[OpenBMC Development Guide]: Getting Started and In-depth Analysis of 10 Key Areas

Unlock now

Published: 2025-02-21 12:16:26 Views: 391 Subscribers: 34 VIP专栏

ZIP openbmc bmc web Download Now

OpenBMC webserver This component attempts to be a "do everything" embedded webserver for OpenBMC. Features The webserve Expand

Copy full text

Table of contents

SW_Sun Wei

Development technology expert

An engineer at a well-known technology company, he has extensive work experience and expertise in the field of technology development. He has been responsible for the design and development of multiple complex software systems involving large-scale data processing, distributed systems, and high-performance computing.

openbmc/ **openbmc**

OpenBMC Distribution

357 Contributors | 33 Issues | 2k Stars | 999 Forks

summary

This article first gives an overview of OpenBMC and introduces its basic concepts and basic knowledge. It then elaborates on the process of building the OpenBMC environment, including key steps such as software architecture introduction, environment preparation, startup and boot. In the programming basics chapter, this article explains how to use OpenBMC's command line interface, RESTful API, and Python script programming for basic development. In advanced practice, specific methods for system monitoring and management, software development and customization, and security enhancement are discussed. Finally, the cutting-edge application cases of OpenBMC in data center applications and automated operation and maintenance practices are analyzed, and its future development trends are prospected. This article aims to provide readers with a comprehensive OpenBMC learning and practice guide to help developers and system administrators deeply understand and use OpenBMC technology.

Keywords

OpenBMC; software architecture; environment construction; command line interface; RESTful API; system monitoring; security enhancement

Reference resource link: OpenBMC New Model Development Guide: From Machine Layer to Kernel Changes

1. OpenBMC Overview and Basic Knowledge

1.1 Introduction to OpenBMC

OpenBMC is an open source project that aims to provide a complete firmware stack for the Baseboard Management Controller (BMC). BMC is an integral part of server hardware, responsible for monitoring the physical status of the server, enabling remote management, and providing power control. The OpenBMC project allows developers to create customized BMC firmware for a variety of hardware platforms using the Linux operating system and the Yocto Project toolset.

1.2 Core Values of OpenBMC

The core value of the OpenBMC project lies in its open source nature, which provides great freedom and flexibility to hardware manufacturers and users. Users can customize the firmware according to their needs to better integrate and manage server hardware. In addition, the active participation of the community promotes rapid problem solving and feature development, ensuring the stability and scalability of the project.

1.3 Preparations before using OpenBMC

Before starting to use OpenBMC, developers need to have a clear understanding of the system requirements. First, you need to have a certain Linux knowledge base, including command line operations and basic network knowledge. Second, being familiar with the OpenBMC project documentation, wiki, and community forums is essential for solving problems and getting help. Prepare a suitable development environment, such as a Linux desktop operating system, and master the skills of how to obtain OpenBMC source code and build a development environment.

2. OpenBMC environment construction

2.1 Introduction to OpenBMC Software Architecture

2.1.1 Software Components Overview

OpenBMC is an open source project that aims to provide firmware support for servers based on ARM and x86 architectures. Its software architecture consists of multiple components, including system services, protocol implementations, hardware abstraction layers, and user interfaces. One of the core components of OpenBMC is phosphor-dbus, a service framework that uses D-Bus (a cross-platform system message bus) as a communication mechanism. All services run in low-privilege user space to ensure system security.

2.1.2 Functions and roles of each component

- **phosphor-dbus** : responsible for implementing the mapping between service interface and D-Bus, and handling requests from different services.
- **phosphor-rest-server** : Provides RESTful API services, allowing remote interaction with the BMC through HTTP/HTTPS protocols.
- **IPMI (Intelligent Platform Management Interface)** : is a hardware interface standard used for system health monitoring and management. It is integrated into phosphor-dbus and exposed to users through the REST interface.
- **Hardware Abstraction Layer (HAL)** : Provides a unified operating interface for hardware. It is responsible for implementing the logic of interacting with the hardware, such as sensor reading, power control, etc.
- **System Services** : including basic services such as system status monitoring, logging, network configuration, etc., which together constitute the operating basis of OpenBMC.

2.2 OpenBMC Environment Preparation

2.2.1 Necessary hardware and software requirements

Before building the OpenBMC environment, you need to prepare the following hardware and software resources:

- **Compatible development board** : Choose a development board that supports OpenBMC, such as BeagleBone Black or MinnowBoard Max.
- **Boot media** : An 8GB or larger USB drive to store the OpenBMC boot image.
- **Operating system** : Ubuntu is recommended as the development and testing environment.
- **Build tools** : Dependent tools required to compile OpenBMC, such as cmake, git, python3, and cross-compilation toolchain.

2.2.2 Installation steps and configuration methods

The steps to install OpenBMC are briefly described as follows:

1. Install the necessary development tools and dependency packages in the Ubuntu system.
2. Clone the OpenBMC code repository to a local working directory.
3. Configure the code base and select the target architecture appropriate for your development board.
4. Compile the OpenBMC image.
5. Burn the generated image to a USB drive or directly to the internal storage of the development board.
6. Insert the USB drive or boot the board to run OpenBMC.

```
# 安装依赖工具
sudo apt-get install cmake git python3 gcc-arm-linux-gnueabihf device-tree-compiler

# 克隆代码库
git clone https://github.com/openbmc/openbmc

# 进入代码目录并创建构建目录
cd openbmc
mkdir build
```

Next, you need to burn the image file you created to a USB drive or development board.

2.3 OpenBMC startup and boot

2.3.1 BIOS settings and adjustments

Before booting OpenBMC, you may need to do some basic setup in the BIOS to ensure system compatibility, such as setting the boot order to boot from USB or from onboard storage. Make sure boot security features in the BIOS (such as Secure Boot) are disabled to help avoid potential problems during the boot process.

2.3.2 Interpretation of key logs during startup

The OpenBMC boot process can be logged through the serial port, which is very helpful for debugging and analyzing boot problems. By analyzing the boot log, you can understand the startup sequence of system services, error messages, and the initialization of the system environment.

```
[    1.000000] Starting up OpenBMC...
[    1.100000] D-Bus service started, listening on 0.0.0.0:22
[    2.000000] Starting RESTful API server...
[    2.200000] phosphor-rest-server started on port 8080
[    3.100000] phosphor-dbus service ready
```

The above is a simplified boot log example that reflects the key components and initialization steps of OpenBMC.

In this chapter, we have discussed the software architecture of OpenBMC, environment preparation, key configuration and log interpretation during the boot process. The following chapters will introduce the programming basics of OpenBMC in depth, including the use of command line interface, application of RESTful API and Python script programming, which are important skills to familiarize yourself with and expand the functions of OpenBMC.

3. OpenBMC Programming Basics

The OpenBMC Programming Basics chapter mainly introduces how to interact and program with OpenBMC through the command line interface, RESTful API, and Python scripts. In this section, we will explore in depth how to use these interfaces, as well as how to write, test, and deploy scripts and programs to extend the functionality of OpenBMC.

3.1 OpenBMC Command Line Interface

OpenBMC provides a powerful command line interface (CLI) that allows users to perform various system management and configuration operations. CLI is the cornerstone for developers and administrators to understand and control BMC systems.

3.1.1 How to use common commands

In the OpenBMC command line, users can perform a variety of operations. Among them, `bmc`, `ipmitool` and `fw-util` are several commonly used command tools.

`bmc` The tool is used to manage the BMC system, including viewing the system status, restarting the BMC, and viewing and setting the BMC system log level.

```
# 查看BMC版本信息
bmc-info version

# 查看系统状态
bmc-info state

# 重启BMC服务
bmc-restart
```

`ipmitool` It is a command line tool for managing IPMI (Intelligent Platform Management Interface) based systems. It provides a series of commands for monitoring and managing the status of server hardware.

```
# 获取系统事件日志
ipmitool sel list

# 获取系统传感器信息
ipmitool sdr elist
```

`fw-util` Command is used to update BMC firmware.

```
# 列出可用的固件设备
fw-util list-devices

# 更新固件
fw-util update <固件文件路径>
```

3.1.2 Command Line Interaction Tips

Mastering some command line interaction skills can improve management efficiency, such as using history commands (`history`), command aliases (`alias`), and the auto-completion function (`tab`).

```
# 查看最近执行的命令
```

```
history

# 设置别名，简化命令输入
alias bmci='bmc-info'

# 使用 tab 自动补全命令
# 当输入 bmci 后按 tab 键，系统自动补全命令为 bmc-info
bmci<tab>
```

3.2 OpenBMC RESTful API Application

OpenBMC also provides a RESTful API based on the standard HTTP protocol, allowing developers to remotely access and manage BMC in a more flexible way.

3.2.1 RESTful API Concept and OpenBMC Implementation

RESTful API is a stateless, resource-centric approach that utilizes the HTTP protocol, where resources are identified by URIs and operations are performed using HTTP methods such as GET, POST, PUT, DELETE.

OpenBMC's RESTful API can interact with BMC through HTTP requests. Users can use curl or any tool that supports HTTP requests to communicate with the API.

```
# 获取当前登录用户信息
curl -u <用户名>:<密码> http://<OpenBMC IP地址>/api/xyz/openbmc_project/logging/actions/u

# 添加一个新的日志条目
curl -u <用户名>:<密码> -X POST -d '{"data": {"message": "New log entry"}}' http://<OpenBMC I
```

3.2.2 API interaction examples and common operations

API interaction examples and common operations cover how to use the RESTful API to manage users, system logs, power status, etc.

Take the management system log as an example:

```
# 列出所有日志条目
curl -u <用户名>:<密码> http://<OpenBMC IP地址>/api/xyz/openbmc_project/logging/entries

# 获取特定日志条目的详细信息
curl -u <用户名>:<密码> http://<OpenBMC IP地址>/api/xyz/openbmc_project/logging/entries/<
```

3.3 Python script programming for OpenBMC

Python has become the preferred language for OpenBMC scripting because of its readability and concise syntax.

3.3.1 Overview of Python Application in OpenBMC

OpenBMC provides a Python library, `redfish_client`, which enables developers to use Python scripts to interact with the RESTful API.

3.3.2 How to write and test Python scripts

The following is `redfish_client` an example of a Python script that uses the library to query the system status.

```
import redfish_client
from redfish_client.rest import ApiException

# 创建一个redfish对象
redfish = redfish_client.redfish_client(base_url='http://<OpenBMC IP地址>', username='<用户:

# 获取系统状态
try:
    response = redfish.get("/redfish/v1/Systems/1/")
    print(response)
except ApiException as e:
    print("Exception when calling RedfishServiceApi->get_service_root: %s\n" % e)
```

In order to test and run Python scripts, you usually need a configured Python environment and installed `redfish_client` libraries. When testing scripts, you can use Python's built-in libraries such as `unittest` to write unit tests.

```
import unittest
```

```
class TestOpenBMCPythonScript(unittest.TestCase):
    def test_system_status(self):
        # 此处应当调用编写好的获取系统状态的函数
        pass

if __name__ == '__main__':
    unittest.main()
```

Through the above examples and explanations, we can see that using Python scripting can greatly simplify the interaction with OpenBMC and provide richer programming logic and error handling capabilities. Mastering these basic knowledge will enable developers to achieve more automated and personalized management tasks on the OpenBMC platform.

Please note: This chapter content is simplified and abstracted to a certain extent under the premise of complying with the above requirements to ensure readability and clarity. The actual chapter content should be more detailed and in-depth, and contain more abundant operation examples and code comments.

Copy full text

4. OpenBMC Advanced Practice

As an open source BMC (Baseboard Management Controller) software solution, OpenBMC is widely used in many fields such as servers, network equipment and embedded systems. After becoming familiar with its basic knowledge and environment construction, we will explore the advanced practices of OpenBMC in depth, including system monitoring and management, software development and customization, and security enhancement.

4.1 OpenBMC system monitoring and management

OpenBMC provides a variety of tools and interfaces for monitoring system health and managing devices. System administrators can use these tools to detect system anomalies in a timely manner and take corresponding measures.

4.1.1 Use of system monitoring tools

OpenBMC integrates multiple system monitoring tools, such as IPMI, Redfish, and other dedicated monitoring services. These tools can provide real-time data on hardware indicators such as temperature, voltage, fan speed, etc.

- **IPMI (Intelligent Platform Management Interface)** : is a widely used hardware management interface that can be monitored through the command line or dedicated IPMI tools.
- **Redfish** : A management interface based on the RESTful API that provides a rich data model and standard interfaces to facilitate device management for developers and system administrators.
- **Dedicated monitoring services** : For example, OpenBMC's built-in Sensor Framework is used to collect and manage sensor data.

The use of these tools can help monitor hardware status and promptly alert you when problems occur.

Example code snippet:

```
# 通过ipmitool获取系统温度数据
ipmitool sdr elist | grep -i temperature
```

Logical analysis and parameter description: The above command will list all temperature sensor data obtained through the IPMI interface. `ipmitool` It is a commonly used command line tool for monitoring and managing IPMI-based systems. `sdr elist` The subcommand is used to display the sensor data record list and `grep -i temperature` filter out temperature-related data.

4.1.2 Performance Analysis and Troubleshooting

Analyzing system performance and diagnosing faults are important aspects of system management. OpenBMC supports performance monitoring and troubleshooting through log analysis and performance testing tools.

- **Log analysis** : OpenBMC's log files record key information about system operation. Problems can be quickly located by analyzing the logs.
- **Performance testing tools** : For example, use the iperf tool to test network performance, or use the stress tool to test system load capacity.

When diagnosing a fault, you should first check the log file to find abnormal information. Then, based on the possible error information, execute the corresponding performance testing tool to further locate the problem.

Example code snippet:

```
# 使用iperf测试网络性能
```

```
iperf -s
```

Logical analysis and parameter description: The above command starts the iperf server, which is used to monitor the network connection from the client to test the upload and download speed of the network. `-s` The parameter indicates that iperf runs in server mode.

4.2 OpenBMC Software Development and Customization

The flexibility and open source nature of OpenBMC allow developers to develop and customize software according to their own needs, which is of great significance for deployment on specific hardware platforms or in special demand scenarios.

4.2.1 Source code acquisition and compilation process

In order to develop and customize software, you first need to obtain the OpenBMC source code from the official repository and then compile it according to the target hardware platform.

- **Get the source code :** Clone the code from the OpenBMC GitHub repository (https://github.com/openbmc/openbmc) to your local computer.
- **Compilation process** : Depending on the platform, execute the corresponding compilation script and configuration options.

```
# 获取OpenBMC源码
git clone https://github.com/openbmc/openbmc.git
cd openbmc

# 配置和编译目标平台的OpenBMC
./scripts/configure <platform>
```

Logic Analysis and Parameter Description: `<platform>` It is a placeholder for the platform name, indicating that the user needs to select the appropriate compilation configuration according to their target hardware platform. `scripts/configure` The script will prepare the compilation environment and generate the appropriate Makefile file. Once the configuration is completed, use `make` the command to start the compilation process.

4.2.2 Software Customization, Modification and Release

After obtaining and compiling the source code, developers can customize the source code as needed and publish the modified software package.

- **Customized modification** : modify the configuration files in the software package, add or modify functional modules, etc.
- **Release software packages** : Use package management tools, such as RPM or DEB, to package the customized software.

Example code snippet:

```
# 创建RPM包
spectool -g <spec_file>
rpmbuild -ba <spec_file>
```

Logical analysis and parameter description: `<spec_file>` It is a placeholder for a spec file, which is a file that describes how to build an RPM package. First, `spectool` the command is used to extract the source code and patches from the spec file, and then `rpmbuild` the command uses this information to build the RPM package. `-ba` The parameters instruct rpmbuild to build the software package completely from beginning to end.

4.3 OpenBMC security enhancements

In an increasingly complex network environment, system security becomes increasingly important. OpenBMC provides a variety of security mechanisms, such as access control, security updates, and vulnerability protection.

4.3.1 Configuration and management of security mechanisms

OpenBMC allows administrators to configure different security mechanisms to protect the system.

- **Access control** : Limit access to the BMC by configuring user permissions.
- **Secure Updates** : Ensure the origin and integrity of software packages by setting up signature checks and a secure software update process.
- **Vulnerability protection** : Regularly check and update the security vulnerability database to quickly respond to security issues.

For example, you can restrict access to specific IP addresses by editing the OpenBMC configuration file.

Example code snippet:

```
# 编辑配置文件，限制访问
vim /etc/bmcweb.conf
```

Logical analysis and parameter description: After editing `/etc/bmcweb.conf` the file, you can set it `TrustedHosts` to define the IP address range that is allowed to access. This is a simple access control method that helps prevent unauthorized remote access.

4.3.2 Preventive measures for common security vulnerabilities

Administrators need to take effective preventive measures against common security vulnerabilities that OpenBMC may face.

- **Update and patch** : Regularly update OpenBMC to the latest version and apply security patches in a timely manner.
- **Use strong passwords** : Set complex passwords for the OpenBMC management interface.
- **Network isolation** : Ensure that the BMC network is isolated from the business network to reduce risks.

For example, use the OpenSSL tool to generate strong passwords.

Example code snippet:

```
# 使用openssl生成强密码
openssl rand -base64 16
```

Logical analysis and parameter description: The above command uses OpenSSL `rand` command to generate a 16-byte random string based on base64 encoding, which can be used as the password of the OpenBMC management interface. The password generated by this method is complex and difficult to guess, thus increasing security.

In summary, the advanced practice section focuses on the advanced functions and operations of OpenBMC, such as system monitoring and management, software development and customization, and security enhancement. This knowledge requires not only a solid foundation for administrators, but also continuous learning and practice in order to better use OpenBMC in daily work. In this section, we provide the use of monitoring tools, cases of performance analysis and fault diagnosis, source code acquisition and compilation process, methods of software customization, modification and release, and security enhancement measures. These knowledge points are all important components of OpenBMC advanced practice.

5. OpenBMC's cutting-edge applications and case studies

5.1 Application of OpenBMC in Data Center

With the development of cloud computing and big data technology, the demand for server management in data centers is growing. Data centers need a reliable, efficient and flexible Baseboard Management Controller (BMC) to manage and maintain server infrastructure. OpenBMC has been widely used in the data center field due to its openness, modularity and community support.

5.1.1 Analysis of Data Center's Requirements for BMC

The BMC requirements of the data center can be analyzed from the following dimensions:

1. **Scalability** : The size of a data center may change at any time, so the BMC needs to be able to expand flexibly to support server deployments of different sizes.
2. **Remote management** : Data centers usually have a large number of servers, so remote management is a basic requirement. This includes remote power on, power off, and reset of servers, as well as remote diagnosis and repair of problems.
3. **High availability** : Data centers cannot tolerate long periods of downtime, so BMC must provide high availability features to ensure service continuity.
4. **Security** : The BMC in the data center requires strict security control to prevent unauthorized access and potential security threats.
5. **Integration and compatibility** : The BMC needs to be able to integrate seamlessly with existing monitoring, management tools and other systems in the data center.

5.1.2 OpenBMC deployment case in data center

The following is an example of OpenBMC deployment in a data center:

1. **Deployment background** : A cloud computing company plans to expand its data center and needs to deploy thousands of servers.
2. **OpenBMC Deployment** : OpenBMC was chosen as its server management solution to meet the needs of scalability, remote management, and high availability.
3. **Customization** : Due to interface differences between existing management tools and OpenBMC, the enterprise conducted customized development to achieve seamless integration.

4. **Security Enhancement** : Enhance security by implementing OpenBMC's security mechanisms, such as using TLS/SSL encryption, setting access control lists (ACLs), etc.
5. **Monitoring and Automation** : Integrate monitoring tools to track server health status and automate server deployment and maintenance through scripting.

The deployment results show that OpenBMC not only meets the needs of the data center, but also achieves good integration with the existing system through its flexible interface and customizable features, and improves the overall operation and maintenance efficiency of the data center.

5.2 OpenBMC automated operation and maintenance practice

Automated operation and maintenance is an important part of data center maintenance. It can greatly improve operation and maintenance efficiency, reduce human errors, and ensure stable operation of the system. As the infrastructure for server management, OpenBMC provides good support for automated operation and maintenance.

5.2.1 Automated operation and maintenance tools and processes

OpenBMC provides RESTful API and command line interface, which are the basis for the implementation of automated operation and maintenance tools. The operation and maintenance team can use these interfaces to write scripts to complete the following tasks:

- **Automated deployment** : Use OpenBMC's API to automate the deployment of new server hardware.
- **Status monitoring** : Regularly check the health status of the server and warn of potential problems.
- **Fault handling** : Automatically restart services or perform remote diagnosis when a fault is detected.
- **Configuration Management** : Automatically update and manage server configurations to maintain consistency and compliance.

5.2.2 OpenBMC's role in automated operations

In the automated operation and maintenance process, the role of OpenBMC can be viewed from the following aspects:

1. **As a central control point** : OpenBMC acts as the central control point for server hardware. Through its API, it can control server startup, shutdown, restart and other operations.
2. **Data provider** : OpenBMC provides detailed system status data, which is essential for operation and maintenance monitoring and troubleshooting.
3. **Terminal for executing commands** : Through the OpenBMC command line interface, operation and maintenance personnel can send instructions to manage server hardware.

With OpenBMC, the automated operation and maintenance team can achieve integrated management and control from the server hardware layer to the software layer.

5.3 Future Development Prospects of OpenBMC

Since its birth, OpenBMC has been continuously improved and developed with the support of the community and the use of many corporate users, showing strong vitality and innovation. With the advancement of technology and changes in market demand, OpenBMC is also facing new challenges and development opportunities.

5.3.1 Current Development Trends and Challenges

The current development trends of OpenBMC include:

1. **Functional enhancement** : New features are continuously added, such as compatibility with more hardware, support for new management protocols, etc.
2. **User experience improvement** : Optimize the management interface and user interaction to improve usability and efficiency.
3. **Performance improvement** : Optimize API performance, reduce latency, and increase processing speed.
4. **Security enhancement** : As network attacks become increasingly sophisticated, the security of OpenBMC also needs to be continuously improved.

The challenges are:

1. **Hardware compatibility** : Needs to be continually updated to support new hardware.
2. **Large-scale deployment** : How to efficiently manage large-scale server clusters has become a major challenge.
3. **Security risks** : Need to guard against higher level security threats.

5.3.2 Predictions and recommendations

In response to the above challenges and trends, here are some suggestions:

1. **Strengthen community cooperation** : Promote the continuous improvement of OpenBMC functions and performance through community cooperation.
2. **Strengthen security** : Conduct security audits regularly and introduce new security features, such as hardware-level security protection.

3. **Simplify the deployment process** : Provide more automated deployment options to lower the user threshold.
4. **Increased investment** : Support and development of new hardware requires increased investment from industries and businesses.

As a representative of open source BMC, OpenBMC has demonstrated its great potential in the fields of data centers and automated operation and maintenance. With the further development of technology, OpenBMC is expected to play a role in a

Copy full text

Related Recommendations

Open BMC WEBUI development documentation
Open BMC WEBUI development documentation

Open BMC User Management Development Documentation
Open BMC User Management Development Documentation

Open BMC WebUI Refactoring: Vue.js Modern Development and Accessible Design
" Open BMC WEBUI Development Document" Open BMC (Open Baseband Management Controller) is an open source project dedi...

Open BMC Infrastructure Documentation: Organization, Application, and Automated Testing
This document provides an in-depth understanding of the Open BMC architecture and development process . First, the Open BMC ...

Open BMC BIOS Configuration Guide: UEFI/BIOS Configuration Details
The Open BMC BIOS related documents mention several important concepts and technologies, including UEFI, BIOS configuration, ...

[Open BMC and SPDM beginners ' guide]: From environment construction to performance optimization, mas...
[[Open BMC and SPDM beginners' guide]: From environment construction to performance optimization, master all key steps in one...

Open BMC test case sharing: a platform for industry best practices and experience exchange
This article first reviews the basic concepts of Open BMC , and then details how to build a development and testing environment, inc...

[Open BMC Practical Tips]: Master the skills of adding Sensor information in IPMI within 10 minutes
[【Open BMC Practical Tips】: Master the skills of adding Sensor information in IPMI in 10 minutes](https:// open graph.githubassets....

Facebook Open BMC vs. Traditional BMC : Advantage Comparison Analysis and Reliability Improvement Stra...
![Facebook Open BMC vs Traditional BMC : Advantage Comparison Analysis and Reliability Improvement Strategy]...Through the co...