Тестрирование программного обеспечения
Лабораторная работа 3
Составление тест-плана
тестирования OpenBMC

Дорогин Никита ИП-312

ВАРИАНТ 8

ЧАСТЬ 1: Изучение OpenBMC.

Основные функции

Управление питанием.
openbmc-systemd

Содержит следующие функции (цели):

obmc-host-start@.target — активирует загрузку системы (запуск хоста)

obmc-host-startmin@.target — представляет собой минимум систем и служб, необходимых для запуска.

obmc-chassis-poweron@0.target — собирает службы, связанные с запуском хоста, то есть после выполнения этой функции при следующем запуске хоста эти службы также будут запущены.

obmc-host-shutdown@.target — «мягкое» выключение, хосту отправляется запрос и даётся определённое время на выполнение.

obmc-chassis-hard-poweroff@.target — принудительно остановит «мягкое» выключение и немедленно обрежет питание хосту.

obmc-host-reboot@.target — выполнит «мягкое» выключение и сразу после снова запустит хоста.

obmc-host-quiesce@0.target — запускается при аварийных сценариях (критических ошибках: аппаратных, питания, перегрев и.т.д), переводит систему в «состояние покоя», когда продолжение нормальной работы невозможно или опасно. Хост переходит в режим ожидания действий извне.

obmc-chassis-emergency-poweroff@.target — представляет собой оболочку для obmc-chassis-hard-poweroff@.target и obmc-host-quiesce@0.target. Запускается

при обнаружении приложениями критических ошибок питания или проблем с температурой, требующий немедленного завершения работы системы.
Эта цель конфликтует с некоторыми не-критичными целями, чтобы только самые критичные продолжали свою работу.

Мониторинг аппаратного обеспечения.
sensor-architecture

BMCWeb Redfish поддерживает возвращение информации о состоянии сенсоров. Он опирается на два типа ObjectMapper ассоциаций, чтобы найти в D-Bus необходимую информацию.

Тип 1: Связь chassis с сенсорами
Сенсоры группируются по шасси в Redfish. Для связи шасси со всеми сенсорами внутри этого шасси используется ассоциация ObjectMapper. Это включает в себя сенсоры всего оборудования, которое считается находящимся внутри шасси.

Например, в одном шасси могут находиться:

- два датчика вентиляторов
- датчик температуры окружающей среды
- датчик выходного напряжения VRM (модуля регулирования напряжения)

Названия ассоциаций

- "all_sensors"
  Содержит список всех сенсоров для данного шасси.
- "chassis"
  Содержит шасси, связанные с данным сенсором.

Пример ассоциаций:

- /xyz/openbmc_project/inventory/system/chassis/all_sensors
  - "endpoints" property contains
    - /xyz/openbmc_project/sensors/fan_tach/fan0_0
    - /xyz/openbmc_project/sensors/fan_tach/fan0_1
    - /xyz/openbmc_project/sensors/temperature/ambient
    - /xyz/openbmc_project/sensors/voltage/p0_vdn_voltage
- /xyz/openbmc_project/sensors/fan_tach/fan0_0/chassis
  - "endpoints" property contains
    - /xyz/openbmc_project/inventory/system/chassis

Тип 2: Связь оборудования с сенсорами.

Сенсор обычно связан с компонентом аппаратного обеспечения низкого уровня, таким как вентилятор, блок питания, модуль регулирования напряжения (VRM) или процессор (CPU). Поддержка сенсоров в Redfish может получать следующую информацию от связанного аппаратного компонента:

- Присутствие (интерфейс Inventory.Item)
- Рабочее состояние (интерфейс OperationalStatus)
- Производитель, Модель, Номер детали, Серийный номер (интерфейс Decorator.Asset)

По этой причине ассоциация ObjectMapper используется для связи компонента аппаратного обеспечения низкого уровня с его сенсорами. Например, модуль VRM процессора может иметь датчики температуры и выходного напряжения, или двухроторный вентилятор может иметь два датчика частоты вращения (тахометра).

Пути объектов D-Bus

Ассоциация связывает вместе следующие пути объектов D-Bus:

- Путь к объекту инвентаря компонента аппаратного обеспечения низкого уровня
- Список путей к объектам сенсоров, связанных с этим аппаратным компонентом

Имена ассоциаций

- "sensors"

  Содержит список сенсоров для этого компонента аппаратного обеспечения низкого уровня

- "inventory"

  Содержит компонент инвентаря аппаратного обеспечения низкого уровня для этого сенсора

Примеры ассоциаций

/xyz/openbmc_project/inventory/system/chassis/motherboard/fan0/sensors

Свойство "endpoints" содержит:

/xyz/openbmc_project/sensors/fan_tach/fan0_0

/xyz/openbmc_project/sensors/fan_tach/fan0_1

/xyz/openbmc_project/sensors/fan_tach/fan0_0/inventory

Свойство "endpoints" содержит:

/xyz/openbmc_project/inventory/system/chassis/motherboard/fan0

Удаленный доступ через IPMI/Redfish.
IPMITOOL-cheatsheet, REDFISH-cheatsheet

IPMI:
Доступ уже имеется и для управления необходимо только знать команды, например:

```
ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power status

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power on

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power off

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power reset

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power reset

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power cycle

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power soft

ipmitool -C 17 -H "$BMC_IP" -I lanplus -U "$BMC_USER" -P "$BMC_PASSWD" power diag
```

Установка соединения через Redfish:

Метод 1:

```
export bmc=xx.xx.xx.xx
curl --insecure -H "Content-Type: application/json" -X POST -D headers.txt
https://${bmc}/redfish/v1/SessionService/Sessions -d    '{"UserName":"root",
"Password":"0penBmc"}'
```

Будет создан файл headers.txt. В этом файле нужно найти `"X-Auth-Token"` и сохранить его в переменную окружения:

```
export bmc_token=<token>
```

Метод 2:
export bmc=xx.xx.xx.xx
export token=`curl -k -H "Content-Type: application/json" -X POST https://${bmc}/login -d '{"username" :  "root", "password" :  "0penBmc"}' | grep token | awk '{print $2;}' | tr -d ""`
curl -k -H "X-Auth-Token: $token" https://${bmc}/redfish/v1/...

Логирование событий.
host-managment

Структура журнала событий находится в иерархии `/xyz/openbmc_project/logging/entry`. Каждое событие представляет собой отдельный объект в этой структуре, ссылающийся на номер.

BMC и встроенное ПО хоста на серверах с архитектурой POWER могут передавать журналы событий в BMC. Обычно эти журналы событий передаются в случаях, когда встроенное ПО хоста не может запустить ОС или не может надежно записывать события в ОС.

Свойства, связанные с записью в журнале событий:

- Message (Сообщение): Тип события (например, `"xyz.openbmc_project.Inventory.Error.NotPresent"`).
- Resolved (Решено): Указывает, была ли устранена проблема.
- Severity (Важность): Уровень серьезности проблемы (`"Info"`, `"Error"` и т.д.).
- Timestamp (Временная метка): Дата события в формате Epoch time.
- Associations (Ассоциации): URI, ссылающийся на вышедший из строя компонент в инвентаре.

Получить список всех зарегистрированных событий:

```
curl -k -H "X-Auth-Token: $token"
https://${bmc}/xyz/openbmc_project/logging/entry
{
  "data": [
    "/xyz/openbmc_project/logging/entry/3",
    "/xyz/openbmc_project/logging/entry/2",
    "/xyz/openbmc_project/logging/entry/1",
    "/xyz/openbmc_project/logging/entry/7",
    "/xyz/openbmc_project/logging/entry/6",
    "/xyz/openbmc_project/logging/entry/5",
    "/xyz/openbmc_project/logging/entry/4"
  ],
  "message": "200 OK",
```

```
  "status": "ok"
}
```

Прочитать конкретную запись в журнале событий (например, запись 1):

```
curl -k -H "X-Auth-Token: $token"
https://${bmc}/xyz/openbmc_project/logging/entry/1
{
  "data": {
    "AdditionalData": [
      "_PID=183"
    ],
    "Id": 1,
    "Message": "xyz.openbmc_project.Common.Error.InternalFailure",
    "Purpose":
"xyz.openbmc_project.Software.Version.VersionPurpose.BMC",
    "Resolved": false,
    "Severity": "xyz.openbmc_project.Logging.Entry.Level.Error",
    "Timestamp": 1563191362822,
    "Version": "2.8.0-dev-132-gd1c1b74-dirty",
    "associations": []
  },
  "message": "200 OK",
  "status": "ok"
}
```

Удалить запись из журнала событий (в этом примере запись 1):

```
curl -k -H "X-Auth-Token: $token" -H "Content-Type:
application/json" -X POST \
-d '{"data" : []}'
https://${bmc}/xyz/openbmc_project/logging/entry/1/action/Delete
```

Очистить весь журнал событий:

```
curl -k -H "X-Auth-Token: $token" -H "Content-Type:
application/json" -X POST \
-d '{"data" : []}'
https://${bmc}/xyz/openbmc_project/logging/action/DeleteAll
```

ЧАСТЬ 2: Разработка тест-плана.

1. Цели тестирования:
   - Проверить корректность работы основных функций OpenBMC.
   - Убедиться, что система соответствует требованиям.

2. Объем тестирования:
   - Управление питанием.
   - Мониторинг аппаратного обеспечения.

- IPMI.
- Логирование.

3. Подходы и методы тестирования:
   - Функциональное тестирование.

4. Ресурсы:
   - ПК с операционной системой Linux, на который установлен образ OpenBMC.
   - Тестовое оборудование: вентиляторы.
   - Инструменты для тестирования: IPMI-клиент.

5. График выполнения тестов:
   - Управление питанием — 30.09.2025 23:00 - 23:15
   - Мониторинг аппаратного обеспечения — 30.09.2025 23:20 - 23:35
   - IPMI — 30.09.2025 23:20 - 23:35
   - Логирование — 30.09.2025 23:40 — 23:55

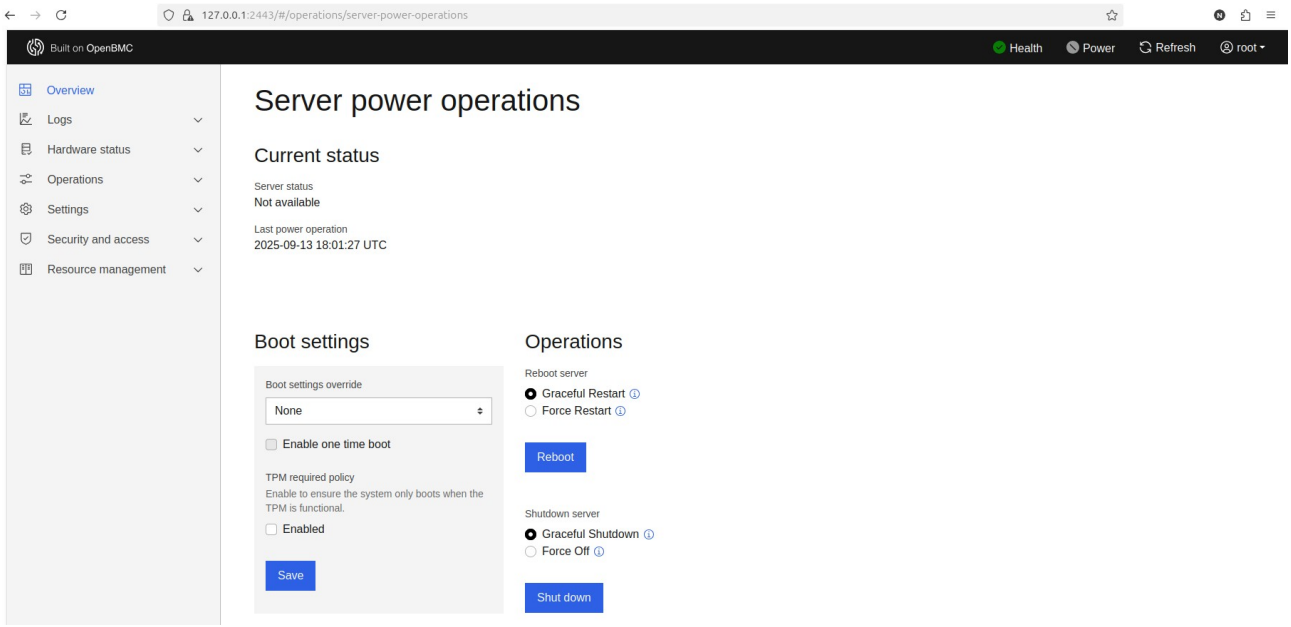
6. Критерии начала и завершения тестирования:
   - Критерии начала: все оборудование настроено, тест-кейсы подготовлены.
   - Критерии завершения: все тест-кейсы проведены

ЧАСТЬ 3: Составление тест-кейсов.

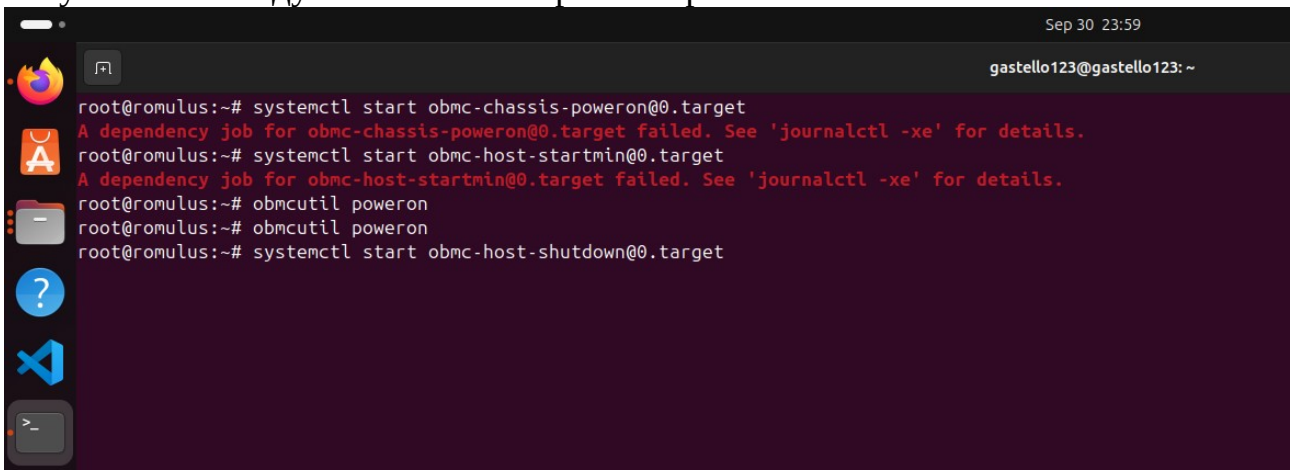
1. Управление питанием:
   - Выключение сервера.

У нас открыт в терминале образ OpenBMC, мы зашли туда в качестве root, а также в браузере открыт web-ui, так же зашли в качестве root.

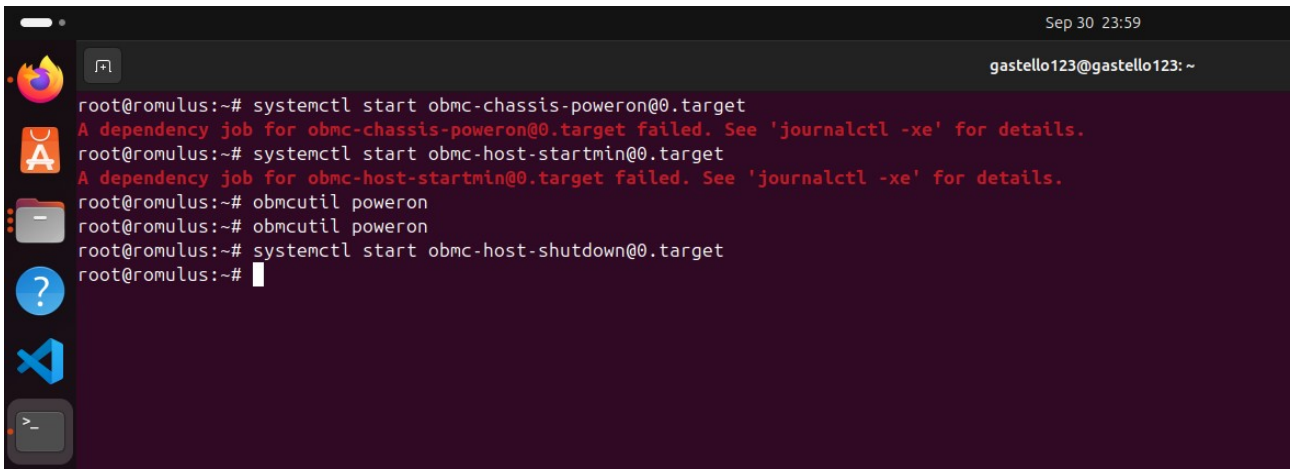
Статус сервера — недоступен. Попытки запустить его через obmc-host-start@.target не увенчались успехом из-за аппаратной неисправности.
Поэтому для включения используем obmcutil poweron получаем тот статус, что имеем.

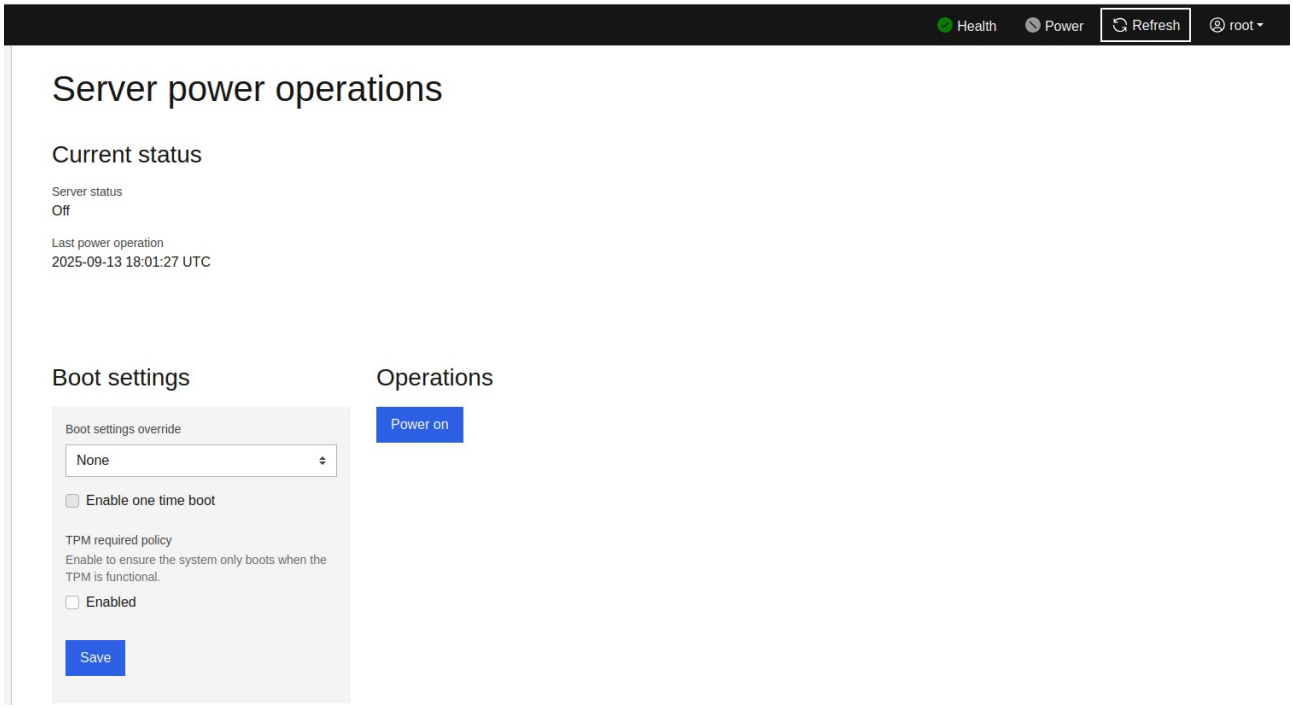


Запускаем команду «мягкого» завершения работы.



Проходит несколько секунд и завершение работы выполнено.






```
root@romulus:~# busctl get-property xyz.openbmc_project.State.Host \
>     /xyz/openbmc_project/state/host0 \
>     xyz.openbmc_project.State.Host CurrentHostState
s "xyz.openbmc_project.State.Host.HostState.Off"
root@romulus:~# busctl get-property xyz.openbmc_project.State.Chassis \
>     /xyz/openbmc_project/state/chassis0 \
>     xyz.openbmc_project.State.Chassis CurrentPowerState
s "xyz.openbmc_project.State.Chassis.PowerState.Off"
root@romulus:~#
```

obmc-chassis-hard-poweroff@.target также работает, но с более короткой паузой.

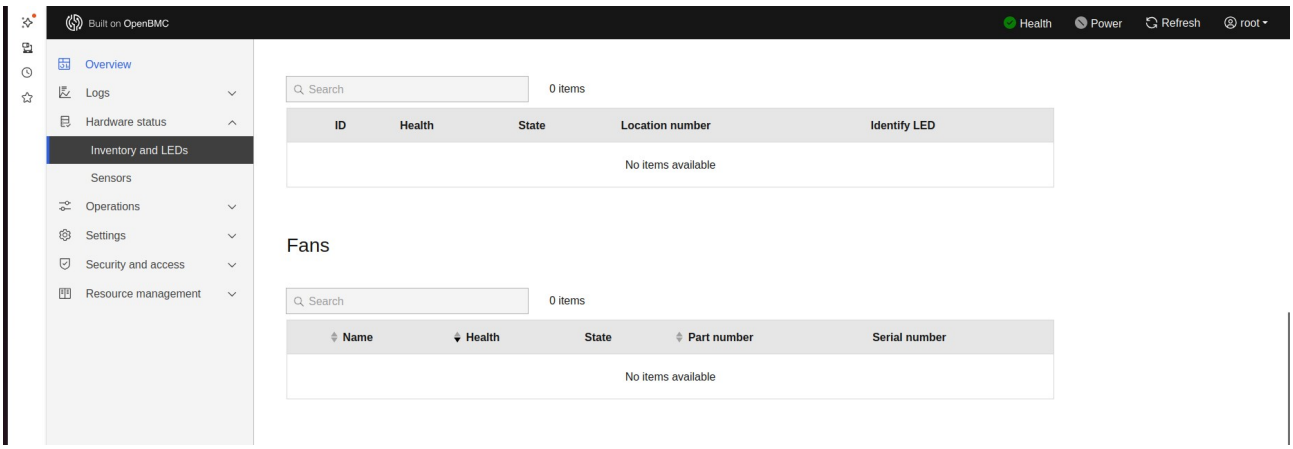
```
root@romulus:~# systemctl start obmc-chassis-hard-poweroff@0.target
root@romulus:~# busctl get-property xyz.openbmc_project.State.Host     /xyz/openbmc_project/state/host0     xyz.openbmc_project.State.Host CurrentHostState
s "xyz.openbmc_project.State.Host.HostState.Off"
root@romulus:~# busctl get-property xyz.openbmc_project.State.Chassis     /xyz/openbmc_project/state/chassis0     xyz.openbmc_project.State.Chassis CurrentPowerState
s "xyz.openbmc_project.State.Chassis.PowerState.Off"
root@romulus:~#
```

Тест-кейс выполнен!

2. Мониторинг аппаратного обеспечения:
    - Проверка состояния вентиляторов.

Для начала нужно проверить, есть ли вентиляторы.

```
root@romulus:~# busctl tree xyz.openbmc_project.Hwmon --no-pager | grep fan
root@romulus:~# find /xyz/openbmc_project/sensors -name "*fan*" 2>/dev/null
root@romulus:~# busctl tree | grep -i sensor
        |- /org/freedesktop/systemd1/unit/phosphor_2dreset_2dsensor_2dstates_400_2eservice
        |- /org/freedesktop/systemd1/unit/sensor_2dmonitor_2eservice
        |- /org/freedesktop/systemd1/unit/system_2dphosphor_5cx2dreset_5cx2dsensor_5cx2dstates_2eslice
    `- /xyz/openbmc_project/sensors
      `- /xyz/openbmc_project/sensors/voltage
        `- /xyz/openbmc_project/sensors/voltage/vbat
    `- /xyz/openbmc_project/sensors
      `- /xyz/openbmc_project/sensors/fan_tach
        |- /xyz/openbmc_project/sensors/fan_tach/fan0
        |- /xyz/openbmc_project/sensors/fan_tach/fan1
        `- /xyz/openbmc_project/sensors/fan_tach/fan2
```



У нас что-то не так с Hwmon поэтому вентиляторы не отображаются ни в терминале, ни в web-ui.

С помощью команды busctl list | grep -i hwmon найдём сервис, который реально отвечает за вентиляторы. И уже с него проверяем. Находим 3 вентилятора.

```
root@romulus:~# busctl tree xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 --no-pager
`- /xyz
  `- /xyz/openbmc_project
    `- /xyz/openbmc_project/sensors
      `- /xyz/openbmc_project/sensors/fan_tach
        |- /xyz/openbmc_project/sensors/fan_tach/fan0
        |- /xyz/openbmc_project/sensors/fan_tach/fan1
        `- /xyz/openbmc_project/sensors/fan_tach/fan2
root@romulus:~#
```

Пытаемся посмотреть количество оборотов в минуту у каждого вентилятора и получаем код -110, означающий ошибку чтения датчика.

```
root@romulus:~# busctl get-property xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 /xyz/openbmc_project/sensors/fan_tach/fan0 xyz.ope
nbmc_project.Sensor.Value Value
d -110
root@romulus:~# busctl get-property xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 /xyz/openbmc_project/sensors/fan_tach/fan1 xyz.ope
nbmc_project.Sensor.Value Value
d -110
root@romulus:~# busctl get-property xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 /xyz/openbmc_project/sensors/fan_tach/fan2 xyz.ope
nbmc_project.Sensor.Value Value
d -110
root@romulus:~# busctl introspect xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 /xyz/openbmc_project/sensors/fan_tach/fan0 xyz.openb
mc_project.Sensor.Threshold.Warning
NAME TYPE SIGNATURE RESULT/VALUE FLAGS
root@romulus:~# busctl introspect xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 /xyz/openbmc_project/sensors/fan_tach/fan1 xyz.openb
mc_project.Sensor.Threshold.Warning
NAME TYPE SIGNATURE RESULT/VALUE FLAGS
root@romulus:~# busctl introspect xyz.openbmc_project.Hwmon-be3af85815c942eb953efaea0a3d08f00d83e8114bc726ee0719925e9e0a11c7.Hwmon1 /xyz/openbmc_project/sensors/fan_tach/fan2 xyz.openb
mc_project.Sensor.Threshold.Warning
NAME TYPE SIGNATURE RESULT/VALUE FLAGS
```

Тест-кейс заблокирован!

ПРОЙДЕНО:

- Вентиляторы обнаружены в D-Bus (fan0, fan1, fan2)
- Сервис Hwmon работает корректно
- Структура сенсоров создана правильно

НЕ ПРОЙДЕНО:

- Вентиляторы показывают ошибочное значение: -110 RPM
- Отсутствуют пороговые значения (Warning/Critical thresholds)
- Вентиляторы не отображаются в веб-интерфейсе (из-за ошибочных значений)

3. Удаленный доступ:
   - Подключение к серверу через IPMI.

BLOCKED

4. Логирование событий:
   - Проверка записи логов при включении сервера.

BLOCKED

Последние два теста так же нереализуемы ввиду той проблемы, что была указана в первом тесте:

- Аппаратная эмуляция OpenPOWER в QEMU не функционирует
- FSI (Fabric Serial Interface) не может просканировать процессоры
- SBE (Self Boot Engine) не может инициализироваться
- Хост находится в состоянии "Not Available"

ЧАСТЬ 4: Оформление тест-плана.

Тест-план OpenBMC

1. Введение

1.1. Назначение

Данный документ описывает план тестирования функциональности OpenBMC для проверки корректности работы основных систем управления сервером.

1.2. Область применения
Тестирование проводится для проверки основных функций базового контроллера управления (BMC) в рамках эмуляции OpenPOWER системы.

2. Цели тестирования
2.1. Основные цели
- Проверить корректность работы основных функций OpenBMC
- Убедиться, что система соответствует базовым требованиям управления сервером
- Выявить проблемы в текущей конфигурации эмуляции

2.2. Критерии успеха
- Критические функции управления питанием работают стабильно
- Система мониторинга обнаруживает и отслеживает аппаратные компоненты
- Логирование событий функционирует корректно

3. Объем тестирования
- Управление питанием
- Мониторинг аппаратного обеспечения (с ограничениями)
- IPMI (заблокировано)
- Логирование событий (заблокировано)

4. Подходы и методы тестирования
- Функциональное тестирование - проверка соответствия функций заявленным требованиям.
- Интеграционное тестирование - проверка взаимодействия компонентов системы
- Системное тестирование - проверка работы системы в целом

5. Ресурсы
5.1. Оборудование
- ПК с операционной системой Linux
- Образ OpenBMC для эмуляции Romulus (OpenPOWER)
- QEMU для виртуализации окружения

5.2. Программное обеспечение
- OpenBMC образ версии 3.0.0-dev
- IPMI-клиент (ipmitool)
- Терминальный клиент для доступа к BMC
- Веб-браузер для доступа к web-ui

5.3. Человеческие ресурсы
- Студент - 1 человек

- Время на выполнение: 1 час

6. График тестирования

Компонент	**Время выполнения**	**Статус**
Управление питанием	30.09.2025 23:00 - 23:15	ВЫПОЛНЕНО
Мониторинг аппаратного обеспечения	30.09.2025 23:20 - 23:35	ВЫПОЛНЕНО С ОГРАНИЧЕНИЯМИ
IPMI	30.09.2025 23:20 - 23:35	ЗАБЛОКИРОВАНО
Логирование	30.09.2025 23:40 - 23:55	ЗАБЛОКИРОВАНО

7. Критерии начала и завершения

7.1. Критерии начала тестирования

- Оборудование настроено и готово к работе
- Образ OpenBMC загружен и доступен
- Тест-кейсы подготовлены и проверены
- Доступны инструменты мониторинга и управления

7.2. Критерии завершения тестирования

- Все запланированные тест-кейсы выполнены
- Результаты тестирования документированы
- Обнаруженные проблемы зафиксированы
- Сформирован отчет о тестировании

Вывод

В ходе лабораторной работы были освоены навыки составления тест-плана для тестирования OpenBMC. Изучены основные функции OpenBMC, разработаны тест-кейсы и оформили тест-план, который может быть использован для проверки корректности работы системы.