测试驱动调试笔记

1. 主控板bootware切换uefi方法：
   1. 先升级btw文件，重启之后，进入Bootware菜单。
   2. 按Ctrl+A 进入Bootware命令行
   3. tftp 20000000 HARRISONVILLE.fd
   4. spiflash write 20000000 fd800000 800000



1. uefi切换回bootware的方法：（操作失败）
   1. 把c3000.btw放到启动盘的/efi/boot目录下
   2. 启动时按Ctl+d进入菜单
   3. 选择更新bootroot选项
   4. 选择从文件系统更新（Update from file system）
2. Bootware启动linux elf文件的方法：（操作失败）
   1. 按上述流程一更新bootware
   2. 按ctrl+a进入菜单
   3. tftp 40000000 xxx.elf下载elf文件
   4. bootlnx 40000000启动linux
3. uefi使用方法：
   1. map -r 显示挂接设备信息
   2. fs0: 进入u盘
   3. cd efi\boot 进入安装目录
   4. grubx64.efi 执行安装命令
4. grub下修改显示重定向到串口
   1. 按e进入编辑命令行
   2. 在linux行增加console=ttyS0,115200
   3. 按ctl+x进入系统
5. 制作启动U盘的方法
   1. 下载rufus程序，并启动
   2. 插入u盘，选择iso镜像
   3. 选择类型为GPT
   4. 开始执行即可



1. H3linux启动后串口不可用问题解决方法：
   1. 启动后按ESC进入BIOS
   2. Setup utility—>advance -> sourth bridge->uart设置
   3. 把HSUART0 设置为 Legacy，其他设置为hide
2. 接口板PTS版本编译环境构建方法
   1. 安装l3linx版本为 [h3Linux\_kernel\_4\_14.iso](http://10.153.3.130/h3linux/old/01.ISO/beta.01/h3Linux_kernel_4_14.iso)，路径为<http://10.153.3.130/h3linux/old/01.ISO/beta.01/>
   2. 编辑/etc/yum.repos.d/CentOS-Base.repo,注掉所有的mirrorlist=行，放开所有的baseurl=行
   3. 编辑/etc/hosts文件，增加 10.153.3.130 mirror.centos.org
   4. 安装gcc，yum install gcc
   5. 把代码包放入系统中执行make即可
3. H3linux内核编译和驱动安装说明：
   1. rpm -ihv kernel-alt-4.14.0-49.21.src.rpm安装源码，安装后会在/root目录下发现rpmbuild目录，源码存放在、/root/rpmbuild/BUILD/kernel-alt-4.14.0-49.el7a/linux-4.14.0-49.21.x86\_64/路径下
   2. vim SPECS/kernel-alt.spec修改版本号%define buildid .el7.centos
   3. yum install rpm-build安装编译工具
   4. yum -y install m4 xmlto asciidoc hmaccalc python-devel newt-devel git pesign elfutils-devel binutils-devel bison audit-libs-devel java-devel numactl-devel pciutils-devel安装依赖
   5. yum provides "perl(ExtUtils::Embed)"查看perl版本，yum -y install perl-ExtUtils-Embed-1.30-297.el7.noarch安装perl
   6. rpmbuild -ba SPECS/kernel-alt.spec --without debug --without debuginfo --without python --without perf --without tools开始编译内核
4. IPC芯片驱动安装和初始化
   1. 内核安装rpm -ivh \* --force
   2. 安装内核驱动 insmod bsp\_spi.ko | true
   3. 用户态初始化 ./bcm5396\_init
   4. 查看端口状态 ./count\_print 回车，然后输入端口号
   5. 设置面板口ip ifconfig enp3s0f0 192.168.1.105
   6. Ping本地主机
   7. 设置内部口ip ifconfig enp1sf0 192.184.7.6
   8. Ping 测试仪内部接口

1. 内核rpm包制作
   1. 1.下载kernel源码：

wget https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.18.18.tar.gz

* 1. 解压源码

tar -xvf linux-4.18.18.tar.gz

* 1. 配置kernel config
  2. make rpm（4.19以后的版本用make rpm-pkg ##同时构建源和二进制RPM软件包或者make binrpm-pkg ##仅构建二进制RPM软件包）
  3. 提取kernel.spec
  4. 创建rpmbuild目录

mkdir rpmbuild/{BUILD,SOURCES,SPECS,RPMS,BUILDROOT} -pv

* 1. 拷贝文件

cp linux-4.18.18.tar.gz rpmbuild/SOURCES

cp kernel.spec rpmbuild/SPECS

* 1. 开始rpmbuild

rpmbuild -bb --with baseonly --without debug --without debuginfo --target=x86\_64rpmbuild/SPECS/kernel.spec

rpmbuild 参数说明:

(1) --with <option> 指明编译某一种 kernel，可选的 option 有 baseonly、smponly 和 xenonly 等。

(2) --without <option> 指明去除一些功能模块，可选的 option 有 xen、smp、up、pae、kdump、debug 和 debuginfo 等。其中，debug 选项会剔除内核中的调试代码，debuginfo 选项会禁止建立 kernel-debuginfo 包。

1. 5396部分寄存器：
   1. 0x0/0x60+portId 设置为0x8b ------千兆、全双工、自协商
   2. 0x31/portId\*4 设置端口掩码保证端口互通/端口隔离
   3. 0x10+portId/0x20 设置为0x01f0 ------SGMII PHY模式
   4. 0x10+portId/0x0 设置为 |0x40 ------MII自协商
   5. 0x10+portId/0x0 设置为 |0x8000 ------重置
   6. 0x80+portId/0x0 设置为 0x9140------phy重置
   7. 0x80+portId/x Phy相关寄存器设置
   8. Phy芯片解复位 dffe0009 => ff dffe000a =>ff
2. Linux下lvm用U盘扩容根系统的简单方法（注意：扩充后会巨慢无比，不建议）
   1. 安装系统的时候选择lvm方式启动
   2. 将U盘插入查看盘符，这里为/dev/sda
   3. 用parted /dev/sda给U盘重新分区输入mklabel msdos设置盘符，然后yes退出
   4. vgextend openeuler /dev/sda 给现有的vg扩容
   5. lvextend /dev/openeuler/root /dev/sda 给lv扩容到根目录
   6. resize2fs /dev/openeuler/root 把扩充的存储添加到硬盘空间
   7. 此时用df -h可以看到根目录空间已经扩容
3. Linux下修改网卡Mac地址的方法：
   1. #ifconfig eth0 down
   2. #cd /etc/sysconfig/network-scripts
   3. #vi ifcfg-eth0
   4. 修改其中的"HWADDR=xx:xx:xx:xx:xx:xx"为"MACADDR=xx:xx:xx:xx:xx:xx"
   5. #ifconfig eth0 up
   6. #service network start
4. 修改网卡名字的方法：
   1. system("ip link set dev ens1f0 down");
   2. system("ifconfig ens1f0 down");
   3. system("ip link set ens1f0 name 3");
   4. system("ip link set dev 3 up");
   5. system("ifconfig 3 up");
5. Openeuler docker镜像使用
   1. Docker load -i openEuler-docker.aarch64.tar.xz
   2. docker run -d --name taishang -it -d --privileged=true -p 10022:22 -p 10443:443 -v /vm/taishan:/etc/yum.repos.d:rw openeuler-20.03-lts /bin/bash
   3. docker start taishang
   4. docker exec -it taishang /bin/bash
6. 文件系统支持中文的两种方法
   1. 新做语言包

localedef -i en\_US -f UTF-8 en\_US.UTF-8

localedef -i en\_US -f UTF-8 en\_US.UTF-8

localedef -i zh\_CN -f UTF-8 zh\_CN.UTF-8

localedef -i zh\_CN -f GB2312 zh\_CN

localedef -i zh\_CN -f GB2312 zh\_CN.GB2312

localedef -i zh\_CN -f GBK zh\_CN.GBK

* 1. 删除不需要的语言包

localedef --list-archive | egrep -v ^"en\_US|zh" | xargs localedef --delete-from-archive

mv -f locale-archive locale-archive.tmpl

build-locale-archive

1. 支持中文的方法
   1. 下载安装包

glibc-common-2.28-79.oe1.x86\_64.rpm

下载方法 在centos 8上调用dnf download glibc-common

kde-filesystem-4-47.el7.x86\_64.rpm

kde-l10n-4.10.5-2.el7.noarch.rpm

kde-l10n-Chinese-4.10.5-2.el7.noarch.rpm

下载方法 在centos 7上调用yum install kde-l10n-Chinese.noarch --downloadonly --downloaddir=.（centos不支持）

* 1. rpm包解压

rpm2cpio rpmfile.rpm | cpio -idmv

把解压的包拷贝到文件系统目录

1. linux无密码登录
   1. 运行systemctl edit serial-getty@ttyS0.service ，并添加以下内容:

[Service]

ExecStart=

ExecStart=-/sbin/agetty -o '-p -- \\u' --keep-baud 115200,38400,9600 --noclear --autologin root ttyS0 $TERM

* 1. 这将导致 agetty自动登录root用户，但仅通过此更改，系统仍会提示您输入 root 密码。我们可以配置/etc/pam.d/login验证 root无需密码即可登录控制台。将以下内容添加到 /etc/pam.d/login 的顶部:

auth sufficient pam\_listfile.so item=tty sense=allow file=/etc/securetty onerr=fail apply=root

1. 升级openeuler 2203的几个关键点：
   1. 把docker文件系统中的/usr/lib64覆盖拷贝到新系统中
   2. 很多工具都依赖一个基础库libsystemd-shared-249.so，需要拷贝到/usr/lib64目录下
   3. 需要删除/usr/lib/目录下的libseccomp.so.2.3.1，并去掉libseccomp.so.2的软连接
   4. 需要把新系统下的libseccomp.so.2拷贝过来
   5. 需要拷贝libselinux.so.1和libkmod.so.2到新的系统下
   6. 需要拷贝/usr/bin/journalctl到新的系统下
   7. 需要拷贝/usr/lib/systemd/systemd-logind到新的系统下
   8. ####Openssh重新下载安装，最新版本的ssh可能会导致系统启动失败
   9. 拷贝scp到新系统下
   10. 拷贝/usr/bin/ssh\* 到新系统下
   11. cp /usr/lib/systemd/systemd initrd//usr/lib/systemd/system
   12. cd initrd//usr/sbin
   13. ln -s ../lib/systemd/systemd init
   14. 支持gdb需要完整安装guile
2. 启动sshd错误说明：
   1. 修改文件权限为只读chmod 0600 /etc/ssh/ssh\_host\_\*
   2. 添加目录 mkdir -p /var/empty/sshd
3. 设置开机自动启动：
   1. 服务如果需要自动启动，需要在/etc/systemd/system/multi-user.target.wants/\*\*\*.service添加链接文件到/usr/lib/systemd/system/\*\*\*.service
   2. 如果去除自动启动，移除此文件即可。
4. rpcgen的用法
   1. yum install rpcgen
   2. yum install libtirpc-devel
   3. 构建test.x函数

program TESTPROG {

version VERSION {

string TEST(string) = 1;

} = 1;

} = 87654321;

* 1. rpcgen test.x
  2. rpcgen -Sc -o test\_client.c test.x
  3. rpcgen -Ss -o test\_srv\_func.c test.x
  4. gcc -Wall -ltirpc -I/usr/include/tirpc/ -o test\_client test\_client.c test\_clnt.c
  5. gcc -Wall -ltirpc -I/usr/include/tirpc/ -o test\_server test\_clnt.c test\_srv\_func.c test\_svc.c
  6. rpcgen -Sm test.x > Makefile

1. 科技部板卡的5396端口连接说明：
   1. 修改前：
      1. 1和2口通往IPC通道，在5396内部分别和3口相连
      2. 3口通往内部CPU的enp8s0，在5396内部和1和2口以及7口同时互联
      3. 7口通往面板网管口，在5396内部和3口互联
      4. 4口和5口分别通往CPUB和CPUC，在5396内部分别和6口相连
      5. 6口通往CPUA的enp9s0，在5396内部和4口和5口同时互联
   2. 修改后：
      1. 把1口和2口的连接从enp8s0，切换到4和5口，这样主控板同过IPC通道和CPUB和CPUC直接打通
      2. 4口和5口保留原来和CPUA的连接，同时增加到1口和2口的连接，这样CPUA、CPUB和CPUC保持互通，同时CPUB和CPUC打开到主控板的通道，但是CPUA和主控板隔离开。
   3. BCM5396\_Isolate\_Set函数说明：
      1. SSP\_RegWrite函数的第三个参数代表源端口，第五个参数代表需要打通的远端端口的bit信息，每个bit表示一个端口。
2. 修改SOCKET操作中Select连接超过1024个的时候出现进程挂死的修改说明：
   1. 在内核代码中搜索\_\_FD\_SETSIZE的值，发现默认是1024，修改成一个较大的数值，然后重新编译内核即可。
3. kdb重启支持：在cmdline.txt中添加
   1. kdb.cmd\_enable=1
4. intel openbmc编译方法：
   1. export TEMPLATECONF=meta-openbmc-mods/meta-wht/conf
   2. source oe-init-build-env
   3. bitbake intel-platforms
5. ipmitool使用命令
   1. ipmitool -I lanplus -H 10.112.134.112 -U test -P 20050919!a sol activate
6. python作为客户端操作zmq的脚本

import zmq

import sys

context = zmq.Context()

print("Connecting to server...")

socket = context.socket(zmq.REQ)

socket.connect("tcp://localhost:5555")

while True:

input1 = input("请输入内容：").strip()

if input1 == 'b':

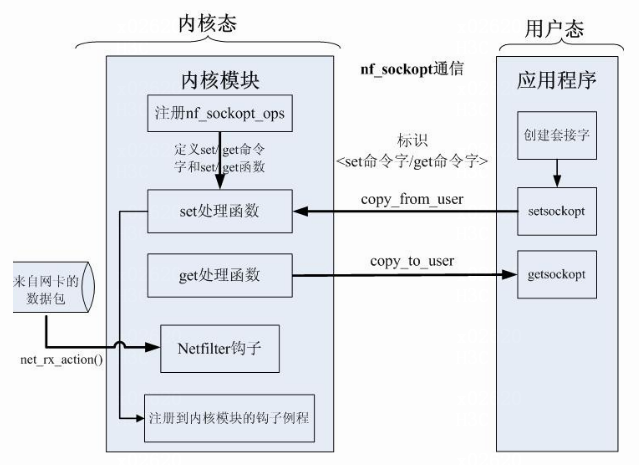
sys.exit()

socket.send(input1.encode('utf-8'))

message = socket.recv()

print("Received reply: ", message.decode('utf-8'))

1. PTS下发的修改方式说明：
   1. 在driver/cmdmessage.c中RPC\_开头的函数原来会调用Socket接口通过Tcp连接到CPUB或者CPUC上获取信息，修改后直接调用Reply接口，中间略掉了TCP连接的操作，从而可以在CPUB和CPUC上直接执行。
   2. 涉及到54282的修改函数RPC\_GetCopperSpeed、FUNC\_SwitchClock、FUNC\_SubGetCopperPhyLink、FUNC\_1gCopperGtCtrl、RPC\_SetCopperloop、RPC\_SetCopper、set\_innerloop\_and\_check\_link
2. PTSRPCServer的Zmq接口说明：
   1. PTSDevice::run -> prepare\_control\_plane 调用PTSCardStruct::launch\_control\_plan调用PTSRpcServer::start 启动线程PTSRpcServerInterface::\_rpc\_thread\_cb\_capture ，调用PTSRpcServerReqRes:: \_rpc\_thread\_cb\_capture ->\_rpc\_thread\_cb\_int-> handle\_request-> process\_request-> process\_request\_raw调用PTSJsonRpcV2ParsedObject::execute->\_execute调用PTSRpcCommand::run函数用来解析从客户端发来的命令字。
3. Realtek 芯片sdk的编译方法：
   1. 先修改内核位置：make/Makefile.settings 的OSS\_LINUX\_PATH赋值修改为OSS\_LINUX\_PATH := $(ROOT\_PATH)/../../../kernel-openEuler-22.03-LTS/
   2. make 9311bextc config=1
   3. make menuconfig，设置网卡支持，需要开启 Nic Driver和 Linux Net Driver
   4. make
4. 使用方法：（在x86\_64  linux5.10平台上）
   1. mknod --mode=666 /dev/rtcore c 200 0
   2. insmod module/rtcore/lkm/rtcore.ko
   3. insmod module/rtk/lkm/rtk.ko
   4. insmod module/rtnic/lkm/rtnic.ko
   5. insmod module/rtdrv/lkm/rtdrv.ko
5. Realtek sdk分析：
   1. Sdk的编译分为内核态模式用户态模式两种，在数据结构dal\_mapper\_t中定义了所有的驱动对外接口指针，不同的芯片会根据不同的情况实现驱动接口。Diag接口会通过调用封装后的驱动接口实现不同的调试功能。驱动接口通过调用rtrpc开头的函数实现对内核态函数的调用，实际操作的时候通过socket通知到内核态。
   2. 在内核态模式下，所有的初始化都在内核态完成，用户的调用通过socket接口通知到内核态。内核态模式下的初始化函数包括rtk\_sdk\_init调用了内核态rtk\_init，rtk\_init调用rtk\_chip\_attach，挂接mac和phy芯片；\_rtcore\_dev\_init调用rtcore\_dev\_init，通过ioctl功能和mmap实现了一些硬件操作功能接口；\_rtdrv\_init调用了rtdrv\_init，用来实现芯片的各种复杂功能；\_rtnic\_init调用了内核态的rtnic\_init，实现了一个网卡驱动。rtsdk\_init函数分别调用rtcore\_dev\_init、rtk\_init、rtnic\_init、rtnic\_oob\_init、rtdrv\_init。Rtnic\_init初始化了一个网卡设备，Rtnic\_oob\_init初始化了一个mdio设备，rtcore\_dev\_init通过ioctl功能和mmap实现了一些硬件操作功能接口,rtk\_init调用rtk\_chip\_attach。
   3. 用户态模式下，把寄存器空间映射到用户态，所有对寄存器的操作都在用户态完成，Loal\_init.c中有用户态到内核态的映射过程，函数ioal\_init通过mmap的方式把内核态空间映射到ioal\_db数据结构中。内核态最终通过rtcore\_dev\_init调用rtcore\_mmap实现硬件操作
   4. Mem32.c中有对内存空间操作的用户态实现，最终调用ioal\_pci\_u2k\_mem32\_write和ioal\_pci\_u2k\_mem32\_read函数通过ioctl调用内核态动作，内核态最终通过rtcore\_dev\_init调用rtcore\_ioctl实现硬件的动作。
   5. 内核态初始化流程在rtdrv\_init中调用了rtdrv\_modules\_init，里面把各种驱动操作函数映射到了全局数据结构rtdrv\_module\_db\_set和rtdrv\_module\_db\_get中，最终的调用通过rtdrv\_setFunc\_dispatch和rtdrv\_getFunc\_dispatch实现。这些调用最终通过nf\_register\_sockopt注册到socket操作选项的链表中。用户态通过setsockopt 和getsockopt 调用即可实现内核态操作，参考rtrpc\_init。



1. Go语言接口的理解：
   1. Go语言定义一个接口，相当于定义了一个新的数据类型，可以使用这个数据类型定义变量，这个变量拥有了这个数据类型的方法，而这个方法如何实现，则由具体的对象来完成，不同的对象可以有不同的实现方法，比如给车定义一个接口，车的速度定义一个方法，不同的车对象设置不同的速度，使用的时候调用速度+车的名字就可以得到速度结果。
2. Go语言make和new的用法：
   1. make只能为[slice](https://so.csdn.net/so/search?q=slice&spm=1001.2101.3001.7020" \t "_blank), map, channel分配内存，并返回一个初始化的值。
   2. new 函数只接受一个参数，这个参数是一个类型，并且返回一个指向该类型内存地址的指针。同时 new 函数会把分配的内存置为零，也就是类型的零值。
3. 关于cpld操作
   1. 在用户态，通过把/dem/mem映射到用户空间的0xdffe0000位置，并存放到全局变量中，然后通过地址偏移找到对应的寄存器未知进行操作
4. Fpgadom的实现说明：
   1. 在内核态，通过调用class\_create创建了一个设备类"fpgai2c-cdev"，然后调用device\_create创建了一个名为/dev/fpga\_dom的设备。
   2. 上述设备提供了四个ioctl的访问接口其中DRV\_BSP\_FPGA\_I2C\_RD、DRV\_BSP\_FPGA\_I2C\_WR接口调用dom\_rw通过pci接口读写寄存器实现对fpga的操作；而DRV\_BSP\_FPGA\_PE\_RD、DRV\_BSP\_FPGA\_PE\_WR接口则通过dma的方式从内存中读写寄存器信息，上述两组接口配合使用，需要光模块数据的时候，用户态调用I2C操作接口把需要的信息写入到pci寄存器空间，然后调用pe操作dma方式从光模块中读出数据并返回，需要写入光模块的时候，同样调用I2C操作接口把需要的信息写入到pcie寄存器空间，然后调用pe操作dma方式把数据写入到光模块。
   3. 在内核态创建了一个设备类dware-dev，然后创建了一个同名设备。该设备实现了两个接口，一个读一个写，操作方式是直接读写全局地址空间
   4. 在用户态通过ioctl功能调用上述读写功能实现了flash更新的使能和去使能。对于逻辑文件的读写则采用了直接文件操作的方式，逻辑文件的flash存放位置在/dev/mtdblock0，这个设备文件是初始化的时候创建的，为什么能对应逻辑文件的存放位置没搞清楚。
   5. Pci设备的驱动和上述具体设备的驱动分离放在了pci.c文件中，里面包括中断和ioremp等操作。
5. 关于openbmc的使用
   1. 安装ubuntu 20.04版本
   2. 安装软件sudo apt install git python3-distutils gcc g++ make file wget \

gawk diffstat bzip2 cpio chrpath zstd lz4 bzip2

* 1. 下载openbmc git clone <https://github.com/openbmc/openbmc>
  2. cd openbmc 执行. setup evb-ast2600
  3. 执行bitbake obmc-phosphor-image
  4. 提取软件代码
     1. 查找软件包bitbake -s | grep <softname>
     2. 提取软件包，以提取u-boot为例devtool modify u-boot
  5. 驱动设备树文件所在位置linux-kernel/arch/arm/boot/dts

1. openbmc添加图形界面
   1. 修改~/openbmc/build/evb-ast2600/conf/local.conf，添加：

*CORE\_IMAGE\_EXTRA\_INSTALL += "webui-vue"*

* 1. 新建image文件 ~/openbmc/meta-phosphor/recipes-phosphor/image/obmc-phosphor-image.bbappend，写入配置：

*OBMC\_IMAGE\_EXTRA\_INSTALL\_${MACHINE} += "webui-vue"*

* 1. 新建文件 ~/openbmc/meta-phosphor/recipes-phosphor/packagegroups/packagegroup-obmc-apps.bbappend，写入配置：

*RDEPENDS\_${PN}-inventory\_${MACHINE} += "webui-vue"*

1. openbmc添加自定义文件
   1. 创建recipes-phosphor/helloworld/hello.bb，内容如下：

如果添加网上下载文件：

*DESCRIPTION = "GNU Helloworld application"*

*SECTION = "examples"*

*LICENSE = "GPLv3"*

*LIC\_FILES\_CHKSUM = "file://COPYING;md5=d32239bcb673463ab874e80d47fae504"*

*PV = "2.9"*

*SRC\_URI = "${GNU\_MIRROR}/hello/hello-${PV}.tar.gz"*

*SRC\_URI[sha256sum] = "ecbb7a2214196c57ff9340aa71458e1559abd38f6d8d169666846935df191ea7"*

*inherit autotools-brokensep gettext*

或许添加自己写的文件：

*SUMMARY = "Chassis test hello "*

*DESCRIPTION = "this is a hello test app"*

*FILESEXTRAPATHS\_append := "${THISDIR}/files:"*

*LICENSE = "CLOSED"*

*LIC\_FILES\_CHKSUM = ""*

*SRC\_URI += " \*

*file://CMakeLists.txt \*

*file://testhello.cpp \*

*"*

*S = "${WORKDIR}/"*

*inherit cmake*

或许添加脚本文件：

*FILESEXTRAPATHS\_append := "${THISDIR}/files:"*

*LICENSE = "Apache-2.0"*

*LIC\_FILES\_CHKSUM = "file://${COREBASE}/meta/files/common-licenses/Apache-2.0;md5=89aea4e17d99a7cacdbeed46a0096b10"*

*S = "${WORKDIR}/"*

*SRC\_URI = "file://testsh.sh "*

*do\_install() {*

*install -d ${D}/${bindir}*

*install -m 0755 ${S}testsh.sh ${D}/${bindir}/*

*}*

* 1. 修改image文件 ~/openbmc/meta-phosphor/recipes-phosphor/image/obmc-phosphor-image.bbappend，写入配置：

*OBMC\_IMAGE\_EXTRA\_INSTALL\_${MACHINE} += "hello"*

* 1. 修改conf文件 ~/openbmc/build/evb-ast2600/conf/local.conf，写入配置：

*OBMC\_IMAGE\_EXTRA\_INSTALL\_${MACHINE} += "hello"*

1. 构建initramfs的方法：

dracut test.img

1. 解压

mv test.img test.img.gz

gunzip test.img.gz

cpio -idmv < test.img

1. 解决sshd root无法登陆的方法：
   1. /etc/ssh/sshd\_config中，去掉UsePAM的配置
2. 解决vsftpd root无法登陆的方法：
   1. 创建/etc/shells
   2. 添加内容

/bin/sh

/bin/bash

/sbin/nologin

1. Gdb安装需要依赖的包：

Libipt、babel、guile、gdb-header、gdb

1. 去掉linux登陆密码的方法

Vim /etc/passwd

root::0:0::/root:/bin/sh

Vim /etc/shadow

root::0:99999:7:::

1. Vpp在openeuler22.03上的编译方法
   1. 卸载原有的ninja，然后手工安装ninja1.11.1
   2. 编辑makefile，搜索centos-8，把对应的地方复制，并把关键字替换成openEuler-22.03；去掉epel-release
   3. 编辑./extras/rpm/vpp.spec，把make install-dep替换成make install-dep -f /root/working/vpp/Makefile
   4. 按顺序make install-dep /make install-ext-deps /make build /make pkg-rpm
2. Vpp启动方法
   1. 编辑/etc/vpp/startup.conf，修改如下内容

dpdk｛

dev 0000:18:00.0 {

name eth0

}

｝

plugins {

path /usr/lib/vpp\_plugins

}

* 1. systemctl start vpp
  2. vppctl

1. Ovs编译
   1. cd openvswitch-3.1.2/

./boot.sh

./configure --with-dpdk=yes

make -j20

make install

1. ovs启动流程

mkdir /usr/local/var/run/openvswitch/ -p

mkdir /usr/local/etc/openvswitch/ -p

ovsdb-tool create /usr/local/etc/openvswitch/conf.db /usr/local/share/openvswitch/vswitch.ovsschema

ovsdb-server --remote=punix:/usr/local/var/run/openvswitch/db.sock \

--remote=db:Open\_vSwitch,Open\_vSwitch,manager\_options \

--private-key=db:Open\_vSwitch,SSL,private\_key \

--certificate=db:Open\_vSwitch,SSL,certificate \

--bootstrap-ca-cert=db:Open\_vSwitch,SSL,ca\_cert \

--pidfile --detach

ovs-vsctl --no-wait init

ovs-vswitchd --pidfile --detach

/usr/local/share/openvswitch/scripts/ovs-ctl start

ovs-vsctl --no-wait set Open\_vSwitch . other\_config:dpdk-init=true

ovs-vsctl --no-wait set Open\_vSwitch . other\_config:dpdk-socket-mem="1024,0"

ovs-vsctl set Open\_vSwitch . other\_config:pmd-cpu-mask=0x10

ovs-vsctl get Open\_vSwitch . other\_config:dpdk-socket-mem

ovs-vsctl get Open\_vSwitch . other\_config:pmd-cpu-mask

ovs-vsctl get Open\_vSwitch . other\_config:dpdk-init

ovs-vswitchd unix:/var/run/openvswitch/db.sock \

-vconsole:emer -vsyslog:err -vfile:info --mlockall --no-chdir \

--log-file=/var/log/openvswitch/ovs-vswitchd.log \

--pidfile=/var/run/openvswitch/ovs-vswitchd.pid \

--detach --monitor

ovs-vsctl add-br br0 -- set bridge br0 datapath\_type=netdev

ovs-vsctl add-br br1 -- set bridge br1 datapath\_type=netdev

ovs-vsctl show

dpdk-devbind --status

ovs-vsctl add-port br0 dpdk0 -- set Interface dpdk0 type=dpdk options:dpdk-devargs=0000:17:00.0

ovs-vsctl add-port br1 dpdk1 -- set Interface dpdk1 type=dpdk options:dpdk-devargs=0000:18:00.0

1. dpdk编译配置
   1. For Ubuntu: sudo apt-get install libnuma-dev

For Red Hat: sudo yum install numactl-devel

* 1. 在grub中增加GRUB\_CMDLINE\_LINUX="default\_hugepagesz=1GB hugepagesz=1G hugepages=20"，然后update-grub
  2. git clone http://dpdk.org/git/dpdk-stable

cd dpdk-stable

git checkout v22.11/v21.11/v20.11

git clone git://dpdk.org/dpdk-kmods

cp -r <dpdk\_sw\_database>/drivers/net/qdma ./drivers/net/

#cp -r <dpdk\_sw\_database>/examples/qdma\_testapp ./examples/

* 1. cd <server\_dir>/<dpdk\_test\_area>/dpdk-stable

meson build

cd build

ninja

ninja install

ldconfig

* 1. cd <server\_dir>/<dpdk\_test\_area>/dpdk-stable/dpdk-kmods/linux/igb\_uio

make

1. 绿区代理设置
   1. github代理设置

git config --global https.proxy http://x02620:20160618aF@proxy.h3c.com:8080

git config --global https.sslverify false

git config --global http.proxy http://x02620:20160618aF@proxy.h3c.com:8080

git config --global http.sslverify false

git config --global http.proxyauthmethod basic

* 1. wget代理设置，编辑/etc/profile.d/proxy.sh，输入如下内容

export http\_proxy="http://x02620:20160618aF@proxy.h3c.com:8080/"

export https\_proxy="http://x02620:20160618aF@proxy.h3c.com:8080/"

export ftp\_proxy="http://x02620:20160618aF@proxy.h3c.com:8080/"

export no\_proxy="127.0.0.1,localhost,10.153.3.130"

source /etc/profile.d/proxy.sh

* 1. npm代理设置

npm config set proxy http://x02620:20160618aF@proxy.h3c.com:8080

npm confi set https-proxy http://x02620:20160618aF@proxy.h3c.com:8080

1. 文件系统制作方法
   1. 用dracut制作ramfs，并解压
   2. 把docker中的文件系统拷贝到ramfs中
   3. 删除/etc目录，然后拷贝老系统中的/etc目录到新系统中
   4. 下载systemd并解压到新系统中，拷贝usr/lib/systemd/libsystemd-shared-249.so到usr/lib64目录下
   5. 下载libseccomp并解压到当前目录
   6. 解决login启动报错，安装util-linux
   7. 解决dbus启动报错，下载dbus并解压到系统中，拷贝dbus-daemon覆盖新系统，拷贝 /usr/share/dbus-1/system.conf到新系统中
   8. 构建系统常用工具列表

babel-2.9.1-1.oe2203.noarch.rpm gpm-1.20.7-25.oe2203.x86\_64.rpm net-tools-2.10-3.oe2203.x86\_64.rpm

babeltrace-1.5.8-2.oe2203.x86\_64.rpm guile-2.0.14-18.oe2203.x86\_64.rpm openssh-8.8p1-22.oe2203.x86\_64.rpm

cjson-1.7.15-1.oe2203.x86\_64.rpm i2c-tools-4.3-1.oe2203.x86\_64.rpm openssh-clients-8.8p1-22.oe2203.x86\_64.rpm

cpio-2.13-8.oe2203.x86\_64.rpm ipmitool-1.8.18-18.oe2203.x86\_64.rpm openssh-server-8.8p1-17.oe2203.x86\_64.rpm

dbus-1.12.20-10.oe2203.x86\_64.rpm iputils-20210722-6.oe2203.x86\_64.rpm pam-1.5.2-6.oe2203.x86\_64.rpm

dbus-daemon-1.12.20-10.oe2203.x86\_64.rpm kmod-29-3.oe2203.x86\_64.rpm passwd-0.80-9.oe2203.x86\_64.rpm

dbus-devel-1.12.20-10.oe2203.x86\_64.rpm libaio-0.3.112-2.oe2203.x86\_64.rpm pciutils-3.7.0-2.oe2203.x86\_64.rpm

dbus-tools-1.12.20-10.oe2203.x86\_64.rpm libcgroup-0.42.2-2.oe2203.x86\_64.rpm setserial-2.17-51.oe2203.x86\_64.rpm

device-mapper-1.02.181-4.oe2203.x86\_64.rpm libipt-2.0.4-1.oe2203.x86\_64.rpm shadow-4.9-7.oe2203.x86\_64.rpm

dhcp-4.4.2-15.oe2203.x86\_64.rpm libseccomp-2.5.3-1.oe2203.x86\_64.rpm systemd-249-46.oe2203.x86\_64.rpm

dmidecode-3.3-3.oe2203.x86\_64.rpm libsemanage-3.3-3.oe2203.x86\_64.rpm tftp-5.2-30.oe2203.x86\_64.rpm

docker-engine-18.09.0-323.oe2203.x86\_64.rpm libtool-ltdl-2.4.6-34.oe2203.x86\_64.rpm tftp-server-5.2-30.oe2203.x86\_64.rpm

ethtool-5.15-1.oe2203.x86\_64.rpm libuser-0.63-4.oe2203.x86\_64.rpm vim-enhanced-9.0-15.oe2203.x86\_64.rpm

expect-5.45.4-6.oe2203.x86\_64.rpm lockdev-1.0.4-0.31.oe2203.x86\_64.rpm vsftpd-3.0.3-33.oe2203.x86\_64.rpm

gc-8.0.6-1.oe2203.x86\_64.rpm minicom-2.8-1.oe2203.x86\_64.rpm

gdb-11.1-1.oe2203.x86\_64.rpm mtd-utils-2.1.4-2.oe2203.x86\_64.rpm

1. 解决ramfs docker不可用问题

mkdir -p /etc/systemd/system/docker.service.d

cat >/etc/systemd/system/docker.service.d/10-ramdisk.conf <<EOF

[Service]

Environment=DOCKER\_RAMDISK=true

EOF

systemctl daemon-reload

systemctl restart docker

1. dhcp服务器增加下载文件

option bootfile-name "test.elf";

1. dpdk内存设置位置

/sys/devices/system/node/node0/hugepages/hugepages-2048kB/

在启动文件中启动大内存

default\_hugepagesz=1G hugepagesz=1G hugepages=20

1. 把文件系统打包成压缩格式

find \* | cpio -R root:root -H newc -o | gzip > ../linuxboot/rootfs.gz

1. 制作linux启动iso
   1. Bios

xorriso -as mkisofs -o ../testlinux.iso -b isolinux.bin -c boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table ./

* 1. uefi

xorriso -as mkisofs -isohybrid-mbr ../syslinux-5.10/mbr/isohdpfx.bin -c boot/boot.cat -e boot/uefi.img -no-emul-boot -isohybrid-gpt-basdat -o ../testuef.iso ../iso

1. c语言忽略Ctrl+C关键命令操作

/\* Ignore Ctrl+C signal. \*/

signal(SIGINT, SIG\_IGN);

1. linux支持KDB
   1. 在cmdline.txt中添加

kdb.cmd\_enable=1

* 1. 在串口下输入： echo g > /proc/sysrq-trigger 命令，就可以主动进入kdb了

1. 用yum制作文件系统

mkdir /initrd

yum install -y iproute openssh vsftpd tftp-server dhcp pciutils net-tools iputils i2c-tools libuser passwd expect kmod docker mtd-utils dbus cjson-devel ipmitool gdb yum --installroot=/initrd

复制原有的/etc/systemd覆盖新的目录

复制原有的/root覆盖新的目录

删除替换/usr/share/locale目录，删除/var/cache目录以节省空间

复制原有的/etc/passwd /etc/shadow覆盖新的文件

复制原有的/etc/dhcp/dhcpd.conf覆盖新的文件

复制原有目录下/usr/lib64/libmodbus\*到新的目录

1. 制作linux u盘启动盘的方法：
   1. 用rufus把u盘的分区类型修改成grub2
   2. 下载grub for windows，输入如下命令

grub-install.exe --boot-directory=F:\ --efi-directory=F: --removable --target=x86\_64-efi

* 1. 把linux系统下的grub.cfg拷贝到u盘的grub目录下
  2. 修改grub.cfg的menuentry选项

linux /boot/bzImage rw rdinit=/sbin/init rootfstype=tmpfs

initrd /boot/ramfs.gz

* 1. 在u盘创建boot目录，并把bzImage和ramfs.gz拷贝到这个目录下

1. 在ubuntu server上安装openbox

sudo apt-get install xserver-xorg-core

sudo apt-get install x-window-system-core

sudo apt-get install openbox

startx 就可以进入桌面，右键菜单运行程序

obmenu 可以创建右键菜单

obconf 可以管理窗口，比如外观设置为CIML的Unity窗口样式，鼠标设置为自动获得焦点，桌面留1个

lxappearance 可以设置图标主题为Humanity，鼠标指针为DMZ（安装dmz-cursor-theme）

修改后右键 restart 生效

1. 制作可启动iso
   1. 把已有的iso mount到一个目录下
   2. 用rsync命令把iso中的内容拷贝到本地
   3. 修改EFI/BOOT/grub.cfg，添加关键信息如下

inst.stage2=hd:LABEL=openEuler-22.03-LTS-x86\_64 inst.ks=hd:LABEL=openEuler-22.03-LTS-x86\_64:/ks.cfg inst.text console=tty0 console=ttyS0,115200

* 1. 修改ks.cfg文件，关键信息如下：

%post --nochroot

# 拷贝安装包到新系统中

cd /mnt/install/repo/devm

tarname="opt usr etc root var"

\cp -r ${tarname} /mnt/sysimage/

%end

%post

cd /opt

tarname="setup.sh"

sh ${tarname}

%end

* 1. 进入本地目录，执行如下命令

xorriso -as mkisofs \

-isohybrid-mbr /usr/share/syslinux/isohdpfx.bin \

-V openEuler-22.03-LTS-x86\_64 \

-c isolinux/boot.cat \

-b isolinux/isolinux.bin \

-no-emul-boot \

-boot-load-size 4 \

-boot-info-table \

-eltorito-alt-boot \

-e images/efiboot.img \

-no-emul-boot \

-isohybrid-gpt-basdat \

-o openEuler-2203-tm.iso \

mkiso/

* 1. 修改EFI/BOOT/grub.cfg，添加关键信息如下

inst.stage2=hd:LABEL=openEuler-22.03-LTS-x86\_64 inst.ks=hd:LABEL=openEuler-22.03-LTS-x86\_64:/ks.cfg inst.text console=tty0 console=ttyS0,115200

1. 打开串口的方法：

echo "GRUB\_CMDLINE\_LINUX\_DEFAULT=\"console=tty0 console=ttyS0,115200\"" >> /etc/default/grub

grub2-mkconfig -o /boot/efi/EFI/openEuler/grub.cfg

1. 构建一个可执行service文件例子

vim /usr/lib/systemd/system/zdy.service

[Unit]

Description=描述

Environment=环境变量或参数(系统环境变量此时无法使用)

After=network.target

[Service]

Type=forking

EnvironmentFile=所需环境变量文件或参数文件

ExecStart=启动命令(需指定全路径)

ExecStop=停止命令(需指定全路径)

User=以什么用户执行命令

[Install]

WantedBy=multi-user.target

1. openEuler安装包的制作方法：
   1. 拷贝文件到iso目录下

mkdir /mnt/src

mount –o loop /root/openEuler-22.03-LTS-x86\_64-dvd.iso /mnt/src

mkdir /root/iso

cp -r /mnt/src/. /root/iso

* 1. 修改grub.cfg

### BEGIN /etc/grub.d/10\_linux ###

menuentry 'Install openEuler 22.03-LTS from byobmc' --class openEuler --class gnu-linux --class gnu --class os {

# linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=openEuler-22.03-LTS-x86\_64 fpi\_to\_tail=off

linuxefi /images/pxeboot/vmlinuz ks=cdrom:/ks/ks.cfg console=tty0 console=ttyS0,115200

initrdefi /images/pxeboot/initrd.img

}

menuentry 'Install openEuler 22.03-LTS from openbmc or usb' --class openEuler --class gnu-linux --class gnu --class os {

linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=openEuler-22.03-LTS-x86\_64 inst.ks=hd:LABEL=openEuler-22.03-LTS-x86\_64:/ks.cfg inst.text console=tty0 console=ttyS0,115200 fpi\_to\_tail=off

# linuxefi /images/pxeboot/vmlinuz ks=cdrom:/ks/ks.cfg console=tty0 console=ttyS0,115200

initrdefi /images/pxeboot/initrd.img

}

submenu 'Troubleshooting -->' {

menuentry 'Install openEuler 22.03-LTS in basic graphics mode' --class openEuler --class gnu-linux --class gnu --class os {

linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=openEuler-22.03-LTS-x86\_64 nomodeset fpi\_to\_tail=off

initrdefi /images/pxeboot/initrd.img

}

menuentry 'Rescue a openEuler system' --class openEuler --class gnu-linux --class gnu --class os {

linuxefi /images/pxeboot/vmlinuz inst.stage2=hd:LABEL=openEuler-22.03-LTS-x86\_64 rescue fpi\_to\_tail=off

initrdefi /images/pxeboot/initrd.img

}

}

* 1. 修改isolinux.cfg文件

default vesamenu.c32

timeout 600

display boot.msg

# Clear the screen when exiting the menu, instead of leaving the menu displayed.

# For vesamenu, this means the graphical background is still displayed without

# the menu itself for as long as the screen remains in graphics mode.

menu clear

menu background splash.png

menu title openeuler

menu vshift 8

menu rows 18

menu margin 8

#menu hidden

menu helpmsgrow 15

menu tabmsgrow 13

menu color border \* #00000000 #00000000 none

menu color sel 0 #ffffffff #00000000 none

menu color title 0 #ff7ba3d0 #00000000 none

menu color tabmsg 0 #ff3a6496 #00000000 none

menu color unsel 0 #84b8ffff #00000000 none

menu color hotsel 0 #84b8ffff #00000000 none

menu color hotkey 0 #ffffffff #00000000 none

menu color help 0 #ffffffff #00000000 none

menu color scrollbar 0 #ffffffff #ff355594 none

menu color timeout 0 #ffffffff #00000000 none

menu color timeout\_msg 0 #ffffffff #00000000 none

menu color cmdmark 0 #84b8ffff #00000000 none

menu color cmdline 0 #ffffffff #00000000 none

label linux

menu label ^Install openeuler

kernel vmlinuz

append initrd=initrd.img inst.stage2=hd:/dev/cdrom inst.ks=cdrom:/ks/ks.cfg quiet

label local

menu label Boot from ^local drive

localboot 0xffff

* 1. 安装一个最小化的linux系统，然后安装必要的软件，并拷贝到当前系统下

rpm -qa|sort > 1.txt

scp 1.txt 10.114.201.103:/root/iso

* 1. 提取rpm包信息

cd /root/iso

mkdir all

mv Packages/\*.rpm all

for FN in `cat 1.txt`; do mv all/${FN}.rpm Packages; done

rm -rf 1.txt all

* 1. 修改ks.cfg

#version=DEVEL

text

ignoredisk --only-use=sda

# Partition clearing information

clearpart --none --initlabel

# Use graphical install

#graphical

# Use CDROM installation media

cdrom

# Keyboard layouts

keyboard --vckeymap=cn --xlayouts='us'

# System language

lang en\_US.UTF-8

bootloader --location=mbr --driveorder=sda --append="crashkernel=auto"

#bootloader location=mbr

# Network information

network --bootproto=dhcp --device=enp4s0 --onboot=yes --ipv6=auto --no-activate

network --hostname=localhost

# Root password

rootpw openEuler

firewall --disabled

selinux --disabled

# Run the Setup Agent on first boot

firstboot --disabled

# System services

services --disabled="chronyd"

# System timezone

timezone Asia/Shanghai

#autopart --type=lvm

clearpart --all --initlabel

part /boot --fstype xfs --size=512

part swap --fstype="swap" --size=2048

part / --fstype xfs --size=1 --grow

reboot

%packages

@^minimal-environment

%end

#enable kdump

#%addon com\_redhat\_kdump --enable --reserve-mb='128'

%post

#enable kdump

sed -i "s/ ro / ro crashkernel=256M,high /" /boot/efi/EFI/openEuler/grub.cfg

%end

* 1. 修改normal.xml

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE comps

PUBLIC '-//openEuler//DTD Comps info//EN'

'comps.dtd'>

<comps>

<group>

<id>core</id>

<name>Core</name>

<name xml:lang="zh\_CN">核心</name>

<description>Smallest possible installation</description>

<description xml:lang="zh\_CN">最小安装</description>

<default>true</default>

<uservisible>false</uservisible>

<packagelist>

<packagereq type="mandatory">audit</packagereq>

<packagereq type="mandatory">kernel</packagereq>

<packagereq type="mandatory">basesystem</packagereq>

<packagereq type="mandatory">bash</packagereq>

<packagereq type="mandatory">coreutils</packagereq>

<packagereq type="mandatory">cronie</packagereq>

<packagereq type="mandatory">curl</packagereq>

<packagereq type="mandatory">dnf</packagereq>

<packagereq type="mandatory">e2fsprogs</packagereq>

<packagereq type="mandatory">filesystem</packagereq>

<packagereq type="mandatory">firewalld</packagereq>

<packagereq type="mandatory">glibc</packagereq>

<packagereq type="mandatory">grubby</packagereq>

<packagereq type="mandatory">hostname</packagereq>

<packagereq type="mandatory">initscripts</packagereq>

<packagereq type="mandatory">iproute</packagereq>

<packagereq type="mandatory">iprutils</packagereq>

<packagereq type="mandatory">iputils</packagereq>

<packagereq type="mandatory">irqbalance</packagereq>

<packagereq type="mandatory">kbd</packagereq>

<packagereq type="mandatory">kexec-tools</packagereq>

<packagereq type="mandatory">less</packagereq>

<packagereq type="mandatory">man-db</packagereq>

<packagereq type="mandatory">ncurses</packagereq>

<packagereq type="mandatory">openssh</packagereq>

<packagereq type="mandatory">openssh-server</packagereq>

<packagereq type="mandatory">openssh-clients</packagereq>

<packagereq type="mandatory">openEuler-release</packagereq>

<packagereq type="mandatory">openEuler-latest-release</packagereq>

<packagereq type="mandatory">parted</packagereq>

<packagereq type="mandatory">passwd</packagereq>

<packagereq type="mandatory">policycoreutils</packagereq>

<packagereq type="mandatory">procps-ng</packagereq>

<packagereq type="mandatory">rng-tools</packagereq>

<packagereq type="mandatory">rootfiles</packagereq>

<packagereq type="mandatory">rpm</packagereq>

<packagereq type="mandatory">selinux-policy-targeted</packagereq>

<packagereq type="mandatory">setup</packagereq>

<packagereq type="mandatory">shadow</packagereq>

<packagereq type="mandatory">sssd</packagereq>

<packagereq type="mandatory">sudo</packagereq>

<packagereq type="mandatory">systemd</packagereq>

<packagereq type="mandatory">tuned</packagereq>

<packagereq type="mandatory">util-linux</packagereq>

<packagereq type="mandatory">vim-minimal</packagereq>

<packagereq type="mandatory">xfsprogs</packagereq>

<packagereq type="mandatory">yum</packagereq>

<packagereq type="mandatory">wget</packagereq>

<packagereq type="default">NetworkManager</packagereq>

<packagereq type="default">NetworkManager-config-server</packagereq>

<packagereq type="default">authselect</packagereq>

<packagereq type="default">dnf-plugins-core</packagereq>

<packagereq type="default">dracut-config-rescue</packagereq>

<packagereq type="default">kernel-tools</packagereq>

<packagereq type="default">sysfsutils</packagereq>

<packagereq type="default">linux-firmware</packagereq>

<packagereq type="default">lshw</packagereq>

<packagereq type="default">lsscsi</packagereq>

<packagereq type="default">rsyslog</packagereq>

<packagereq type="default">security-tool</packagereq>

<packagereq type="default">sg3\_utils</packagereq>

<packagereq type="optional">dracut-config-generic</packagereq>

<packagereq type="optional">dracut-network</packagereq>

<packagereq type="optional">rdma-core</packagereq>

<packagereq type="optional">selinux-policy-mls</packagereq>

</packagelist>

</group>

<environment>

<id>minimal-environment</id>

<name>Minimal Install</name>

<name xml:lang="zh\_CN">最小安装</name>

<description>Basic functionality.</description>

<description xml:lang="zh\_CN">基本功能。</description>

<display\_order>1</display\_order>

<grouplist>

<groupid>core</groupid>

</grouplist>

</environment>

</comps>

* 1. 在xml文件的default中添加上述步骤c中安装的的软件

<packagereq type="default">net-tools</packagereq>

<packagereq type="default">gdb</packagereq>

<packagereq type="default">ipmitool</packagereq>

* 1. 重新生成 repodata

cd /root/iso

rm -rf repodata

createrepo -g normal.xml ./

* 1. 生成iso文件

xorriso -as mkisofs \

-isohybrid-mbr /usr/share/syslinux/isohdpfx.bin \

-V openeuler-2203 \

-c isolinux/boot.cat \

-b isolinux/isolinux.bin \

-no-emul-boot \

-boot-load-size 4 \

-boot-info-table \

-eltorito-alt-boot \

-e images/efiboot.img \

-no-emul-boot \

-isohybrid-gpt-basdat \

-o openEuler-2203-temp.iso \

iso/

1. Openbmc对内核的修改：
   1. 解决ncsi-netlink异常

Security options---

去掉Harden common str/mem functions against buffer overflows

* 1. Nsci口在主cpu重启时保持up

Networking support---

Networking option---

添加Get NCSI OEM MAC Address

添加Keep PHY Link up

* 1. 内核修改下次持久生效

把修改配置放到文件meta-aspeed/recipes-kernel/linux/linux-aspeed/aspeed-g6/defconfig中

* 1. 添加或修改内核代码

把修改放到../../meta-aspeed/recipes-kernel/linux/linux-aspeed中，然后修改../../meta-aspeed/recipes-kernel/linux/linux-aspeed\_git.bb文件添加SRC\_URI变量可以把当前文件添加到编译临时目录中，用do\_configure:prepend通过$(WORKDIR)可以找到对应文件并拷贝到编译目录下

1. 用自己编译的内核制作linux安装包
   1. 获取一个已有的linux安装包
   2. 把安装包挂载到目录下
   3. 拷贝出安装包下所有的内容
   4. 用自己编译的内核替换images\pxeboot\vmlinuxz
   5. 用自己编译的内核替换isolinux\vmlinuxz
   6. 把images\install.img挂载到install目录下。
   7. 拷贝出install\LiveCD\root.img 到build\LiveCD\
   8. 把拷贝出的build\LiveCd\root.img挂载到temp目录下
   9. 用自己编译的内核替换temp\boot\vmlinux\*
   10. mksquashfs build/ install.img -comp xz;
2. 操作系统支持intel ssd硬盘的修改
   1. Make menuconfig进入device drivers
   2. 进入Serial ATA and Parallel ATA drivers，增加intel相关的选项
   3. 进入File systems选项，进入DOS/FAT/EXFAT/NT Filesystems选项
   4. 增加VFAT fs support
   5. 把default codepage修改成936
   6. 把default iocharset修改成utf8
   7. 进入Native language support选项
   8. 增加simple Chinese charset选项
3. 操作系统增加bios flash读写功能
   1. Make menuconfig进入device drivers
   2. 进入 Memory Technology Device(MTD) support
   3. 进入 SPI NOR device support
   4. 把两个intel相关的flash选项选上
   5. 执行echo "GRUB\_CMDLINE\_LINUX\_DEFAULT=\"console=tty0 console=ttyS0,115200 intel-spi.writeable=1 kdb.cmd\_enable=1\"" >> /etc/default/grub
   6. grub2-mkconfig -o /boot/efi/EFI/openEuler/grub.cfg
4. 替换linux内核
   1. 修改Makefile中的参数EXTRAVERSION = -60.18.0.50.oe2203.x86\_64
   2. 拷贝替换系统的.config文件
   3. Make menuconfig进入Enable Loadable module support
   4. 增加module versioning support选项
   5. 编译内核rpm包的命令：make INSTALL\_MOD\_STRIP=1 rpm-pkg -j20
5. 将普通用户添加都docker组的方法
   1. sudo groupadd docker
   2. sudo gpasswd -a test docker
   3. newgrp docker
6. openeuler上设置samba

yum install samba

smbpasswd -a root

systemctl start smb

setenforce 0

systemctl stop firewalld

1. byobmc的编译说明：
   1. 把[www.example.com](http://www.example.com)修改成[www.google.com](http://www.google.com)
   2. 去掉conf/local.conf中的本地镜像ip信息配置
2. 脚本方式免密

ssh-keygen -q -t rsa -N '' -f ~/.ssh/id\_rsa <<<y 2>&1 >/dev/null

sshpass -p 123456 ssh-copy-id -o StrictHostKeyChecking=no root@10.114.201.104

1. 关于sysrq
   1. 打开方法

echo 1 > /proc/sys/kernel/sysrq打开SysRq组合键

修改/etc/sysctl.conf中的kernel.sysrq=1

* 1. 使用方法

echo ‘快捷键’ > /proc/sysrq-trigger。

或者同时按下Alt 和 Print Screen 键，然后按下快捷键，不过在xwindows下无效。

快捷键如下

* m - 导出关于内存分配的信息，
* t - 导出线程状态信息，
* p - 导出当前CPU寄存器信息和标志位的信息，
* c - 故意让系统崩溃(在使用netdump或diskdump的时候有用)，
* s - 即时同步所有挂载的文件系统，
* u - 即时重新挂载所有的文件系统为只读，
* b - 即时重新启动系统，
* o - 即时关机(如果机器设置并支持此项功能)。

1. 构建elf包

objcopy -I binary -O elf64-x86-64 --add-section .bzImage=$(KERNEL\_TARGET)

添加字段到elf包中

Objcopy TM\_KERNEL.elf –add-section .cmdline=cmdline.txt

提取字段

Objcopy TM\_KERNEL.elf --dump-section .cmdline=cmdline.txt

删除字段

Objcopy TM\_KERNEL.elf --remove-section .cmdline=cmdline.txt

1. Dhcpd文件配置

default-lease-time 600;

max-lease-time 7200;

ddns-update-style none;

authoritative;

subnet 169.255.1.0 netmask 255.255.255.0 {

range 169.255.1.10 169.255.1.39;

option routers 169.255.1.1;

option subnet-mask 255.255.255.0;

option bootfile-name "TM\_KERNEL.elf";

option dhcp-lease-time 3600;

option dhcp-renewal-time 1800;

option dhcp-rebinding-time 2700;

}