|  |
| --- |
| 浪潮电子信息产业股份有限公司 |
| **OpenBMC入门指导手册** |
| V1.0 |
|  |
| **技术研发部** |
| **2020/11/5** |

|  |
| --- |
|  |

# 版本变更历史

|  |  |  |  |
| --- | --- | --- | --- |
| **时间** | **版本** | 描述说明 | 作者 |
| 2020-11-5 | **1.0** | 初始版本 | 刘锡伟 |
|  |  |  |  |
|  |  |  |  |

目录

[1、概述](#_Toc247238206_WPSOffice_Level1) [5](#_Toc247238206_WPSOffice_Level1)

[2、技术要求](#_Toc270337027_WPSOffice_Level1) [5](#_Toc270337027_WPSOffice_Level1)

[2.1了解：](#_Toc270337027_WPSOffice_Level2) [5](#_Toc270337027_WPSOffice_Level2)

[2.2熟悉：](#_Toc425849180_WPSOffice_Level2) [5](#_Toc425849180_WPSOffice_Level2)

[3、硬件配置](#_Toc425849180_WPSOffice_Level1) [6](#_Toc425849180_WPSOffice_Level1)

[3.1硬盘空间：](#_Toc1686406157_WPSOffice_Level2) [6](#_Toc1686406157_WPSOffice_Level2)

[3.2内存大小：](#_Toc1690573603_WPSOffice_Level2) [6](#_Toc1690573603_WPSOffice_Level2)

[4、软件配置](#_Toc1686406157_WPSOffice_Level1) [6](#_Toc1686406157_WPSOffice_Level1)

[4.1系统选择：](#_Toc1377232998_WPSOffice_Level2) [6](#_Toc1377232998_WPSOffice_Level2)

[4.2软件安装：](#_Toc888271870_WPSOffice_Level2) [6](#_Toc888271870_WPSOffice_Level2)

[4.3更新代码：](#_Toc1333639132_WPSOffice_Level2) [7](#_Toc1333639132_WPSOffice_Level2)

[5、代码编译](#_Toc1690573603_WPSOffice_Level1) [7](#_Toc1690573603_WPSOffice_Level1)

[5.1导入环境变量](#_Toc110468003_WPSOffice_Level2) [7](#_Toc110468003_WPSOffice_Level2)

[5.2编译](#_Toc305652990_WPSOffice_Level2) [8](#_Toc305652990_WPSOffice_Level2)

[6、调试](#_Toc1377232998_WPSOffice_Level1) [9](#_Toc1377232998_WPSOffice_Level1)

[6.1开发包下载失败：](#_Toc434140690_WPSOffice_Level2) [9](#_Toc434140690_WPSOffice_Level2)

[6.2缓存问题：](#_Toc1248032452_WPSOffice_Level2) [9](#_Toc1248032452_WPSOffice_Level2)

[6.3目录问题](#_Toc830569139_WPSOffice_Level2) [10](#_Toc830569139_WPSOffice_Level2)

[6.4远程地址问题](#_Toc1515015360_WPSOffice_Level2) [10](#_Toc1515015360_WPSOffice_Level2)

[7、获取源码](#_Toc888271870_WPSOffice_Level1) [11](#_Toc888271870_WPSOffice_Level1)

[7.1 Devtool工具](#_Toc766161875_WPSOffice_Level2) [11](#_Toc766161875_WPSOffice_Level2)

[7.2 Git工具](#_Toc1194553926_WPSOffice_Level2) [12](#_Toc1194553926_WPSOffice_Level2)

[a. git pull origin master --rebase](#_Toc270337027_WPSOffice_Level3) [12](#_Toc270337027_WPSOffice_Level3)

[b. git fetch origin; git rebase origin/master](#_Toc425849180_WPSOffice_Level3) [12](#_Toc425849180_WPSOffice_Level3)

[8、参考](#_Toc1333639132_WPSOffice_Level1) [14](#_Toc1333639132_WPSOffice_Level1)

[8.1邮件列表](#_Toc469473758_WPSOffice_Level2) [14](#_Toc469473758_WPSOffice_Level2)

[8.2设计文档](#_Toc675307191_WPSOffice_Level2) [14](#_Toc675307191_WPSOffice_Level2)

[8.3学习视频](#_Toc132347245_WPSOffice_Level2) [14](#_Toc132347245_WPSOffice_Level2)

[8.4 gerrit地址](#_Toc1247302529_WPSOffice_Level2) [14](#_Toc1247302529_WPSOffice_Level2)

**1、概述**

OpenBMC: 与BMC功能相同、架构不同， 是依赖于Upstream（例如： OpenEmbedded, Yocto, systemd, linux-aspeed等）的一种开源项目。

参考：

<https://github.com/openbmc/openbmc/blob/master/README.md#openbmc>

**2、技术要求**

在学习和开发OpenBMC项目前，建议大家先了解一下以下技术：

**2.1了解：**

Yocto，bibake基本用法等

**2.2熟悉：**

C++11、14、17以及20的基本语法

git

Python的基本语法

dBus

Yaml格式

Dts(device tree)

Systemd

**3、硬件配置**

目前使用主流的系统是Ubuntu和CentOs，Inspur多数使用Ubuntu。

**3.1硬盘空间：**

由于OpenBMC是一种开源项目，并且一直在持续开发中，而且Upstream和OpenBMC的开发包都持续更新中，所以建议大家硬盘空间越大越好， 理论上是需要大于100G，实际最好是500G以上。

**3.2内存大小：**

建议使用台式机。如果使用笔记本， 内存至少需要8G内存。因为编译OpenBMC时候会占用很大内存，内存不足很有可能会出现卡死状态。

**4、软件配置**

**4.1系统选择：**

在OpenBMC社区中， 目前已经支持Ubuntu20.04，建议大家将自己的系统升级到20.04。

**4.2软件安装：**

更新代码之前，需要配置好所需的软件，一般情况安装以下软件即可：

git build-essential libsdl1.2-dev texinfo gawk chrpath diffstat

Python3.7.1 docker等

参考：

<https://github.com/openbmc/openbmc/blob/master/README.md#1-prerequisite>

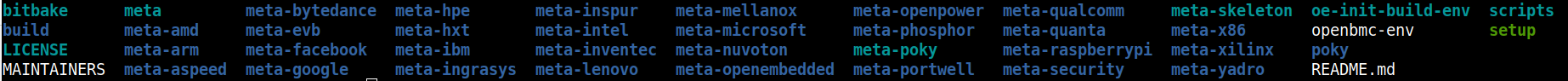
**4.3更新代码：**

可以使用git更新代码，目前我们代码存放在gitlab上，我们可以通过gitlab直接获取

Git clone <https://10.151.12.63/IPS/eBMC/openbmc.git>

注： 如果没有gitlab帐号，找相关人员注册。

如果成功，会生成一个openbmc的文件夹，里面的文件目录如下：



到此，我们可以认为OpenBMC的代码已经更新完成，接下来开始编译，测试等。

**5、代码编译**

由于我们基于NF5280M6做开发， 所以我们的整个手册都以NF5280M6为例。

**5.1导入环境变量**

我们编译NF5280M6，需要导入NF5280M6的开发配置模板

早期方式：

Export TEMPLATECONF=meta-inspur/meta-nf5280m6/conf

Source openbmc-env

更新后的方式：

Source setup nf5280m6

目前建议使用第二种方式

成功后，会直接进入了build目录下面，第二种方式会直接进入到build/nf5280m6/目录下。

**5.2编译**

OpenBMC项目是基于Yocto开发，并且所有的编译都是在docker中进行。

Bitbake：作为OpenEmbedded构建系统中的构建引擎，它是任务执行器和调度器。类似于我们熟悉的Make构建引擎，所有的代码编译都使用它。

Obmc-phosphor-image: 没有特殊含义， 在OpenBMC中生成镜像的一种名称

编译： bitbake obmc-phosphor-image（这是编译整个镜像，所有的包）

注意： 第一次编译时间会很长（很有可能是一天以上）， 因为需要下载大量的包，具体时间根据网速决定。

如果编译成功后， 会在以下目录中生成image-bmc的软链接镜像。

build/tmp/deploy/images/nf5280m6/

**6、调试**

当我们第一次编译或者以后编译镜像的过程中，经常会出现各种问题导致编译失败，这里主要介绍一下经常出现的几种问题以及如何解决

注意：如果我们在编译镜像前没有改动过代码， 出现的任何编译问题都不是代码问题， 所以需要考虑从其他角度去解决。

**6.1开发包下载失败：**

当编译过程中报错，并且报错类型现实下载失败，这时候大部分是我们网速的问题，或者也有可能是需要代理下载（因为OpenBMC有些部分包是需要配置VPN代理才可以下载， 例如本地CI测试等。但是编译obmc-phosphor-image时，应该不会出现使用代理下载的情况，大部分原因是网络断开或者网速问题导致）

**6.2缓存问题：**

如果我们编译了多次镜像，之后又拉取过最新的代码，这时很有可能会出现某个开发包编译失败，这时候如果不是下载失败， 很有可能是因为缓存问题导致，因为当我们编译过一次后会把编译的一些信息存入缓存目录中，这样下次会优先使用缓存目录，这时我们需要清理缓存并单独重新编译。

使用如下命令：

Bitbake xxx -c clean

Bitbake xxx

这样基本上就会成功。

**6.3目录问题**

如果我们使用上述两种方式都没有解决，可以查看报错信息是否是目录错误导致编译失败，需要查看以下目录的内容，目录结构是否正确：

openbmc/build/conf/bblayers.conf

如果有错误，建议大家删除build/tmp文件夹，重新编译

**6.4远程地址问题**

当我们在编译的过程中，也有可能会出现某个包一直编译失败，而且这个包是我们从未见过的，这时我们可以怀疑是否是github的远程地址被改动了（至于为什么改动，没有找到原因），这时我们可以手动更改远程地址，命令如下：

查看远程地址： git remote -v

更改远程地址： git remote set-url origin <地址rul>

然后再次确定地址正确后，最好在重新拉取一下代码，如果代码结构有变化，只能删除目录重新git clone。

以上就是经常遇见的问题，如果有其他问题，希望大家更新。

**7、获取源码**

由于OpenBMC在开源社区中，基本上都是以配置文件的形式存在，没有源码。编译的过程中通过\*.bb以及\*.bbappend获取相应recipes的URL和commitID的方式拉取源码并编译，如果我们要进行源码调式，更改代码等操作，需要拉取源码并编译：

**7.1 Devtool工具**

Devtool工具只有在导入环境变量之后才会存在（5.1章节）

获取代码： 以dbus-sensors为例

如果不确定，可以先查看这个repo是否存在

Bitbake -s | grep dbus-sensors

如果存在， 拉取代码

Devtool modify dbus-sensors

如果没有workspace，我们需要创建

Devtool create-workspace(会生成build/workspace目录)

代码拉取成功后， 在build/workspace/source/dbus-sensors存在

注意： 如果我们拉取代码后，再次编译的时候，会使用我们下载的这份代码进行编译。

删除代码：

Devtool reset dbus-sensors

记住一定要移除文件夹rm -rf build/workspace/source/dbus-sensors

以上是最常用的功能，如果有兴趣研究其他功能， 可以-h查看其他命令。

**7.2 Git工具**

学会使用git最基本的命令开发，拉取，上传代码

拉取代码：

我们的OpenBMC代码都存在gitlab上，第一次拉取代码时，使用git clone获取， 之后我们需要使用pull/fetch获取代码

注意：当我们每次需要上传代码时候，应该习惯性的先拉取最新的代码

有两种方式，假设我们的远程名称是origin 分支是master

1. git pull origin master --rebase

b. git fetch origin; git rebase origin/master

建议使用b

更新代码：

注意：这里只是介绍更新代码的操作流程，在开源社区中如果修改代码需要从gerrit里拉取代码并推送，而gitlab中的代码尽量使用patch的形式保存，因为我们要与社区同步，并且我们的代码不开源。

改完代码需要上传时候，commit信息需要包含以下三部分：

Header：基本上是描述这个功能的做什么的，是增加，修改，删除还是更新等。

<空行>

Body：具体描述此功能的意义。

<空行>

Tested：如果有测试，需要描述一下测试结果，或者有测试结果展示等。

注意：commit信息每行不得大于72（或者100）个字节。

Gitlab更新代码：

Git add \*

Git commit -s

Git push <origin> <HEAD:master>

注意：每次更改代码时候，多个commits是否有依赖？是否需要基于master分支提交？

**8、参考**

**8.1邮件列表**

<https://lists.ozlabs.org/listinfo/openbmc>

**8.2设计文档**

<https://github.com/openbmc/docs>

**8.3学习视频**

<https://github.com/openbmc/openbmc/wiki/Presentations>

**8.4 gerrit地址**

<https://gerrit.openbmc-project.xyz/>