование сборки и проведены эксперименты

Yocto/Bitbake caching overview

🔗 Общие сведения

- В Yocto (bitbake) кэшируются исходные файлы, промежуточные результаты сборки (и результаты предыдущих сборок), зависимости, а также скачанный до момента текущей сборки готовые утилиты (которые используются в сборке, но сами эти утилиты не собираются из исходников, а скачиваются в виде готовых инструментов).
- Имеется возможность организации кэш-серверов (то есть определенное место, в котором заранее собраны и кэшированы какие-то блоки), в этом случае необходимо определенным образом настроить сборку - как это делается описано в [источнике 1](#) - частным случаем может быть локальная сборка в директории А, к кэшу которой предоставляется доступ из директории Б, в которой планируется запуск сборки.
- По умолчанию кэш подгружается из папки sstate-cache (если пересобираем образ).
- При пересборке образа происходит сравнение hash сумм для кэшированных данных - так выносится решение об использовании кэша предыдущих сборок.
- Кэш формируется по мере выполнения сборки - выполнили задачу - кэшировали
- С помощью флагов можно настроить - какие данные хотим кэшировать.

```
Loading cache: 100% |#####| Time: 0:00:00
```

```
Loaded 1849 entries from dependency cache.
```

```
NOTE: Resolving any missing task queue dependencies
```

Начало работы над получением системной переменной WORKDIR

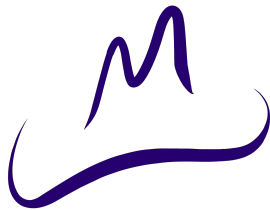
Переменную WORKDIR необходимо знать, если мы хотим корректно обрабатывать какие-либо данные из вышеперечисленных, так как иногда WORKDIR определяется нетривиально, как, например, здесь:

```
user@yadrolab-wp2:~/dok/poky$ bitbake -e gcc-source-13.2.0 | grep ^WORKDIR=  
WORKDIR="/home/user/dok/poky/build/tmp/work-shared/gcc-13.2.0-r0"
```

Данная директория под названием /work-shared/gcc-13.2.0-r0 является рабочим каталогом для рецепта gcc-source-13.2.0, хотя название рецепта нигде не упомянуто, и, без знания WORKDIR мы бы не знали, где искать эти данные. А также, исходя из названия директории, есть опасность перепутать этот каталог, например, с рабочим каталогом для gcc:

```
user@yadrolab-wp2:~/dok/poky$ bitbake -e gcc | grep ^WORKDIR=  
WORKDIR="/home/user/dok/poky/build/tmp/work/core2-64-poky-linux/gcc/13.2.0"
```

Screencast



Добавление зависимостей, общей с хостом папки и compose

```
(base) oumua@oumua:~/yadro/os_profiling/src/yocto-build$ docker-compose build --build-arg UID=$(id -u) --build-arg GID=$(id -g)
Building yocto_project
+ Building 1.5s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.82kB
=> [internal] load metadata for docker.io/library/ubuntu:20.04
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/ubuntu:20.04@sha256:80ef4a44043dec4490506e6cc4289eeda2d106a70148b74b5ae91ee670e9c35d
=> CACHED [2/5] RUN apt-get update && DEBIAN_FRONTEND=noninteractive apt-get -y install gawk wget git
=> CACHED [3/5] RUN python3 -m pip install -U psutil numpy matplotlib
=> CACHED [4/5] RUN rm /bin/sh && ln -s bash /bin/sh && groupadd -g 1000 yocto_user && useradd -u 1000 -g 1000 -ms /bin/bash
=> CACHED [5/5] WORKDIR /home/yocto_user/project
=> exporting to image
=> => exporting layers
=> => writing image sha256:cb6a44d0820889e744692fbd4bdc0a6b91a79edacb657360a8b0f1692d671e63
=> => naming to docker.io/library/yocto-image
```

```
(base) oumua@oumua:~/yadro/os_profiling/src/yocto-build$ docker-compose up
Recreating yocto_project ... done
Attaching to yocto_project
yocto_project | Cloning into 'poky'...
```

Вторая итерация. Команда MSE 07

