Atlas 200 DK
V100R020C00
使用指南

文档版本	01
发布日期	2021-06-25

华为技术有限公司

版权所有 © 华为技术有限公司 2021。 保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

HUAWEI和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 简介.....1
2 准备配件及开发服务器.....2
3 硬件安装.....4
3.1 拆卸上盖.....4
3.2 安装摄像头（主板为 IT21DMDA）.....6
3.3 安装摄像头（主板为 IT21VDMB）.....11
4 软件安装.....16
4.1 安装运行环境.....16
4.1.1 制作 SD 卡.....16
4.1.1.1 简介.....16
4.1.1.2 读卡器场景.....16
4.1.1.3 无读卡器场景.....21
4.1.2 连接 Atlas 200 DK 开发者板与 Ubuntu 服务器.....26
4.2 安装开发环境.....30
4.2.1 获取软件包.....30
4.2.2 配置 Ubuntu X86 系统.....30
4.2.3 安装开发套件包.....34
4.2.4 部署 Media 模块.....35
4.2.5 配置交叉编译环境.....36
4.2.6 安装 MindStudio.....36
4.3 安装开发环境与运行环境.....36
5 常用操作.....46
5.1 升级与卸载.....46
5.1.1 升级 Atlas 200 DK.....46
5.1.2 升级开发套件包.....51
5.1.3 卸载开发套件包.....51
5.2 上电 Atlas 200 DK 开发者板.....52
5.3 下电 Atlas 200 DK 开发者板.....55
5.4 通过串口连接 Atlas 200 DK.....56
5.5 查看开发者板系统软件的版本.....57
5.6 查看开发者板硬件底板的版本.....58
5.7 查看 Atlas 200 AI 加速模块的版本.....61

5.8 如何查看开发者板的 firmware 版本.....65
5.9 如何查看摄像头所属 Channel.....65
5.10 如何安装 windows 的 USB 网卡驱动.....66
5.11 设置用户有效期.....69
5.12 安装 3.5.2 版本 cmake.....70
5.13 配置系统网络代理.....70
5.14 参数说明.....71

6 FAQ.....73

6.1 run 包安装时提示软件已经安装.....73
6.2 pip3.7.5 install 报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”.....73
6.3 使用 pip3.7.5 install 软件时提示" Could not find a version that satisfies the requirement xxx".....74
6.4 SD 卡制作过程中人为拔卡导致系统存在冗余挂载盘.....75
6.5 Ubuntu 服务器连接开发者板无法建立信任关系.....76
6.6 开发者板无法正常连接 Ubuntu 服务器如何处理.....76

1 简介

华为Atlas 200 DK开发者套件 Atlas 200 Developer Kit（以下简称Atlas 200 DK ）是以华为Ascend 310芯片为核心的一个开发者板形态产品，为开发者提供一站式开发套件，助力开发者快速进行AI应用程序的开发。

本文主要介绍用户在使用Atlas 200 DK开发、运行AI应用程序前的准备工作，包括系统SD卡的制作，Atlas 200 DK与Ubuntu服务器的连接，开发工具的安装等。

Atlas 200 DK的系统框图如下所示：

图 1-1 Atlas 200 DK 与 Mind Studio 连接图

说明

图中的IP 192.168.1.2/192.168.0.2是Atlas 200 DK的IP，在制卡阶段可以进行选择。

Atlas 200 DK主要包含Hi3559 Camera模块以及Atlas 200 AI加速模块，开发工具MindStudio所在PC通过USB接口或者网线与Atlas 200 DK开发者板连接。

开发工具 MindStudio包含了开发套件、以及各个工具模块（如模型管理工具、编译工具、日志工具等），其中开发套件提供了设备侧编译时所依赖的库文件，用户开发使用的工具、依赖库以及公共头文件等。

2 准备配件及开发服务器

介绍使用Atlas 200 DK前需要准备的配件及开发服务器。

配件准备

Atlas 200 DK使用需要用户提前自购如表2-1所示配件。

表 2-1 配件信息

配件名称	描述	推荐型号
SD卡	用于制作Atlas 200 DK开发者板启动系统。	推荐使用经过测试的SD卡： ● 三星UHS-I U3 CLASS 10 64G ● 金士顿UHS-I U1 CLASS 10 64G
摄像头	用于与Atlas 200 DK连接获取视频，参考3.2 安装摄像头（主板为IT21DMDA）和3.3 安装摄像头（主板为IT21VDMB）。	推荐使用树莓派摄像头。 型号：RASPBERRY PI V2.1 若Atlas 200 DK主板为IT21DMDA，使用树莓派摄像头时需要额外购买黄色的15 pin树梅派专用排线。
摄像头支架（可选）	用于固定摄像头，参考3.2 安装摄像头（主板为IT21DMDA）和3.3 安装摄像头（主板为IT21VDMB）。	树莓派透明摄像头支架。
Type-C连接线	用于Ubuntu服务器通过USB方式连接Atlas 200 DK。	支持USB3.0的Type-C连接线。
网线	用于Ubuntu服务器通过网线方式连接Atlas 200 DK。	普通网线，接口类型为RJ45。

版权所有 © 华为技术有限公司

配件名称	描述	推荐型号
串口线（可选）	用于Atlas 200 DK启动灯状态异常或者制卡成功但无法正常与UI Host通信时，通过串口查看启动日志，参考**6.6 开发者板无法正常连接Ubuntu服务器如何处理**。	USB转TTL串口线，3.3V接口电平。

服务器准备

准备一个操作系统为Ubuntu X86架构的服务器，用途如下：

- 为Atlas 200 DK制作SD卡启动盘。读卡器或者Atlas 200 DK会通过USB与此Ubuntu服务器连接，制作Atlas 200 DK的系统启动盘。
- 用于部署开发环境，进行应用程序的开发。
- Ubuntu操作系统的版本可以为：18.04.4或18.04.5。

  请从**http://releases.ubuntu.com/releases/**下载对应版本软件进行安装：

  可以下载桌面版“ubuntu-*{version}*-desktop-**amd64**.iso”，或Server版“ubuntu-*{version}*-server-**amd64**.iso”，*{version}*为操作系统版本。
- Ubuntu操作系统中需要存在python3.x。
- 系统空余空间超过20G。
- 系统内存大于4G。

3 硬件安装

3.1 拆卸上盖
3.2 安装摄像头（主板为IT21DMDA）
3.3 安装摄像头（主板为IT21VDMB）

3.1 拆卸上盖

当需要使用摄像头或内部接口时，需要拆卸Atlas 200 DK的上盖，操作步骤如下所示。

步骤1 查看Atlas 200 DK开发者板上是否有摄像头线缆接出。

- 是，执行**步骤3**。
- 否，执行**步骤2**。

步骤2 当Atlas 200 DK开发者板上没有摄像头线缆接出时，向上按压塑料卡扣，使上盖一端松动，如**图3-1**所示。

图 3-1 拆卸上盖

步骤3 当Atlas 200 DK开发者板上有摄像头线缆接出时，将一字螺丝刀插入上盖于底板之间的凹槽，旋转螺丝刀，将上盖的一端从盒体上撬出，如图3-2中①所示。

图 3-2 拆卸上盖

步骤4 向上取出上盖。

----结束

3.2 安装摄像头（主板为 IT21DMDA）

操作步骤

步骤1 将树莓派摄像头自带的白色排线换成黄色的摄像头排线。

1. 在摄像头一端，拔出摄像头上的黑色排线固定件。如图3-3所示。

图 3-3 排线固定件

2. 取出白色摄像头排线。如图3-4所示。

图 3-4 白色摄像头排线

3. 将黄色摄像头排线宽的一端金属线方向朝上，水平插入摄像头的排线插槽中，直至不能移动。如图3-5所示。

图 3-5 连接摄像头

4. 扣上黑色排线固定件。

步骤2 将摄像头上的固定胶片套在黄色摄像头排线上。如图3-6所示。

图 3-6 安装固定胶片

步骤3 连接摄像头排线到Atlas 200 DK开发者板。

1. 拔出Atlas 200 DK开发者板的摄像头连接器固定件。如图3-7所示。

   图 3-7 拔出黑色固定件

2. 将黄色摄像头排线窄的一端金属线朝上，水平插入Atlas 200 DK开发者板的摄像头连接器CAMERA0或者CAMERA1，直至不能移动。并插回固定件。如图3-8所示。

图 3-8 插回固定件

步骤4 参照拆卸前原位置安装Atlas 200 DK开发者板的上盖。

步骤5 安装摄像头支架。

1. 使用摄像头支架上的夹子夹住固定胶片，如图3-9所示。

图 3-9 安装摄像头支架

2. 如上图所示，安装好摄像头支架，支起摄像头。

说明

- 使用前，请撕下摄像头上的保护膜。
- 摄像头支架的底座上带有双面胶，可用于固定支架在桌面上，确保摄像头安装稳固。

----结束

3.3 安装摄像头（主板为 IT21VDMB）

操作步骤

步骤1 将树莓派摄像头自带的白色排线换成黑色的摄像头排线。

1. 在摄像头一端，拔出摄像头上的黑色排线固定件。如图3-10所示。

图 3-10 拔出排线固定件

2. 取出白色摄像头排线。如图3-11所示。

图 3-11 白色摄像头排线

3. 将黑色摄像头排线丝印显示“TO CAMERA”的一端金属线方向朝上，水平插入摄像头的排线插槽中，直至不能移动。如图3-12所示。

图 3-12 连接摄像头

4. 扣上黑色排线固定件。

步骤2 将摄像头上的固定胶片套在黑色摄像头排线上。

步骤3 连接摄像头排线到Atlas 200 DK开发者板。

须知

- 连接器固定件的打开范围最大为90度，向上打开连接器固定件时请勿超过90度。
- 请勿反方向打开连接器固定件，否则会导致连接器断裂。

1. 将Atlas 200 DK开发者板的摄像头连接器固定件由里向上打开90°。如图3-13所示。

图 3-13 掰开黑色固定件

2. 将黑色摄像头排线丝印显示“TO MAIN BD”的一端黑色线朝上，水平插入Atlas 200 DK开发者板的摄像头连接器CAMERA0或者CAMERA1，直至不能移动。并扣上固定件。如图3-14所示。

图 3-14 扣好固定件

步骤4 参照拆卸前原位置安装Atlas 200 DK开发者板的上盖。

步骤5 安装摄像头支架。

1. 使用摄像头支架上的夹子夹住固定胶片，如图3-15所示。

图 3-15 安装摄像头支架

2. 如上图所示，安装好摄像头支架，支起摄像头。

说明

- 使用前，请撕下摄像头上的保护膜。
- 摄像头支架的底座上带有双面胶，可用于固定支架在桌面上，确保摄像头安装稳固。

----结束

4 软件安装

4.1 安装运行环境
4.2 安装开发环境
4.3 安装开发环境与运行环境

4.1 安装运行环境

4.1.1 制作 SD 卡

4.1.1.1 简介

通过SD卡制作功能可以自行制作Atlas 200 DK 开发者板的系统启动盘。

Atlas 200 DK开发者板SD卡制作方式有如下两种：

- 如果有读卡器，可以将SD卡放入读卡器，将读卡器与Ubuntu服务器的USB连接，然后通过制卡脚本进行SD卡的制作。
- 如果没有读卡器，可以将SD卡放入Atlas 200 DK开发者板卡槽，通过跳线帽/杜邦线进行开发者板相关针脚连接，将开发者板与Ubuntu服务器的USB连接，然后通过制卡脚本进行SD卡的制作。

须知

SD卡制作过程中会自动创建默认用户HwHiAiUser，该用户用来运行应用程序。用户HwHiAiUser缺省登录密码为“**Mind@123**”。

4.1.1.2 读卡器场景

介绍有读卡器的场景下将读卡器与Ubuntu服务器的USB连接，通过制卡脚本制作SD卡的操作。

硬件准备

SD卡：准备一张SD卡，建议容量不低于16G。

说明

制作过程将会对卡做格式化处理，请提前备份数据。

软件包准备

获取开发者板驱动与运行包、Ubuntu操作系统镜像包，下载信息如表4-1所示：

表 4-1 软件包信息

包信息	包名	说明	下载地址
Ubuntu操作系统镜像包	ubuntu-18.04.xx-server-arm64.iso	开发者板操作系统镜像包。 Ubuntu操作系统的版本可以为：18.04.4、18.04.5。 **注意：为ARM版本的Server软件包。**	请从华为开源镜像或者Ubuntu官网下载此软件包。
开发者板驱动包与运行包	Ascend310-driver-*{software version}*-ubuntu18.04.aarch64-minirc.tar.gz	开发者板驱动包，包含OS外围软件、AI软件栈、维测相关软件及驱动。 制作SD卡时，会从driver包中获取Firmware信息，不需要单独获取Firmware组件包。	获取链接 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。
	Ascend310-aicpu_kernels-*{software version}*-minirc.tar.gz	AI CPU算子包。	获取链接 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。

包信息	包名	说明	下载地址
	Ascend-acllib-*{software version}*-ubuntu18.04.aarch64-minirc.run	AscendCL（Ascend Computing Language）库文件安装包，用来编译和运行AscendCL应用程序，包含模型加载及执行功能。 制卡时使用HwHiAiUser用户进行此包的安装，并会自动在HwHiAiUser用户的.bashrc文件中设置LD_LIBRARY_PATH的环境变量： **export LD_LIBRARY_PATH=/home/HwHiAiUser/Ascend/acllib/lib64**	获取链接 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。

须知

软件下载后请保持原命名。

操作步骤

步骤1 请将SD卡放入读卡器，并将读卡器与Ubuntu服务器的USB接口连接。

步骤2 在Ubuntu服务器中执行如下命令安装qemu-user-static、binfmt-support、yaml与交叉编译器。

su - root

执行如下命令更新源：

apt-get update

执行如下命令安装相关python依赖：

pip3 install pyyaml

apt-get install qemu-user-static binfmt-support python3-yaml gcc-aarch64-linux-gnu g++-aarch64-linux-gnu

步骤3 在Ubuntu服务器中以root用户执行如下命令创建制卡工作目录。

mkdir /home/ascend/mksd

制卡目录可任意指定。

步骤4 将软件包准备获取的Ubuntu操作系统镜像包、开发者板所有驱动包上传到制卡工作目录中（例如：“/home/ascend/mksd”）。

步骤5 在制卡工作目录下（例如：“/home/ascend/mksd”）执行如下命令获取制卡脚本。

- 下载制卡入口脚本“make_sd_card.py”。

  从gitee下载：

  wget https://gitee.com/ascend/tools/raw/master/makesd/for_1.0.7.alpha/make_sd_card.py

  从github下载：

  wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.7.alpha/make_sd_card.py

- 下载制作SD卡操作系统的脚本“make_ubuntu_sd.sh”。

  从gitee下载：

  wget https://gitee.com/ascend/tools/raw/master/makesd/for_1.0.7.alpha/make_ubuntu_sd.sh

  从github下载：

  wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.7.alpha/make_ubuntu_sd.sh

说明

您可以通过修改“make_sd_card.py”中的如下参数配置Atlas 200 DK的USB网卡IP与NIC网卡IP。

- “NETWORK_CARD_DEFAULT_IP”：NIC网卡的IP地址，默认值“192.168.0.2”。
- “USB_CARD_DEFAULT_IP”：USB网卡的IP地址，默认值“192.168.1.2”。

步骤6 执行制卡脚本。

1. 以root用户执行如下命令查找SD卡所在的USB设备名称。

   fdisk -l

   例如，SD卡所在USB设备名称为“/dev/sda”，可通过插拔SD卡的方式确定设备名称。

2. 运行SD制卡脚本“make_sd_card.py”。

   python3 make_sd_card.py local /dev/sda

   – “local”表示使用本地方式制作SD卡。
   – “/dev/sda”为SD卡所在的USB设备名称。

   如图4-1所示表示制卡成功。

图 4-1 SD 制卡回显信息示例

```
root@ascend-HP-ProDesk-600-G4-PCI-MT:/home/ascend/mksd# python3 make_sd_card.py local /dev/sda
Begin to make SD Card...
Please make sure you have installed dependency packages:
        apt-get install -y qemu-user-static binfmt-support gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
Please input Y: continue, other to install them:Y
Step: Start to make SD Card. It need some time, please wait...
Command:
bash /home/ascend/mksd/make_ubuntu_sd.sh   /dev/sda /home/ascend/mksd ubuntu-18.04.4-server-arm64.iso 192.168.0.2 192.168.1.2 >  /home/ascend/mksd/sd_card_making_log/make_ubuntu_sd.log
Make SD Card successfully!
```

说明

如果制卡失败，可以查看当前目录下的sd_card_making_log文件夹下的日志文件进行分析。

步骤7 制卡成功后，将SD卡从读卡器取出并插入Atlas 200 DK开发者板卡槽。

步骤8 上电Atlas 200 DK开发者板。

须知

- Atlas 200 DK制卡后，首次上电启动过程中会进行固件的升级，升级完成后会自动进行重启的操作，重启后再进行其他组件的安装。
- 首次启动Atlas 200 DK开发者板时不能断电，以免对Atlas 200 DK开发者板造成损害，再次上电需与上次下电时间保持2S以上的安全时间间隔。

Atlas 200 DK开发者板的上电操作及上电后指示灯的状态说明，请参见**5.2 上电Atlas 200 DK开发者板**。

----结束

异常处理

若上电后，开发者板无法正常启动（指示灯状态异常），可通过如下步骤查看相关日志：

步骤1 将Atlas 200 DK下电。

步骤2 从Atlas 200 DK中拔出SD卡，将SD卡插入读卡器，并将读卡器与Ubuntu服务器的USB端口连接。

步骤3 以root用户执行如下命令查找SD卡所在的USB设备及其分区信息。

fdisk -l

显示信息如下所示：

```
Disk /dev/sda: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1           2048 10487807 10485760    5G 83 Linux
/dev/sda2       10487808 12584959  2097152    1G 83 Linux
/dev/sda3       12584960 62333951 49748992 23.7G 83 Linux
```

步骤4 将SD卡的第一个分区挂载到Ubuntu服务器。

1. 使用root用户，创建一个空目录。

   例如：

   mkdir -p *\/home/sdinfo*

2. 将/dev/sda1挂载到/home/sdinfo目录。

   mount /dev/sda1 *\/home/sdinfo*

步骤5 进入/home/sdinfo目录，即进入Atlas 200 DK的文件系统，可在此目录的var/log/ascend_seclog路径下查看相关日志文件。

cd *\/home/sdinfo*

cd var/log/ascend_seclog/

可查看此路径的日志文件，日志文件描述如下所示：

- operation.log：操作日志，记录安装或升级等事件的结果信息，通过此日志可看出安装升级的最终结果是成功还是失败。

日志格式：事件类型+事件级别+用户ID+日期+发起端地址+访问文件名+操作指令+执行结果

- ascend_install.log：安装/升级详细日志，记录安装/升级的详细运维脚本日志，可通过此日志查看安装/升级的详细操作及状态信息。

  日志格式：组件+日期+日志级别+日志内容

- ascend_run_servers.log：Atlas 200 DK启动信息。

须知

- 若开发者无法解决问题，在**Ascend论坛**上发求助帖，并将以上日志信息作为帖子的附件上传，将会有华为工程师为您解答。
- 若Atlas 200 DK启动三次及以上后均失败，解决问题后，需要删除**var/log/ascend_seclog/**目录下的boot_fail_count文件，否则Atlas 200 DK无法正常启动。

----结束

4.1.1.3 无读卡器场景

介绍无读卡器场景下，通过跳线帽/杜邦线短接开发者板引脚的方式制作SD卡的操作。

硬件准备

1. 拆卸上盖，请参见**3.1 拆卸上盖**。
2. 请准备跳线帽或者杜邦线，然后参考**图4-2**短接开发者板上的16号与18号两个针脚。

须知

- 请在开发者板断电条件下执行此操作。开发者板的下电操作要求请参考**5.3 下电Atlas 200 DK开发者板**。
- 请仔细核对针脚位置，若针脚插错将会对Atlas 200 DK开发者板造成严重损坏。
- 面板上用白色字体标出了1、2、40三个针脚的位置，可仔细查看。

图 4-2 杜邦线安装示例图

3. 将Atlas 200 DK开发者板与Ubuntu服务器的USB连接。
4. 将Atlas 200 DK开发者板上电，详细上电操作请参考**5.2 上电Atlas 200 DK开发者板**。

软件包准备

获取开发者板驱动与运行包、Ubuntu操作系统镜像包，下载信息如**表4-2**所示：

表 4-2 软件包信息

包信息	包名	说明	下载地址
Ubuntu操作系统镜像包	ubuntu-18.04.xx-server-arm64.iso	开发者板操作系统镜像包。 Ubuntu操作系统的版本可以为：18.04.4、18.04.5。 **注意：为ARM版本的Server软件包。**	请从**华为开源镜像**或者**Ubuntu官网**下载此软件包。
开发者板驱动包与运行包	Ascend310-driver-*{software version}*-ubuntu18.04.aarch64-minirc.tar.gz	开发者板驱动包，包含OS外围软件、AI软件栈、维测相关软件及驱动。 制作SD卡时，会从driver包中获取Firmware信息，不需要单独获取Firmware组件包。	**获取链接** 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。

包信息	包名	说明	下载地址
	Ascend310-aicpu_kernels-*{software version}*-minirc.tar.gz	AI CPU算子包。	**获取链接** 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。
	Ascend-acllib-*{software version}*-ubuntu18.04.aarch64-minirc.run	AscendCL（Ascend Computing Language）库文件安装包，用来编译和运行AscendCL应用程序，包含模型加载及执行功能。 制卡时使用HwHiAiUser用户进行此包的安装，并会自动在HwHiAiUser用户的.bashrc文件中设置LD_LIBRARY_PATH的环境变量： **export LD_LIBRARY_PATH=/home/HwHiAiUser/Ascend/acllib/lib64**	**获取链接** 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。

须知

软件下载后请保持原命名。

操作步骤

步骤1 在Ubuntu服务器中执行如下命令安装qemu-user-static、binfmt-support、yaml与交叉编译器。

su - root

执行如下命令更新源：

apt-get update

执行如下命令安装相关python依赖：

pip3 install pyyaml

apt-get install qemu-user-static binfmt-support python3-yaml gcc-aarch64-linux-gnu g++-aarch64-linux-gnu

步骤2 在Ubuntu服务器中以root用户执行如下命令创建制卡工作目录。

mkdir /home/ascend/mksd

制卡目录可任意指定。

步骤3 将**软件包准备**获取的Ubuntu操作系统镜像包、开发者板所有驱动包上传到制卡工作目录中（例如：“/home/ascend/mksd”）。

步骤4 在制卡工作目录下（例如：“/home/ascend/mksd”）执行如下命令获取制卡脚本。

- 下载制卡入口脚本“make_sd_card.py”。

  从gitee下载：

  wget https://gitee.com/ascend/tools/raw/master/makesd/for_1.0.7.alpha/make_sd_card.py

  从github下载：

  wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.7.alpha/make_sd_card.py

- 下载制作SD卡操作系统的脚本“make_ubuntu_sd.sh”。

  从gitee下载：

  wget https://gitee.com/ascend/tools/raw/master/makesd/for_1.0.7.alpha/make_ubuntu_sd.sh

  从github下载：

  wget https://raw.githubusercontent.com/Ascend/tools/master/makesd/for_1.0.7.alpha/make_ubuntu_sd.sh

说明

您可以通过修改“make_sd_card.py”中的如下参数配置Atlas 200 DK的USB网卡IP与NIC网卡IP。

- “NETWORK_CARD_DEFAULT_IP”：NIC网卡的IP地址，默认值“192.168.0.2”。
- “USB_CARD_DEFAULT_IP”：USB网卡的IP地址，默认值“192.168.1.2”。

步骤5 执行制卡脚本。

1. 以root用户执行如下命令查找SD卡所在的USB设备名称。

   fdisk -l

   例如，SD卡所在USB设备名称为“/dev/sda”，可通过插拔SD卡的方式确定设备名称。

2. 运行SD制卡脚本“make_sd_card.py”。

   python3 make_sd_card.py local /dev/sda

   – “local”表示使用本地方式制作SD卡。
   – “/dev/sda”为SD卡所在的USB设备名称。

   如图4-3所示表示制卡成功。

图 4-3 SD 制卡回显信息示例

说明

如果制卡失败，可以查看当前目录下的sd_card_making_log文件夹下的日志文件进行分析。

步骤6 制卡成功后，将Atlas 200 DK开发者板下电，详细下电操作请参考**5.3 下电Atlas 200 DK开发者板**。

步骤7 拔掉跳线帽或者杜邦线。

步骤8 上电Atlas 200 DK开发者板。

须知

- Atlas 200 DK制卡后，首次上电启动过程中会进行固件的升级，升级完成后会自动进行重启的操作，重启后再进行其他组件的安装。
- 首次启动Atlas 200 DK开发者板时不能断电，以免对Atlas 200 DK开发者板造成损害，再次上电需与上次下电时间保持2S以上的安全时间间隔。

Atlas 200 DK开发者板的上电操作及上电后指示灯的状态说明，请参见**5.2 上电Atlas 200 DK开发者板**。

----结束

异常处理

若上电后，开发者板无法正常启动（例如，指示灯状态异常），可通过如下步骤查看相关日志：

步骤1 将Atlas 200 DK下电。

步骤2 使用跳线帽/杜邦线短接Atlas 200 DK的16与18号针脚，如**硬件准备**所示，将Atlas 200 DK作为USB设备。

步骤3 将Atlas 200 DK通过USB端口与Ubuntu服务器链接，并上电Atlas 200 DK。

步骤4 以root用户执行如下命令查找SD卡所在的USB设备及其分区信息。

fdisk -l

显示信息如下所示：

```
Disk /dev/sda: 29.7 GiB, 31914983424 bytes, 62333952 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1           2048 10487807 10485760    5G 83 Linux
/dev/sda2       10487808 12584959  2097152    1G 83 Linux
/dev/sda3       12584960 62333951 49748992 23.7G 83 Linux
```

步骤5 将SD卡的第一个分区挂载到Ubuntu服务器。

1. 使用root用户，创建一个空目录。

   例如：

mkdir -p */home/sdinfo*

2. 将/dev/sda1挂载到/home/sdinfo目录。

   mount /dev/sda1 */home/sdinfo*

步骤6 进入/home/sdinfo目录，即进入Atlas 200 DK的文件系统，可在此目录的var/log/ascend_seclog路径下查看相关日志文件。

cd */home/sdinfo*

cd var/log/ascend_seclog/

可查看此路径的日志文件，日志文件描述如下所示：

- operation.log：操作日志，记录安装或升级等事件的结果信息，通过此日志可看出安装升级的最终结果是成功还是失败。

  日志格式：事件类型+事件级别+用户ID+日期+发起端地址+访问文件名+操作指令+执行结果
- ascend_install.log：安装/升级详细日志，记录安装/升级的详细运维脚本日志，可通过此日志查看安装/升级的详细操作及状态信息。

  日志格式：组件+日期+日志级别+日志内容
- ascend_run_servers.log：Atlas 200 DK启动信息。

须知

- 若开发者无法解决问题，在**Ascend论坛**上发求助帖，并将以上日志信息作为帖子的附件上传，将会有华为工程师为您解答。
- 若Atlas 200 DK启动三次及以上后均失败，解决问题后，需要删除**var/log/ascend_seclog/**目录下的boot_fail_count文件，否则Atlas 200 DK无法正常启动。

----结束

4.1.2 连接 Atlas 200 DK 开发者板与 Ubuntu 服务器

场景描述

Atlas 200 DK开发者板支持通过USB端口或者网线与Ubuntu服务器进行连接，连接示例图如图4-4所示。

图 4-4 Atlas 200 DK 与 Mind Studio 连接图

Atlas 200 DK连接Ubuntu服务器有以下场景：

- 使用USB连接线通过USB端口与Ubuntu服务器直连，请参考**通过USB端口直连Ubuntu服务器**。

  此种场景下，Atlas 200 DK不方便接入网络，仅适用于与Ubuntu服务器进行通信。
- 使用网线将Atlas 200 DK接入路由器的网口，通过网络与Ubuntu服务器连接，请参考**使用网线通过路由器连接Ubuntu服务器（推荐）**。

  推荐使用此种方式，此种方式下，Atlas 200 DK可直接接入网络。
- 使用网线将Atlas 200 DK与Ubuntu服务器的网口直连，请参考**使用网线直连Ubuntu服务器**。

  此种场景下，Atlas 200 DK不方便接入网络，仅适用于与Ubuntu服务器进行通信。

说明

若制作SD卡时，您已修改Atlas 200 DK开发者板的IP地址与Ubuntu服务器的虚拟USB网卡的IP地址在同一网段，则以下步骤中修改Ubuntu服务器虚拟USB网卡IP地址的操作跳过。

通过 USB 端口直连 Ubuntu 服务器

Atlas 200 DK开发者板与Ubuntu服务器通过USB端口直连场景下，由于Atlas 200 DK的USB网卡的默认IP地址为192.168.1.2，所以需要修改Ubuntu服务器的USB虚拟网卡的IP地址为192.168.1.x（x取值范围为0~1，3~254）。

下面提供通过脚本配置Ubuntu服务器的USB虚拟网卡的IP地址与手工配置Ubuntu服务器USB虚拟网卡IP地址的两种方法。

须知

若您的Ubuntu服务器是通过Windows主机上的虚拟机进行安装的场景，需要参考**5.10 如何安装windows的USB网卡驱动**在Windows主机上安装USB虚拟网卡驱动，否则Ubuntu服务器无法识别Atlas 200 DK的USB虚拟网卡。

假设您已经通过USB连接线将Atlas 200 DK与Ubuntu服务器的USB端口进行连接，操作步骤如下：

- 通过脚本配置

  a. 从**https://gitee.com/ascend/tools/tree/master/configure_usb_ethernet/for_1.7x.0.0**或者**https://github.com/Huawei-Ascend/tools/tree/master/configure_usb_ethernet/for_1.7x.0.0**下载configure_usb_ethernet.sh到Ubuntu服务器的任一目录，例如*/home/ascend/config_usb_ip/*。

  须知

  通过脚本配置仅针对首次配置USB网卡对应IP地址的场景。USB网卡IP已经配置，若需要修改其IP地址，请参考 **手工配置**修改USB网卡的IP地址。

  b. 以root用户进入配置USB网卡IP地址的脚本所在目录，例如*/home/ascend/config_usb_ip*。

c. 执行如下命令进行USB网卡IP地址的配置。

bash configure_usb_ethernet.sh -s *ip_address*

以指定的IP地址配置Ubuntu服务器中USB网卡的静态IP地址，如果直接执行**bash configure_usb_ethernet.sh**，则以默认IP地址“192.168.1.166”进行配置。

- 如果存在多个USB网卡，则首先执行**ifconfig**命令查询USB网卡名称（若系统中有多个USB网卡，可以通过拔插开发者板进行判定，Ubuntu服务器会将Atlas 200 DK开发者板识别为虚拟USB网卡），再执行如下命令配置指定网卡的IP地址。

  bash configure_usb_ethernet.sh -s *usb_nic_name ip_address*

  usb_nic_name：USB网卡名称。

  ip_address：配置的IP地址。

  命令示例，配置Ubuntu服务器的USB网卡IP为192.168.1.223：

  bash configure_usb_ethernet.sh -s enp0s20f0u8 192.168.1.223

配置完成后，可执行**ifconfig**命令查看IP是否生效。

- 手工配置

a. 以普通用户登录Ubuntu服务器，执行如下命令切换到root用户。

```
su - root
```

b. 获取USB网卡名

```
ifconfig -a
```

若系统中有多个USB网卡，可以通过拔插开发者板进行判定。

c. 在“/etc/netplan/01-netcfg.yaml”文件中添加USB网卡的静态IP。

执行如下命令打开网络配置文件：

```
vi /etc/netplan/01-netcfg.yaml
```

在ethernets层下增加USB网卡网络配置，例如USB网卡名为*enp0s20f0u4*，配置静态IP为*192.168.1.223*，配置方法如下：

```
ethernets:
  ...
  enp0s20f0u4:
    dhcp4: no
    addresses: [192.168.1.223/24]
    gateway4: 192.168.0.1
    nameservers:
        addresses: [255.255.0.0]
```

输入**:wq**保存退出。

d. 执行如下命令重启网络服务。

netplan apply

重启后执行ifconfig命令，可查看USB网卡*enp0s20f0u4*的IP地址是否生效。

使用网线通过路由器连接 Ubuntu 服务器（推荐）

Atlas 200 DK通过网线接入路由器的场景下，需要开启路由器的DHCP功能，然后使用DHCP的方式自动分配Atlas 200 DK的IP地址，所以需要修改Atlas 200 DK的NIC网卡的IP地址获取方式为DHCP。此时，需要同时使用USB方式连接Ubuntu服务器，然后通过Ubuntu服务器 SSH登录到Atlas 200 DK进行NIC网卡的IP获取方式的修改。

假如您已经将Atlas 200 DK通过网线接入路由器，并开启了路由器的DHCP功能，后续操作步骤如下：

1. 使用USB连接线将Atlas 200 DK与Ubuntu服务器连接，并参考**通过USB端口直连Ubuntu服务器**配置Ubuntu服务器的USB网卡的IP地址，使其可与Atlas 200 DK通信。
2. 配置Atlas 200 DK的NIC网卡IP地址获取方式为DHCP。
   a. 在Ubuntu服务器中，以HwHiAiUser用户ssh登录到Atlas 200 DK开发者板。

      ssh HwHiAiUser@192.168.1.2

      Atlas 200 DK的USB网卡的默认IP地址为192.168.1.2。

      用户HwHiAiUser缺省登录密码为“**Mind@123**”。
   b. 切换到root用户，并打开网络配置文件。

      su - root

      vi /etc/netplan/01-netcfg.yaml
   c. 修改eth0网卡的IP地址获取方式为DHCP。

      把eth0的配置修改为如下内容：

      ```
      eth0:
        dhcp4: true
        addresses: []
        optional: true
      ```
   d. 保存退出。

      :wq
3. 执行如下命令重启网络服务。

   netplan apply

   至此，您可以在Atlas 200 DK开发者板上访问网络了。
4. 执行ifconfig命令获取eth0网卡的IP地址，您可以使用此IP地址与Ubuntu服务器通信，当然，您也可以继续使用USB网卡的IP地址与Ubuntu服务器通信。

使用网线直连 Ubuntu 服务器

Atlas 200 DK开发者板与Ubuntu服务器通过网线直连的场景下，需要修改Ubuntu服务器的IP地址为192.168.0.x（x取值范围为0~1，3~254）。

说明

- Atlas 200 DK开发者板的NIC网卡的默认地址为192.168.0.2，子网掩码为24位。
- Atlas 200 DK网口接入网线后，黄色的ACT灯闪烁，代表有数据传输。当是千兆以太网时，绿色的LINK灯会亮，当是百兆/10兆的以太网时，LINK灯正常不会亮。

配置方法如下：

1. 以普通用户登录Ubuntu服务器，执行如下命令切换到root用户。

   ```
   su - root
   ```
2. 配置一个虚拟网卡IP，用于与Atlas 200 DK进行通信。

   例如，配置*eth0:1*的虚拟静态IP地址。

   ```
   ifconfig eth0:1 192.168.0.223 netmask 255.255.0.0 up
   ```

后续处理

Atlas 200 DK开发者板连接Ubuntu服务器后，您可以根据Atlas 200 DK开发者板 LED指示灯的状态决定是否重启Atlas 200 DK开发者板上的操作系统或对Atlas 200 DK开发者板执行断电操作，指示灯状态说明请参考**表5-2**

须知

切忌不要随意重启或断电，尤其是Atlas 200 DK开发者板正在进行升级。

4.2 安装开发环境

4.2.1 获取软件包

软件安装前，请首先获取以下软件包，*{version}*表示软件包版本号，版本号需要保持一致。

表 4-3 软件包介绍

名称	软件包	获取链接	功能描述
开发套件包	Ascend-Toolkit-*{version}*-x86_64-linux_gcc7.3.0.run	获取链接 1. “产品系列”请选择“AI开发者套件”。 2. “产品型号”请选择“Atlas 200 DK开发者套件”。 3. “固件与驱动版本”：选择“1.0.7.alpha”。	• 主要用于用户开发应用、自定义算子和模型转换。开发套件包包含开发应用程序所需的库文件、开发辅助工具（如ATC模型转换工具）。 • 两种架构统一安装：开发环境为x86架构，但运行环境是arm架构，所以再安装一个arm64的开发套件包，配置交叉编译环境。
	Ascend-Toolkit-*{version}*-arm64-linux_gcc7.3.0.run		

4.2.2 配置 Ubuntu X86 系统

检查 root 用户的 umask

1. 以root用户登录安装环境。
2. 检查root用户的umask值。

```
umask
```

3. 如果umask不等于0022，请执行如下操作配置，在该文件的最后一行添加umask 0022后保存。

```
vi ~/.bashrc
source ~/.bashrc
```

创建安装用户

开发套件包需要以非root用户下进行安装，所以需要创建非root用户，请以root用户执行以下命令。

1. 创建非root用户。
```
useradd -d /home/username -m username
```
*username*由用户自定义。
2. 设置非root用户密码。
```
passwd username
```

说明

密码有效期为90天，您可以在/etc/login.defs文件中修改有效期的天数，或者通过chage命令来设置用户的有效期，详情请参见**5.11 设置用户有效期**。

配置安装用户权限

当用户使用非root用户安装时，需要配置安装用户权限。

开发套件包安装前需要下载相关依赖软件，下载依赖软件需要使用**sudo apt-get**权限，请以root用户执行如下操作。

1. 打开“/etc/sudoers”文件：
```
chmod u+w /etc/sudoers
vi /etc/sudoers
```
2. 在该文件“# User privilege specification”下面增加如下内容：
```
username ALL=(ALL:ALL)   NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5, /usr/bin/unzip
```
“username”为执行安装脚本的普通用户名。

说明

请确保“/etc/sudoers”文件的最后一行为“#includedir /etc/sudoers.d”，如果没有该信息，请手动添加。

3. 添加完成后，执行**:wq!**保存文件。
4. 执行以下命令取消“/etc/sudoers”文件的写权限：
```
chmod u-w /etc/sudoers
```

检查源

开发套件包安装过程需要下载相关依赖，请确保安装环境能够连接网络。

请在root用户下执行如下命令检查源是否可用。

```
apt-get update
```

如果命令执行报错或者后续安装依赖时等待时间过长甚至报错，则检查网络是否连接或者把“/etc/apt/sources.list”文件中的源更换为可用的源或使用镜像源。如需配置网络代理，请参见**5.13 配置系统网络代理**。

安装依赖

说明

- 如果用户使用root用户执行以下步骤安装Python及其依赖，注意命令中若有**sudo**，需要删除。
- 如果用户使用非root用户安装Python及其依赖，需要使用**su - username**命令切换到非root用户执行以下步骤。

步骤1 检查系统是否安装Python依赖以及gcc等软件。

分别使用如下命令检查是否安装gcc，make以及Python依赖软件等。

```
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l openssl| grep openssl| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l unzip| grep unzip| grep ii
dpkg -l pciutils| grep pciutils| grep ii
dpkg -l net-tools| grep net-tools| grep ii
dpkg -l libblas-dev| grep libblas-dev| grep ii
dpkg -l gfortran| grep gfortran| grep ii
dpkg -l libblas3| grep libblas3| grep ii
dpkg -l libopenblas-dev| grep libopenblas-dev| grep ii
```

若分别返回如下信息则说明已经安装，进入下一步。

```
gcc (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
g++ (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
GNU Make 4.1
cmake version 3.10.2
zlib1g:arm64   1:1.2.11.dfsg-0ubuntu2 arm64        compression library - runtime
zlib1g-dev:arm64 1:1.2.11.dfsg-0ubuntu2 arm64        compression library - development
libsqlite3-dev:arm64 3.22.0-1ubuntu0.3 arm64        SQLite 3 development files
openssl  1.1.1-1ubuntu2.1~18.04.6 arm64   Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:arm64 1.1.1-1ubuntu2.1~18.04.6 arm64     Secure Sockets Layer toolkit - development files
libffi-dev:arm64 3.2.1-8      arm64        Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 amd64        De-archiver for .zip files
pciutils       1:3.5.2-1ubuntu1 arm64        Linux PCI Utilities
net-tools      1.60+git20161116.90da8a0-1ubuntu1 arm64        NET-3 networking toolkit
libblas-dev:arm64 3.7.1-4ubuntu1 arm64        Basic Linear Algebra Subroutines 3, static library
gfortran       4:7.4.0-1ubuntu2.3 arm64        GNU Fortran 95 compiler
libblas3:arm64 3.7.1-4ubuntu1 arm64     Basic Linear Algebra Reference implementations, shared library
libopenblas-dev:arm64 0.2.20+ds-4  arm64    Optimized BLAS (linear algebra) library (development files)
```

否则请执行如下安装命令（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）：

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev libsqlite3-dev openssl libssl-dev libffi-dev
unzip pciutils net-tools libblas-dev gfortran libblas3 libopenblas-dev
```

步骤2 检查系统是否安装python开发环境。

开发套件包依赖python环境，分别使用命令**python3.7.5 --version**、**pip3.7.5 --version**检查是否已经安装，如果返回如下信息则说明已经安装，进入下一步。

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

否则请根据如下方式安装python3.7.5。

1. 使用wget下载python3.7.5源码包，可以下载到安装环境的任意目录，命令为：
   ```
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

2. 进入下载后的目录，解压源码包，命令为：
```
tar -zxvf Python-3.7.5.tgz
```
3. 进入解压后的文件夹，执行配置、编译和安装命令：
```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-shared
make
sudo make install
```
其中“--prefix”参数用于指定python安装路径，用户根据实际情况进行修改。“--enable-shared”参数用于编译出libpython3.7m.so.1.0动态库。

本手册以--prefix=/usr/local/python3.7.5路径为例进行说明。执行配置、编译和安装命令后，安装包在/usr/local/python3.7.5路径，libpython3.7m.so.1.0动态库在/usr/local/python3.7.5/lib/libpython3.7m.so.1.0路径。
4. 执行如下命令设置软链接：
```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
```
5. 设置python3.7.5环境变量。
   a. 以安装用户在任意目录下执行**vi ~/.bashrc**命令，打开**.bashrc**文件，在文件最后一行后面添加如下内容。
   ```
   #用于设置python3.7.5库文件路径
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   #如果用户环境存在多个python3版本，则指定使用python3.7.5版本
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```
   b. 执行**:wq!**命令保存文件并退出。
   c. 执行**source ~/.bashrc**命令使其立即生效。
6. 安装完成之后，执行如下命令查看安装版本，如果返回相关版本信息，则说明安装成功。
```
python3.7.5 --version
pip3.7.5  --version
python3.7 --version
pip3.7  --version
```

步骤3 安装 Python3开发环境。

安装前请先使用**pip3.7.5 list**命令检查是否安装相关依赖，若已经安装，则请跳过该步骤；若未安装，则安装命令如下（如果只有部分软件未安装，则如下命令修改为只安装还未安装的软件即可）。其中开发套件包中算子比对工具依赖：protobuf、scipy；profiling工具依赖：protobuf、grpcio、grpcio-tools、requests。

如果使用非root用户安装Python及其依赖，用户需要在本步骤中的每句命令结尾加上**--user**，保证安装的正常进行。命令示例为：**pip3.7.5 install attrs --user**

```
pip3.7.5 install attrs
pip3.7.5 install psutil
pip3.7.5 install decorator
pip3.7.5 install numpy
pip3.7.5 install protobuf==3.11.3
pip3.7.5 install scipy
pip3.7.5 install sympy
pip3.7.5 install cffi
pip3.7.5 install grpcio
pip3.7.5 install grpcio-tools
pip3.7.5 install requests
```

如果执行上述命令时无法连接网络，且提示“Could not find a version that satisfies the requirement xxx”，请参见**6.3 使用pip3.7.5 install软件时提示" Could not find a version that satisfies the requirement xxx"**解决。

----结束

4.2.3 安装开发套件包

前提条件

- 请参见4.2.2 配置Ubuntu X86系统完成安装前准备。
- 已获取开发套件包Ascend-Toolkit-*{version}*-x86_64-linux_gcc7.3.0.run和Ascend-Toolkit-*{version}*-arm64-linux_gcc7.3.0.run。

安装步骤

步骤1 以软件包的安装用户（与安装依赖中用户保持一致）登录安装环境。

步骤2 上传4.2.1获取软件包所获取开发套件包到系统任意路径（如“/home”），并进入套件包所在路径。

步骤3 执行如下命令增加可执行权限并校验一致性和完整性。

chmod +x *.run

./*.run --check

*表示开发套件包名，请用户根据实际替换。

步骤4 执行以下命令安装软件。

- 若用户指定安装路径：

  ./*.run --install --install-path=*{path}*

  其中***{path}***为指定的安装路径，请用户自行替换。
- 若用户不指定安装路径：

  ./*.run --install。

安装路径详细介绍请参见安装路径说明。

说明

- 更多安装参数请参见5.14 参数说明。

----结束

安装路径说明

- 如果用户想指定安装路径，则需要先创建安装路径。以安装路径“/home/work”为例， 用户先执行**mkdir -p /home/work**命令创建安装路径，再指定该路径进行软件安装。
- 如果用户未指定安装路径，则软件会安装到默认路径下，默认路径请参见表 软件包安装路径。

表 4-4 软件包安装路径

项目	路径
软件包默认安装路径	“${HOME}/Ascend/${package_name}/{version}/{arch-linux-gccversion}”

项目	路径
安装详细日志路径	“${install_path}/${package_name}/{version}/{arch-linux-gccversion}/ascend_install.log”
安装后软件包版本、CPU架构、GCC版本和安装路径等信息的记录路径	“${install_path}/${package_name}/{version}/{arch-linux-gccversion}/ascend-${package_name}_install.info”

表 软件包安装路径中各变量的含义如表 变量含义所示：

表 4-5 变量含义

参数	含义
${package_name}	软件包目录，以软件包的名称命名。
{version}	版本目录，以软件包的版本号命名。 **说明** 支持安装多个版本，以版本号区分。
{arch-linux_gccversion}	形态目录，以软件包的CPU架构、Linux分支和GCC版本的组合命名。 **说明** 支持安装多个形态版本。
${HOME}	当前用户家目录。
${install_path}	软件包的安装路径。

4.2.4 部署 Media 模块

若开发者需要使用Atlas 200 DK外接摄像头采集的数据作为AI应用数据源，需要额外在开发环境中部署应用程序编译所依赖的头文件与库文件。

操作步骤如下：

步骤1 将Ascend310-driver-*{software version}*-ubuntu18.04.aarch64-minirc.tar.gz以开发环境安装用户上传到开发环境任一目录。

例如：开发环境安装用户为HwHiAiUser，上传到/home/HwHiAiUser/software目录下。

步骤2 以安装用户解压缩driver包。

tar zxvf Ascend310-driver-*{software version}*-ubuntu18.04.aarch64-minirc.tar.gz

媒体控制相关头文件与库文件在解压缩后的driver目录下，如下所示：

- driver/peripheral_api.h: 媒体模块头文件，详细的接口使用说明可参见《**Media API参考**》。

须知

- Media的API接口是基于C语言的接口。
- 若使用的RASPBERRY PI V2.1型号的摄像头，摄像头支持设置的帧率范围为：1~20fps。
- 若使用的RASPBERRY PI V1.3型号的摄像头，摄像头支持设置的帧率范围为：1~15fps。

- driver/libmedia_mini.so：媒体模块库文件。

用户在进行Media模块相关应用程序开发时，需要引用此目录下的头文件与库文件。

----结束

4.2.5 配置交叉编译环境

若开发环境与运行环境的架构不同，则需要在开发环境安装交叉编译工具，具体如表4-6所示：

表 4-6 安装交叉编译工具

开发环境架构	运行环境架构	编译环境配置
x86_64	aarch64	请使用软件包的安装用户，在开发环境执行**aarch64-linux-gnu-g++ --version**命令检查是否安装，若已经安装则可以忽略。 安装命令示例如下（以下命令仅为示例，请用户根据实际情况替换）： **sudo apt-get install g++-aarch64-linux-gnu**

4.2.6 安装 MindStudio

MindStudio是一套基于华为自研昇腾AI处理器开发的AI全栈开发工具平台，包括网络模型移植、应用开发、推理运行及自定义算子开发等功能。通过MindStudio能够进行工程管理、编译、调试、运行、性能分析等全流程开发，提高开发效率。详情可参考《**MindStudio 用户指南**》。

4.3 安装开发环境与运行环境

在X86架构的Ubuntu服务器上，完成ADKInstaller工具对开发环境的安装以及Atlas 200 DK SD卡启动盘的制作。如果不能使用msInstaller工具安装环境，可手动安装。

环境搭建流程

根据下图可以了解ADKInstaller工具的功能以及工具进行环境搭建的总体流程。

图 4-5 环境搭建流程图

安装前准备

已完成Ubuntu X86系统依赖的安装。

获取并启动 ADKInstaller 工具

步骤1 获取ADKInstaller。

1. 以普通用户登录Ubuntu系统。
2. 执行以下命令下载ADKInstaller工具压缩包到系统任意目录，如“/home/HwHiAiUser”。

   wget https://obs-book.obs.cn-east-2.myhuaweicloud.com/temp/ADKInstaller.0.1.1.tar.gz
3. 解压软件压缩包并进入解压文件目录下。

   tar -xzvf ADKInstaller.0.1.1.tar.gz

   cd ADKInstaller
4. 切换至root用户下，执行以下命令为当前用户开启sudo权限。

   su root

   ./add_sudo.sh *username*

   exit

   说明

   - username需要替换为安装Atlas200DK的开发环境的普通用户的用户名，如ascend。
   - 这里需要临时给当前用户开启sudo权限，当安装完环境后可以在root用户下执行 **./del_sudo.sh** *username* 删除当前用户的sudo权限。

步骤2 启动ADKInstaller。

以Ubuntu系统中普通用户执行以下命令，启动ADKInstaller。

./ADKInstaller

启动ADKInstaller后如图4-6所示。

- User：启动ADKInstaller工具的普通用户。如果使用root用户则无法启动软件。
- Password：启动ADKInstaller工具的用户的密码。
- language：选择语言，默认为英文。

图 4-6 登录界面

启动成功后界面如图4-7所示。

图 4-7 软件主页面

----结束

安装运行环境

步骤1 制作SD卡。

说明

- SD卡用于制作Atlas 200 DK的系统启动盘。
- 如果已经拥有一张制作好的SD卡（Atlas 200 DK使用此卡可以正常启动），则单击“Skip”跳过该步骤，直接在后续步骤中对Atlas 200 DK进行升级。

1. 在“STEP 01 Make SD Card”界面，勾选“I accept the terms and conditions of the license agreement”，按界面提示单击“OK”。
2. 如图4-8所示，设置如下参数信息。
   - Host components：选择安装源，制作SD卡时需要安装相关的操作系统依赖（安装在Ubuntu X86服务器）。
   - Target components：选择安装源，用于安装SD卡上的操作系统依赖。
   - Ubuntu components：SD卡上安装的Ubuntu iso。
   - Download Path：文件在Ubuntu X86服务器的下载路径。
   - SD INFO：鼠标悬停时可以查看当前需要制卡的开发者板的USB网卡IP和物理网卡IP。未选择时默认USB网卡IP地址为192.168.158.2，默认物理网卡IP地址为192.168.157.2。

   说明

   - 若当前Ubuntu服务器在使用ADKInstaller工具之前未执行过**apt-get update**命令，建议先手动在命令行下执行**apt-get update**命令，否则apt依赖可能会安装不成功。请参见：**https://gitee.com/lovingascend/ADKInstaller/blob/master/FAQ.md**
   - 国外用户建议选择默认源，即：use the default apt-get source
   - 国内用户建议选择清华源，即：use the https://mirrors.tuna.tsinghua.edu.cn/ubuntu/ apt-get source

图 4-8 制卡

3. 单击界面右下方的“Make SD”，组件下载完成后，弹出图4-9所示界面。

图 4-9 制卡选择

- usb IP：USB网卡IP。
- eth0 IP：物理网卡IP。

4. 将带有SD卡的读卡器插入PC，单击“Refresh”，选择SD卡对应的名称。
5. 选择制卡后Atlas 200 DK的USB网卡IP（默认为192.168.158.2）、物理网卡IP（eth0 IP）。

   此时物理网卡IP会与USB网卡IP绑定，后续物理网卡IP地址会随USB网卡IP地址自动更新。
6. 单击“OK”，开始制作SD卡。
7. 制卡成功后，弹出制卡成功提示框，如**图4-10**所示，单击“OK”。

图 4-10 制卡成功

步骤2 连接开发者板。

1. 将制作完成的SD卡插入Atlas 200 DK。
2. 用Type-C线连接PC、USB端口和Atlas 200 DK的Type-C接口。
3. 上电Atlas 200 DK，等待约15分钟后，单板4个LED灯全亮，说明单板正常启动。
4. 选择界面左侧的“STEP 02 Connect Atlas 200 DK”。

图 4-11 连接 Atlas 200 DK

5. 单击“Refresh”刷新，会自动选择虚拟网卡。如果有多块开发板请选择对应的虚拟网卡。

选择对应的Atlas 200 DK单板IP。点击“Connect”连接单板。

如果对开发板连接有疑问，可以点击“tips”按钮，查看连接的详细说明。

若出现如图4-12所示界面，说明开发者板连接成功。

图 4-12 连接成功

6. 单击OK，完成Atlas 200 DK配置。

----结束

安装开发环境

步骤1 安装MindStudio。

1. 选择“STEP 03 Setup Mind Studio”，进入MindStudio安装界面。
2. 如图4-13，在填写如下参数信息后，单击右下方的“**Setup**”，工具自动下载并安装环境依赖。
   - 勾选“I accept the terms and conditions of the license agreement”。
   - apt-get install：安装Ubuntu系统相关依赖。
   - pip install：安装Python2相关依赖。
   - pip3 install：安装Python3相关依赖。
   - download ：下载软件包版本。
   - Download Path：依赖下载路径。

图 4-13 MindStudio 安装准备

3. 依赖安装完成后，提示开始安装MindStudio。如图4-14所示。

图 4-14 MindStudio 开始安装弹窗

4. 选择“OK”，开始安装MindStudio。
   安装完成后，提示如图4-15所示信息，单击“OK”，启动MindStudio。

图 4-15 MindStudio 安装

5. MindStudio启动时弹出如图4-16所示提示框，单击“OK”，完成配置。
   - Ascend Toolkit Path：选择MindStudio安装用户所安装的开发套件包的所在路径。

图 4-16 导入配置

6. 进入MindStudio启动脚本所在路径（${HOME}/MindStudio-ubuntu/bin），执行以下命令启动MindStudio，如图4-17所示。

   ./MindStudio.sh

图 4-17 MindStudio 启动

步骤2 后续任务。

通过MindStudio进行应用开发、模型转换、自定义算子开发。详情请参考《**MindStudio 用户指南**》。

----结束

5 常用操作

5.1 升级与卸载
5.2 上电Atlas 200 DK开发者板
5.3 下电Atlas 200 DK开发者板
5.4 通过串口连接Atlas 200 DK
5.5 查看开发者板系统软件的版本
5.6 查看开发者板硬件底板的版本
5.7 查看Atlas 200 AI加速模块的版本
5.8 如何查看开发者板的firmware版本
5.9 如何查看摄像头所属Channel
5.10 如何安装windows的USB网卡驱动
5.11 设置用户有效期
5.12 安装3.5.2版本cmake
5.13 配置系统网络代理
5.14 参数说明

5.1 升级与卸载

5.1.1 升级 Atlas 200 DK

如果您需要升级Atlas 200 DK的运行软件，可参考本章节进行操作。

获取软件包

升级软件包信息**表5-1**所示。

表 5-1 软件包信息

包信息	包名	说明	升级操作	获取链接
Driver	Ascend310-driver-*{software version}*-ubuntu18.04.aarch64-minirc.tar.gz	开发者板驱动包，包含OS外围软件、AI软件栈、维测相关软件及驱动。	**升级Driver包**	可点击**获取**
Firmware	Ascend310-firmware-*{software version}*-minirc.run	固件包。	**升级Firmware**	可点击**获取**
AI CPU	Ascend310-aicpu_kernels-*{software version}*-minirc.tar.gz	AI CPU算子包。	**升级AI CPU**	可点击**获取**
ACLlib	Ascend-acllib-*{software version}*-ubuntu18.04.aarch64-minirc.run	AscendCL（Ascend Computing Language）库文件安装包，用来编译和运行AscendCL应用程序，包含模型加载及执行功能。	**升级ACLlib**	可点击**获取**

升级 Driver 包

步骤1 将Driver升级包“Ascend310-driver-{software version}-ubuntu18.04.aarch64-minirc.tar.gz”拷贝到开发者板的“/opt/mini”目录下。

下面以将升级包从Ubuntu服务器（开发环境）拷贝到Atlas 200 DK为例进行操作介绍：

1. 进入Ubuntu服务器中存放Driver升级包的目录，并在当前目录下以HwHiAiUser用户ssh登录到开发者板，然后切换到root用户。

   ssh HwHiAiUser@*192.168.1.2*

   su - root

   说明

   若SSH方式登录Atlas 200 DK开发者板时无法建立信任关系，请参考**6.5 Ubuntu服务器连接开发者板无法建立信任关系**。

2. 进入Atlas 200 DK的/opt/mini目录，并拷贝Driver升级包。

   cd /opt/mini

   scp *username*@192.168.1.223:*/home/ascend/software*/Ascend310-driver-{software version}-ubuntu18.04.aarch64-minirc.tar.gz .

说明

- *username*为Ubuntu服务器中上传升级包的用户名。
- *192.168.1.223*为Ubuntu服务器与Atlas 200 DK在同一网段的IP地址。
- */home/ascend/software*为Ubuntu服务器中存放Driver升级包的路径。

步骤2 在/opt/mini目录下执行如下命令，解压缩Driver包，获取Driver包的升级脚本“minirc_install_phase1.sh”。

tar --no-same-owner -zxf Ascend310-driver-{software version}-ubuntu18.04.aarch64-minirc.tar.gz --strip-components 2 driver/scripts/minirc_install_phase1.sh

执行完成后，解压缩获得的“minirc_install_phase1.sh”脚本会替换目录下的原有脚本。

步骤3 执行以下命令进行Driver包升级前准备。

./minirc_install_phase1.sh

回显如图5-1所示。

图 5-1 执行升级脚本

```
  inflating: /var/mini/install_cache/mini_developerkit/lpm3.img
  inflating: /var/mini/install_cache/mini_developerkit/libmatrixdaemon.so
  inflating: /var/mini/install_cache/mini_developerkit/libcce_aicpudev_alg.a
  inflating: /var/mini/install_cache/mini_developerkit/libdrvupgrade.so
  inflating: /var/mini/install_cache/mini_developerkit/libmmpa.so
  inflating: /var/mini/install_cache/mini_developerkit/tsch_fw.asm
  inflating: /var/mini/install_cache/mini_developerkit/ide_daemon_client_cert.pem
  inflating: /var/mini/install_cache/mini_developerkit/libdrvdevmm.so
  inflating: /var/mini/install_cache/mini_developerkit/libdevmm.so
  inflating: /var/mini/install_cache/buildinfo.txt
  inflating: /var/mini/install_cache/check_sha.sh
2019-01-16,14:01:1547648536  extrack package succ
writing files to sdcard, please waiting several minutes ...
files check pass!
2019-01-16,14:01:1547648561  the first phase of installation finish, please reboot
root@davinci-mini:/opt/mini#
```

步骤4 执行重启命令，进行开发者板的重启，从而完成Driver包的升级。

reboot

须知

升级过程中请勿将Atlas 200 DK开发者板断电，升级时间15分钟左右。

----结束

升级 Firmware

步骤1 将Firmware升级包“Ascend310-firmware-{software version}-minirc.run”拷贝到开发者板的任一目录下，例如拷贝到/home/HwHiAiUser/software。

下面以将升级包从Ubuntu服务器（开发环境）拷贝到Atlas 200 DK为例进行操作介绍：

1. 进入Ubuntu服务器中存放Firmware升级包的目录，并在当前目录下以HwHiAiUser用户ssh登录到开发者板。

   ssh HwHiAiUser@*192.168.1.2*

   说明

   若SSH方式登录Atlas 200 DK开发者板时无法建立信任关系，请参考**6.5 Ubuntu服务器连接开发者板无法建立信任关系**。

2. 进入Atlas 200 DK中需要存放升级包的路径，并拷贝Firmware升级包。

   cd /home/HwHiAiUser/software

   scp *username*@192.168.1.223:*/home/ascend/software*/Ascend310-firmware-{software version}-minirc.run .

   说明

   - *username*为Ubuntu服务器中上传升级包的用户名。
   - *192.168.1.223*为Ubuntu服务器与Atlas 200 DK在同一网段的IP地址。
   - */home/ascend/software*为Ubuntu服务器中存放Firmware升级包的路径。

步骤2 切换到root用户，执行Firmware包的升级。

su root

chmod +x Ascend310-firmware-{software version}-minirc.run

./Ascend310-firmware-{software version}-minirc.run --upgrade

执行完此命令后，firmware相关文件存储在/usr/local/Ascend/firmware目录下。

步骤3 执行重启命令，进行开发者板的重启，从而完成Firmware包的升级。

reboot

须知

升级过程中请勿将Atlas 200 DK开发者板断电，升级时间15分钟左右。

----结束

升级 AI CPU

步骤1 将AI CPU升级包“Ascend310-aicpu_kernels-{software version}-minirc.tar.gz”拷贝到开发者板的任一目录下，例如拷贝到/home/HwHiAiUser/software。

下面以将升级包从Ubuntu服务器（开发环境）拷贝到Atlas 200 DK为例进行操作介绍：

1. 进入Ubuntu服务器中存放AI CPU升级包的目录，并在当前目录下以HwHiAiUser用户ssh登录到开发者板。

   ssh HwHiAiUser@*192.168.1.2*

   说明

   若SSH方式登录Atlas 200 DK开发者板时无法建立信任关系，请参考**6.5 Ubuntu服务器连接开发者板无法建立信任关系**。

2. 进入Atlas 200 DK中需要存放升级包的路径，并拷贝AI CPU升级包。

   cd /home/HwHiAiUser/software

   scp *username*@192.168.1.223:*/home/ascend/software/*Ascend310-aicpu_kernels-{software version}-minirc.tar.gz .

   说明

   - *username*为Ubuntu服务器中上传升级包的用户名。
   - *192.168.1.223*为Ubuntu服务器与Atlas 200 DK在同一网段的IP地址。
   - */home/ascend/software*为Ubuntu服务器中存放AI CPU升级包的路径。

步骤2 执行以下命令解压缩AI CPU升级包。

tar zxvf Ascend310-aicpu_kernels-{software version}-minirc.tar.gz

步骤3 进入AI CPU的升级脚本所在目录，并切换到root用户执行AI CPU的升级。

cd aicpu_kernels_device

su root

./scripts/install.sh --run

----结束

升级 ACLlib

步骤1 将ACLLib升级包“Ascend-acllib-{software version}-ubuntu18.04.aarch64.run”拷贝到开发者板的任一目录下，例如拷贝到/home/HwHiAiUser/software。

下面以将升级包从Ubuntu服务器（开发环境）拷贝到Atlas 200 DK为例进行操作介绍：

1. 进入Ubuntu服务器中存放ACLlib升级包的目录，并在当前目录下以HwHiAiUser用户ssh登录到开发者板。

   ssh HwHiAiUser@*192.168.1.2*

   说明

   若SSH方式登录Atlas 200 DK开发者板时无法建立信任关系，请参考**6.5 Ubuntu服务器连接开发者板无法建立信任关系**。

2. 进入Atlas 200 DK中需要存放升级包的路径，并拷贝ACLlib升级包。

   cd /home/HwHiAiUser/software

   scp *username*@192.168.1.223:*/home/ascend/software*/Ascend-acllib-*{software version}*-ubuntu18.04.aarch64-minirc.run .

   说明

   - *username*为Ubuntu服务器中上传升级包的用户名。
   - *192.168.1.223*为Ubuntu服务器与Atlas 200 DK在同一网段的IP地址。
   - */home/ascend/software*为Ubuntu服务器中存放ACLlib升级包的路径。

步骤2 执行以下命令进行ACLlib包的升级。

chmod +x Ascend-acllib-*{software version}*-ubuntu18.04.aarch64-minirc.run

./Ascend-acllib-*{software version}*-ubuntu18.04.aarch64-minirc.run --upgrade

----结束

升级完成后，您可以参考**5.5 查看开发者板系统软件的版本**查看各组件的版本号。

5.1.2 升级开发套件包

本章节指导用户升级开发套件包（Ascend-Toolkit-*{version}*-*{aarch}*-linux_*{gccversion}*.run）。升级软件包后，不会影响正常业务。

开发套件包在X86系统中进行升级的时候，如果出现如下情况，即询问是否进行热复位，则用户需输入n，安装完毕后重启os生效，当前版本只支持n选项。

```
The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]
```

步骤1 以软件包的安装用户登录软件包的安装环境。

步骤2 进入软件包所在路径。

步骤3 增加对软件包的可执行权限。

chmod +x *软件包名*.**run**

步骤4 执行如下命令校验软件包安装文件的一致性和完整性。

./*软件包名*.**run --check**

步骤5 软件包升级。

- 若用户安装软件包时指定路径，请执行以下命令。

  ./*软件包名*.**run --upgrade --install-path=***<path>***

  其中*<path>*为用户指定的软件包安装目录，软件包名请根据实际包名进行替换。
- 若用户安装软件包时未指定路径，请执行以下命令。

  ./*软件包名*.**run --upgrade**

----结束

5.1.3 卸载开发套件包

本章节指导用户卸载开发套件包（Ascend-Toolkit-*{version}*-*{aarch}*-linux_*{gccversion}*.run）。

步骤1 以软件包的安装用户登录软件包的安装环境。

步骤2 进入软件包所在路径。

步骤3 软件包卸载。

- 若用户安装软件包时指定路径，执行**./***软件包名*.**run --uninstall --install-path=***<path>***命令完成卸载。

  其中*<path>*为用户指定的软件包安装路径。软件包名请根据实际包名进行替换。
- 若用户安装软件包时未指定路径，执行**./***软件包名*.**run --uninstall**命令对软件包进行卸载。

卸载完成后，若显示如下信息，则说明软件卸载成功：

```
[INFO] xxx uninstall success
[INFO] process end
```

*xxx*表示卸载的实际软件包名。

----结束

5.2 上电 Atlas 200 DK 开发者板

须知

首次启动或者升级Atlas 200 DK开发者板时不能断电，以免对Atlas 200 DK开发者板造成损坏，再次上电需与上次下电时间保持2s以上的安全时间间隔。

步骤1 连接电源线，将电源模块接通外部电源。Atlas 200 DK开发者板的电源接口如图5-2所示。接通电源后，Atlas 200 DK开发者板自动启动。

图 5-2 接口说明

1	电源接口
2	USB
3	SD卡
4	网口

步骤2 检查指示灯的状态，确保Atlas 200 DK开发者板上电正常。

指示灯需要在拆卸上盖后才能看到，指示灯的位置及对应的标号如下所示。

图 5-3 LED 位置标注图（主板名称为 IT21DMDA）

图 5-4 LED 位置标注图（主板名称为 IT21VDMB）

说明

指示灯丝印信息从左至右依次为MINI_LED2、MINI_LED1、3559_ACT、3559_VEDIO，与指示灯从左至右一一对应。

表 5-2 指示灯状态说明 1

MINI_LED2	MINI_LED1	当前Atlas 200 DK 开发者套件（型号 3000）开发者板状态	注意事项
灭	灭	Atlas 200 DK 开发者套件（型号 3000）开发者板启动	可以对Atlas 200 DK 开发者套件（型号 3000）开发者板断电或重启。
灭	亮	Ascend 310启动	可以对Atlas 200 DK 开发者套件（型号 3000）开发者板断电或重启，新版本升级时建议不要断电或重启。

MINI_LED2	MINI_LED1	当前Atlas 200 DK 开发者套件（型号3000）开发者板状态	注意事项
闪烁	闪烁	固件升级	• 不能执行Atlas 200 DK 开发者套件（型号 3000）开发者板断电或重启操作，否则会导致固件升级不完整，单板损坏。 • 当新版本升级时才会有固件升级流程，升级时间比较久，预计在15分钟内，请您耐心等待。
亮	亮	Atlas 200 DK 开发者套件（型号 3000）开发者板启动完成	可以对Atlas 200 DK 开发者套件（型号 3000）开发者板断电或重启。

表 5-3 指示灯状态说明 2

3559_ACT	3559_VEDIO	当前Atlas 200 DK 开发者套件（型号3000）开发者板状态	注意事项
灭	灭	Hi3559C系统未启动	无
灭	亮	Hi3559C系统启动中	无
亮	亮	Hi3559C系统启动完成	无

----结束

5.3 下电 Atlas 200 DK 开发者板

注意事项

请根据**步骤2**的说明判断是否可以对Atlas 200 DK开发者板进行下电操作。

操作步骤

步骤1 从电源接口处拔掉电源线切断外部电源，将Atlas 200 DK开发者板完全下电。

须知

Atlas 200 DK当前不支持操作系统级别的shutdown命令关机。

----结束

5.4 通过串口连接 Atlas 200 DK

通过串口连接 Atlas 200 AI 加速模块

Atlas 200 DK开发者板可以通过如下串口查看Atlas 200 AI加速模块的启动信息。

说明

此串口仅用于查看启动信息，启动成功后，Atlas 200 AI加速模块的串口会被禁用，无法登录系统。

通过串口线连接Atlas 200 AI加速模块的示例图如图5-5所示。

图 5-5 Atlas 200 AI 加速模块串口连线示例图

Atlas 200 AI加速模块串口：串口接线按图5-5颜色标示对应接入。

串口线要求：USB转串口线（3.3V）。

通过串口连接 3559 模块

Atlas 200 DK开发者板提供连接3559模块的串口，串口连线示例图如图5-6所示。

说明

此串口仅用于查看启动信息，启动成功后，3559模块的串口会被禁用，无法登录系统。

图 5-6 3559 串口连线示例图

3559串口（Hi3559）：串口接线按图5-6颜色标示对应接入。

串口线要求：USB转串口线（3.3V）。

5.5 查看开发者板系统软件的版本

介绍查看开发者板系统软件版本的操作。

步骤1 在Ubuntu服务器中以HwHiAiUser用户ssh登录到Atlas 200 DK开发者板。

ssh HwHiAiUser@*192.168.1.2*

说明

192.168.1.2请替换为开发者板实际IP地址。

步骤2 切换到root用户，执行如下命令查看Driver的版本号。

su root

cat /var/davinci/driver/version.info

步骤3 执行如下命令查询Firmware的版本号及有效组件版本号。

1. 切换到root用户。

   su root

2. 执行如下命令查询Firmware的版本号。

   cd /var/davinci/driver/

   ./upgrade-tool --device_index -1 --system_version

   若执行过Atlas 200 DK的Firmware组件升级后，您还可以执行如下命令查询版本号：

   cat /usr/local/Ascend/firmware/version.info

3. 执行如下命令查询Firmware的组件版本号。

   cd /var/davinci/driver/

./upgrade-tool --device_index -1 --component -1 --version

步骤4 执行如下命令查询AI CPU的版本号。

1. 切换到root用户。

   su root

2. 执行如下命令查询AI CPU的版本号。

   cat /var/davinci/aicpu_kernels/version.info

步骤5 执行如下命令，可查询ACLlib的版本号。

cat /home/HwHiAiUser/Ascend/acllib/version.info

----结束

5.6 查看开发者板硬件底板的版本

开发者板的硬件底板版本号可以通过查看开发者底板的PCB的版本号获取。

步骤1 新建查询开发者板版本号的代码文件。

以普通用户在Ubuntu服务器任一目录新建i2c_tool_atlas200dk.c文件。

touch i2c_tool_atlas200dk.c

将以下代码拷贝到i2c_tool_atlas200dk.c文件中。

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include <string.h>
#define I2C0_DEV_NAME  "/dev/i2c-0"
#define I2C1_DEV_NAME  "/dev/i2c-1"
#define I2C2_DEV_NAME  "/dev/i2c-2"
#define I2C3_DEV_NAME  "/dev/i2c-3"
#define I2C_RETRIES    0x0701
#define I2C_TIMEOUT    0x0702
#define I2C_SLAVE      0x21
#define I2C_RDWR       0x0707
#define I2C_BUS_MODE   0x0780
#define I2C_M_RD       0x01
#define PCB_ID_VER_A   0x1
#define PCB_ID_VER_B   0x2
#define PCB_ID_VER_C   0x3
#define PCB_ID_VER_D   0x4
#define I2C_SLAVE_PCA9555_BOARDINFO (0x20)
#define BOARD_ID_DEVELOP_C (0xCE)
#define DEVELOP_A_BOM_PCB_MASK (0xF)
#define DEVELOP_C_BOM_PCB_MASK (0x7)

typedef unsigned char uint8;
typedef unsigned short uint16;
struct i2c_msg
{
    uint16 addr; /* slave address */
    uint16 flags;
```

```
    uint16 len;
    uint8 *buf; /*message data pointer*/
};
struct i2c_rdwr_ioctl_data
{
    struct i2c_msg *msgs; /*i2c_msg[] pointer*/
    int nmsgs; /*i2c_msg Nums*/
};
static uint8 i2c_init(char *i2cdev_name);
static uint8 i2c_read(uint8 slave, unsigned char reg,unsigned char *buf);
int fd = 0;

static uint8 i2c_read(unsigned char slave, unsigned char reg,unsigned char *buf)
{
    int ret;
    struct i2c_rdwr_ioctl_data ssm_msg;
    unsigned char regs[2] = {0};

    regs[0] = reg;
    regs[1] = reg;
    ssm_msg.nmsgs=2;
    ssm_msg.msgs=(struct i2c_msg*)malloc(ssm_msg.nmsgs*sizeof(struct i2c_msg));

    if(!ssm_msg.msgs)
    {
        printf("Memory alloc error!\n");
        return -1;
    }
    (ssm_msg.msgs[0]).flags=0;
    (ssm_msg.msgs[0]).addr=slave;
    (ssm_msg.msgs[0]).buf= regs;
    (ssm_msg.msgs[0]).len=1;

    (ssm_msg.msgs[1]).flags=I2C_M_RD;
    (ssm_msg.msgs[1]).addr=slave;
    (ssm_msg.msgs[1]).buf=buf;
    (ssm_msg.msgs[1]).len=2;

    ret=ioctl(fd, I2C_RDWR, &ssm_msg);
    if(ret<0)
    {
        printf("read data error,ret=%#x, errorno=%#x, %s!\n",ret, errno, strerror(errno));
        free(ssm_msg.msgs);
        return -1;
    }

    free(ssm_msg.msgs);
    return 0;
}

static uint8 i2c_init(char *i2cdev_name)
{
    fd = open(i2cdev_name, O_RDWR);
    if(fd < 0)
    {
        printf("Can't open %s!\n", i2cdev_name);
        return -1;
    }

    if(ioctl(fd, I2C_RETRIES, 1)<0)
    {
        printf("set i2c retry fail!\n");
        return -1;
    }

    if(ioctl(fd, I2C_TIMEOUT, 1)<0)
    {
        printf("set i2c timeout fail!\n");
        return -1;
```

```
    }
    return 0;
}

int main(int argc, char *argv[])
{
    char *dev_name = I2C0_DEV_NAME;
    uint8 board_id;
    uint8 pcb_id;
    uint8 buff[2] = {0};
    uint8 ret;

    if (i2c_init(dev_name))
    {
        printf("i2c init fail!\n");
        close(fd);
        return -1;
    }
    usleep(1000*100);

    ret = i2c_read(I2C_SLAVE_PCA9555_BOARDINFO, 0x0, buff);
    if (ret != 0)
    {
        printf("read %s %#x fail, ret %d\n", dev_name, I2C_SLAVE_PCA9555_BOARDINFO,  ret);
    }

    close(fd);

    board_id = buff[0];
    if (board_id == BOARD_ID_DEVELOP_C)
    {
        pcb_id = (buff[1]>>3)&DEVELOP_C_BOM_PCB_MASK;
    }
    else
    {
        pcb_id = (buff[1]>>4)&DEVELOP_A_BOM_PCB_MASK;
    }

    // show PCB ID;
    switch (pcb_id)
    {
    case PCB_ID_VER_A:
        printf("PCB version is: Ver.A !\n");
        break;
    case PCB_ID_VER_B:
        printf("PCB version is: Ver.B !\n");
        break;
    case PCB_ID_VER_C:
        printf("PCB version is: Ver.C !\n");
        break;
    case PCB_ID_VER_D:
        printf("PCB version is: Ver.D !\n");
        break;
    default:
        break;
    }
    return 0;
}
```

步骤2 编译生成获取PCB版本号的可执行文件。

执行如下命令将i2c_tool_atlas200dk.c文件编译生成在开发者板中可执行的文件。

aarch64-linux-gnu-gcc *i2c_tool_atlas200dk.c* **-o** *atlas200dk_version_tool*

*atlas200dk_version_tool*为生成的可执行文件的名称。

步骤3 将**步骤2**生成的可执行文件上传至开发者板。

例如上传到开发者板的HwHiAiUser用户的家目录下。

scp *atlas200dk_version_tool* **HwHiAiUser@***192.168.1.2***:/home/HwHiAiUser**

步骤4 以HwHiAiUser用户ssh登录到开发者板，并执行查询开发者板PCB版本号的操作。

ssh HwHiAiUser@*192.168.1.2*

切换到root用户，并执行查询脚本。

su root

./atlas200dk_version_tool

如下所示，表示当前开发者板为VB版本。

```
root@davinci-mini:/home/HwHiAiUser# ./atlas200dk_version_tool
PCB version is: Ver.B !
```

----结束

5.7 查看 Atlas 200 AI 加速模块的版本

Atlas 200 AI加速模块的版本号可以通过boardid进行判断，也可以通过Atlas 200 AI加速模块的PCB版本号进行判断，下面分别介绍两种查询方式。

方式一（通过命令查看）

步骤1 以HwHiAiUser用户ssh登录到Atlas 200 DK开发者板。

步骤2 执行如下命令查看Atlas 200 AI加速模块的版本。

cat /proc/cmdline

```
console=ttyAMA0,115200 root=/dev/mmcblk1p1 rw rootdelay=1 syslog no_console_suspend
earlycon=pl011,mmio32,0x10cf80000 initrd=0x880004000,200M cma=256M@0x1FC00000
log_redirect=0x1fc000@0x6fe04000 default_hugepagesz=2M reboot_reason=AP_S_COLDBOOT
himntn=1110001000000000000000000000000000000000000000000000000000000000000
kmemdump=0x7C00020 slotid=00 boardid=000 nr_hugepages=25
```

通过boardid的值判断Atlas 200 AI加速模块的版本：

- boardid=000，表示Atlas 200 AI加速模块的版本为VC及VC以前的版本。
- boardid=004，表示Atlas 200 AI加速模块的版本为VD版本。

----结束

方式二（通过查看 Atlas 200 AI 加速模块的 PCB 版本号）

步骤1 新建查询开发者板版本号的代码文件。

以普通用户在Ubuntu服务器任一目录新建i2c_tool_mini.c文件。

touch i2c_tool_mini.c

将以下代码拷贝到i2c_tool_mini.c文件中。

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <fcntl.h>
#include <sys/select.h>
#include <sys/time.h>
#include <errno.h>
#include <string.h>
#define I2C0_DEV_NAME  "/dev/i2c-0"
#define I2C1_DEV_NAME  "/dev/i2c-1"
#define I2C2_DEV_NAME  "/dev/i2c-2"
#define I2C3_DEV_NAME  "/dev/i2c-3"
#define I2C_RETRIES    0x0701
#define I2C_TIMEOUT    0x0702
#define I2C_SLAVE      0x21
#define I2C_RDWR       0x0707
#define I2C_BUS_MODE   0x0780
#define I2C_M_RD       0x01
#define PCB_ID_VER_A   0x10
#define PCB_ID_VER_B   0x20
#define PCB_ID_VER_C   0x30
#define PCB_ID_VER_D   0x40
typedef unsigned char uint8;
typedef unsigned short uint16;
struct i2c_msg
{
    uint16 addr; /* slave address */
    uint16 flags;
    uint16 len;
    uint8 *buf; /*message data pointer*/
};
struct i2c_rdwr_ioctl_data
{
    struct i2c_msg *msgs; /*i2c_msg[] pointer*/
    int nmsgs; /*i2c_msg Nums*/
};
static uint8 i2c_init(char *i2cdev_name);
static uint8 i2c_write(uint8 slave, unsigned char reg, unsigned char value);
static uint8 i2c_read(uint8 slave, unsigned char reg,unsigned char *buf);
int fd = 0;
static uint8 i2c_write(uint8 slave, unsigned char reg, unsigned char value)
{
    int ret;
    struct i2c_rdwr_ioctl_data ssm_msg;
    unsigned char buf[2]={0};

    ssm_msg.nmsgs=1;
    ssm_msg.msgs=(struct i2c_msg*)malloc(ssm_msg.nmsgs*sizeof(struct i2c_msg));
    if(!ssm_msg.msgs)
    {
        printf("Memory alloc error!\n");
        return -1;
    }

    buf[0] = reg;
    buf[1] = value;

    (ssm_msg.msgs[0]).flags=0;
    (ssm_msg.msgs[0]).addr=(uint16)slave;
    (ssm_msg.msgs[0]).buf=buf;
    (ssm_msg.msgs[0]).len=2;

    ret=ioctl(fd, I2C_RDWR, &ssm_msg);
    if(ret<0)
    {
        printf("write error, ret=%#x, errorno=%#x, %s!\n",ret, errno, strerror(errno));
        free(ssm_msg.msgs);
        return -1;
    }

    free(ssm_msg.msgs);
    return 0;
```

```
}
static uint8 i2c_read(unsigned char slave, unsigned char reg,unsigned char *buf)
{
    int ret;
    struct i2c_rdwr_ioctl_data ssm_msg;
    unsigned char regs[2] = {0};

    regs[0] = reg;
    regs[1] = reg;
    ssm_msg.nmsgs=2;
    ssm_msg.msgs=(struct i2c_msg*)malloc(ssm_msg.nmsgs*sizeof(struct i2c_msg));

    if(!ssm_msg.msgs)
    {
        printf("Memory alloc error!\n");
        return -1;
    }
    (ssm_msg.msgs[0]).flags=0;
    (ssm_msg.msgs[0]).addr=slave;
    (ssm_msg.msgs[0]).buf= regs;
    (ssm_msg.msgs[0]).len=1;

    (ssm_msg.msgs[1]).flags=I2C_M_RD;
    (ssm_msg.msgs[1]).addr=slave;
    (ssm_msg.msgs[1]).buf=buf;
    (ssm_msg.msgs[1]).len=1;

    ret=ioctl(fd, I2C_RDWR, &ssm_msg);
    if(ret<0)
    {
        printf("read data error,ret=%#x, errorno=%#x, %s!\n",ret, errno, strerror(errno));
        free(ssm_msg.msgs);
        return -1;
    }

    free(ssm_msg.msgs);
    return 0;
}
static uint8 i2c_init(char *i2cdev_name)
{
    fd = open(i2cdev_name, O_RDWR);
    if(fd < 0)
    {
        printf("Can't open %s!\n", i2cdev_name);
        return -1;
    }

    if(ioctl(fd, I2C_RETRIES, 1)<0)
    {
        printf("set i2c retry fail!\n");
        return -1;
    }

    if(ioctl(fd, I2C_TIMEOUT, 1)<0)
    {
        printf("set i2c timeout fail!\n");
        return -1;
    }
    return 0;
}
int main(int argc, char *argv[])
{
    char *dev_name = I2C0_DEV_NAME;
    uint8 slave;
    uint8 reg;
    uint8 data;
    int ret;

    if (i2c_init(dev_name))
```

```
    {
        printf("i2c init fail!\n");
        close(fd);
        return -1;
    }
    usleep(1000*100);

    // Read PCB ID
    slave = I2C_SLAVE;
    reg = 0x07;
    data = 0x5A;

    ret = i2c_read(slave, reg, &data);
    if (ret != 0)
    {
        printf("read %s %#x %#x to %#x fail!\n", dev_name, slave, data, reg);
    }

    slave = I2C_SLAVE;
    reg = 0x07;
    data = data|0xF0;

    ret = i2c_write(slave, reg, data);
    if (ret != 0)
    {
        printf("write %s %#x %#x to %#x fail!\n", dev_name, slave, data, reg);
    }

    slave = I2C_SLAVE;
    reg = 0x01;
    data = 0x5A;

    ret = i2c_read(slave, reg, &data);
    if (ret != 0)
    {
        printf("read %s %#x %#x to %#x fail!\n", dev_name, slave, data, reg);
    }

    close(fd);

    // show PCB ID;
    switch (data & 0xF0)
    {
    case PCB_ID_VER_A:
        printf("PCB version is: Ver.A !\n");
        break;
    case PCB_ID_VER_B:
        printf("PCB version is: Ver.B !\n");
        break;
    case PCB_ID_VER_C:
        printf("PCB version is: Ver.C !\n");
        break;
    case PCB_ID_VER_D:
        printf("PCB version is: Ver.D !\n");
        break;
    default:
        break;
    }
    return 0;
}
```

步骤2 编译生成获取PCB版本号的可执行文件。

执行如下命令将i2c_tool_mini.c文件编译生成在开发者板中可执行的文件。

aarch64-linux-gnu-gcc *i2c_tool_mini.c* **-o** *mini_version_tool*

*mini_version_tool*为生成的可执行文件的名称。

步骤3 将步骤2生成的可执行文件上传至开发者板。

例如上传到开发者板的HwHiAiUser用户的家目录下。

scp *mini_version_tool* **HwHiAiUser@***192.168.1.2***:/home/HwHiAiUser**

步骤4 以HwHiAiUser用户ssh登录到开发者板，并执行查询开发者板PCB版本号的操作。

ssh HwHiAiUser@*192.168.1.2*

切换到root用户，并执行查询脚本。

su root

./mini_version_tool

如下所示，表示当前AI加速模块的版本为VC版本。

```
root@davinci-mini:/home/HwHiAiUser# ./mini_version_tool
PCB version is: Ver.C !
```

----结束

5.8 如何查看开发者板的 firmware 版本

若无法正常进入Atlas 200 DK开发者板操作系统，可通过查看Atlas 200 DK串口启动日志的方式查看固件的版本号，操作步骤如下，若能正常进入Atlas 200 DK开发者板操作系统，firmware版本号查看可参考**5.5 查看开发者板系统软件的版本**。

步骤1 通过串口连接Atlas 200 DK的Atlas 200 AI加速模块，连接方式请参考**5.4 通过串口连接Atlas 200 DK**。

步骤2 上电Atlas 200 DK开发者板，通过串口终端查看打印日志，如下图所示打印日志即为固件版本号。

图 5-7 查询固件版本号

```
Line 69: [NVE]current_id = 3 number of nvbin = 19, version = 268963089
Line 164: Xloader version is 00000001.00000001.00000013.00000810
Line 252: UEFI version is 1.1.13.810
```

- Xloader的版本号为1.1.T13.B810
- UEFT的版本号为1.1.T13.B810

----结束

5.9 如何查看摄像头所属 Channel

Atlas 200 DK有两个MIPI-CSI接口，可外接两个摄像头。

可通过查看开发者板面板判断当前所使用的摄像头通道，如**图5-8**所示。

图 5-8 查看摄像头所属通道

- “CAMERA0”对应的摄像头通道为“Channel-1”
- “CAMERA1”对应的摄像头通道为“Channel-2”。

5.10 如何安装 windows 的 USB 网卡驱动

Ubuntu服务器通过Windows操作系统上的虚拟机安装的场景下，需要在Windows操作系统上安装USB网卡驱动，即RNDIS（Remote Network Driver Interface Specification）驱动，否则Atlas 200 DK通过USB连接Ubuntu所在的Windows主机时，在Ubuntu操作系统中无法识别Atlas 200 DK的USB虚拟网卡。

假设您已经将Atlas 200 DK通过USB连接线连接到了Ubuntu所在的Windows主机，下面以在Windows 10操作系统上安装RNDIS驱动为例进行介绍。

步骤1 在“计算机管理”操作界面中选择“设备管理器 >其他设备”，如下图所示，RNDIS为未识别状态。

图 5-9 设备管理器

步骤2 右键单击“RNDIS”，选择“更新驱动程序(P)”。

图 5-10 更新 RNDIS

步骤3 在弹出的“更新驱动程序 - RNDIS窗口”中选择“浏览我的计算机以查找驱动程序软件（R）”，然后选择“让我从计算机上的可用驱动程序列表中选取(L)”，单击“下一步(N)”。

步骤4 在“常见硬件类型”列表中选择“网络适配器”，单击“下一步(N)”。

图 5-11 选择网络适配器

步骤5 在“选择要为此硬件安装的设备驱动程序”界面中选择“Microsoft”厂商的“USB RNDIS6适配器”。

图 5-12 选择驱动程序

步骤6 单击“下一步”，在弹出的“更新驱动程序警告”窗口选择“是”。

步骤7 返回“设备管理器 > 网络适配器”，可看到已经正常显示了USB RNDIS6适配器的驱动。

图 5-13 RNDIS 驱动正常显示

----结束

5.11 设置用户有效期

为保证用户的安全性，应设置用户的有效期，使用系统命令chage来设置用户的有效期。

命令为：

```
chage [-m mindays] [-M maxdays] [-d lastday] [-I inactive] [-E expiredate] [-W warndays] user
```

相关参数请参见表5-4。

表 5-4 设置用户有效期

参数	参数说明
-m	口令可更改的最小天数。设置为“0”表示任何时候都可以更改口令。

参数	参数说明
-M	口令保持有效的最大天数。设置为“-1”表示可删除这项口令的检测。
-d	上一次更改的日期。
-I	停滞时期。过期指定天数后，设定密码为失效状态。
-E	用户到期的日期。超过该日期，此用户将不可用。
-W	用户口令到期前，提前收到警告信息的天数。
-l	列出当前的设置。由非特权用户来确定口令或帐户何时过期。

说明

- 表5-4只列举出常用的参数，用户可通过**chage --help**命令查询详细的参数说明。
- 日期格式为YYYY-MM-DD，如chage -E 2020-12-01 *test*表示用户*test*的口令在2020年12月1日过期。
- User必须填写，填写时请替换为具体用户，默认为root用户。

举例说明：修改用户*test*的有效期为2020年12月01日。

```
chage -E 2020-12-01 test
```

5.12 安装 3.5.2 版本 cmake

1. 使用wget下载cmake源码包，可以下载到安装服务器任意目录，命令为：
```
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz --no-check-certificate
```
2. 进入下载后的目录，解压源码包，命令为：
```
tar -zxvf cmake-3.5.2.tar.gz
```
3. 进入解压后的文件夹，执行配置，编译和安装命令：
```
cd cmake-3.5.2
./bootstrap --prefix=/usr
make
sudo make install
```
4. 安装完成后重新执行**cmake --version**查看版本号。

5.13 配置系统网络代理

以下步骤是配置网络代理的通用方法，不一定能适配所有的网络环境，网络代理的配置方式以现场实际的网络环境为准。

前提条件

- 确保服务器网线已连接，代理服务器能连接外网。
- 配置代理是建立在服务器处在内部网络，无法直接连接外部网络的条件下。

配置系统网络代理

步骤1 使用root权限登录用户环境。

步骤2 使用如下命令编辑“/etc/profile”文件：

```
vi /etc/profile
```

在文件最后添加如下内容后保存退出：

```
export http_proxy="http://user:password@proxyserverip:port"
export https_proxy="http://user:password@proxyserverip:port"
```

其中user为用户在内部网络中的用户名，password为用户密码，proxyserverip为代理服务器的ip地址，port为端口。

步骤3 执行如下命令使配置生效。

```
source /etc/profile
```

步骤4 执行如下命令测试是否连接外网。

```
wget www.baidu.com
```

如果能下载到网页html文件则表明服务器已成功连接外网。

说明

使用代理连接网络，如果出现证书错误，则需要先安装代理服务器的证书，才能正常下载第三方组件。

----结束

5.14 参数说明

软件包支持根据命令行完成一键式安装，各个命令之间可以配合使用，用户根据安装需要选择对应参数完成安装，所有参数都是可选参数。

安装命令格式： **./*.run** [options]

详细参数请参见表5-5。

须知

如果通过**./*.run --help**命令查询出的参数未解释在如下表格，则说明该参数预留或适用于其他芯片版本，用户无需关注。

表 5-5 安装包支持的参数说明

参数	说明
--help | -h	查询帮助信息。
--version	查询版本信息。
--info	查询软件包构建信息。
--list	查询软件包文件列表。
--check	检查软件包的一致性和完整性。

参数	说明
--quiet	静默安装，跳过交互式信息。
--noexec	解压软件包到当前目录，但不执行安装脚本。配套--extract=<path>使用，格式为： --noexec --extract=<path>。
--extract=<path>	解压软件包中文件到指定目录。
--tar arg1 [arg2 ...]	对软件包执行tar命令，使用tar后面的参数作为命令的参数。例如执行**--tar xvf**命令，解压run安装包的内容到当前目录。
--install	安装软件包。后面可以指定安装路径--install-path=<path>，也可以不指定安装路径，直接安装到默认路径下。
--install-for-all	安装或升级时，允许其他用户具有安装群组的权限。 当安装或者升级携带该参数时，软件包中创建的目录及文件，其他用户权限=安装群组权限。 该参数需要与--install、--devel、--upgrade等其中一个参数配合使用，例如**./*.run** --install --install-for-all **说明** 使用该参数将会存在安全风险：其他所有用户都有权限访问安装目录，请谨慎使用。
--install-username=<username>	首次安装：可以指定运行用户名，若不指定，则默认是HwHiAiUser。 覆盖安装：沿用上次运行用户名。 该参数需要配合“--install-usergroup=<usergroup>”一起使用。
--install-usergroup=<usergroup>	首次安装：可以指定运行用户组，若不指定，则默认是HwHiAiUser。 覆盖安装：沿用上次运行用户组。 该参数需要配合“--install-username=<username>”一起使用。
--install-path=<path>	指定安装路径，若不指定，将安装到默认路径下： ● 若使用root用户安装，默认安装路径为：/usr/local/Ascend。 ● 若使用非root用户安装，则默认安装路径为：${HOME}/Ascend。 若通过该参数指定了安装目录，运行用户需要对指定的安装路径有可读写权限。
--uninstall	卸载已安装的软件。
--upgrade	升级已安装的软件。自动进行版本号检查，如果版本号不是从小到大，则不能升级。
--devel	按照开发模式安装软件包，即只安装开发环境需安装的文件。

6 FAQ

6.1 run包安装时提示软件已经安装

6.2 pip3.7.5 install报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”

6.3 使用pip3.7.5 install软件时提示" Could not find a version that satisfies the requirement xxx"

6.4 SD卡制作过程中人为拔卡导致系统存在冗余挂载盘

6.5 Ubuntu服务器连接开发者板无法建立信任关系

6.6 开发者板无法正常连接Ubuntu服务器如何处理

6.1 run 包安装时提示软件已经安装

问题描述

软件包安装时提示：

```
run package is already installed, install failed
```

解决方案

软件包不支持重复安装，需要先卸载后，再安装。卸载方法请参见**5.1.3 卸载开发套件包**。

6.2 pip3.7.5 install 报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”

问题描述

安装依赖时，使用**pip3.7.5 install xxx**命令安装相关软件时报错“subprocess.CalledProcessError：Command '('lsb_release', '-a')' return non-zero exit status 1”，提示信息如下：

```
root@debian:/home/work_env/Python-3.7.5# pip3.7.5 install attrs
ERROR: Exception:
Traceback (most recent call last):
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 188, in main
    status = self.run(options, args)
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/commands/install.py", line 286, in run
    with self._build_session(options) as session:
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/cli/base_command.py", line 108, in _build_session
    index_urls=self._get_index_urls(options),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/download.py", line 559, in __init__
    self.headers["User-Agent"] = user_agent()
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_internal/download.py", line 144, in user_agent
    zip(["name", "version", "id"], distro.linux_distribution()),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 122, in linux_distribution
    return _distro.linux_distribution(full_distribution_name)
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 677, in linux_distribution
    self.version(),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 737, in version
    self.lsb_release_attr('release'),
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 899, in lsb_release_attr
    return self._lsb_release_info.get(attribute, '')
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 552, in __get__
    ret = obj.__dict__[self._fname] = self._f(obj)
  File "/usr/local/python3.7.5/lib/python3.7/site-packages/pip/_vendor/distro.py", line 1012, in _lsb_release_info
    stdout = subprocess.check_output(cmd, stderr=devnull)
  File "/usr/local/python3.7.5/lib/python3.7/subprocess.py", line 411, in check_output
    **kwargs).stdout
  File "/usr/local/python3.7.5/lib/python3.7/subprocess.py", line 512, in run
    output=stdout, stderr=stderr)
subprocess.CalledProcessError: Command '('lsb_release', '-a')' returned non-zero exit status 1.
```

可能原因

用户自行编译安装的python3.7.5在执行subprocess模块时，在执行lsb_release -a 时提示找不到lsb_release.py模块，用户自行编译安装的python3.7.5的lib路径是“/usr/local/python3.7.5/lib/python3.7/”，该路径下没有lsb_release.py模块，因此会报错。

解决方法

步骤1 查找缺失文件'lsb_release.py'，使用如下命令：

```
find / -name lsb_release
```

执行上述命令后，得到如下路径，以下仅为示例，用户实际可能有差别：

```
/usr/bin/lsb_release
```

步骤2 将**步骤1**找到的“/usr/bin/lsb_release”文件删除：

```
rm /usr/bin/lsb_release
```

步骤3 然后执行**pip3.7.5 list**查看是否解决。

----结束

6.3 使用 pip3.7.5 install 软件时提示" Could not find a version that satisfies the requirement xxx"

问题描述

安装依赖时，使用**pip3.7.5 install xxx**命令安装相关软件时提示无法连接网络，且提示“Could not find a version that satisfies the requirement xxx"，提示信息如所示。

图 6-1 使用 pip3 安装软件提示信息

```
ascend@dggphicprd32833:~$ pip3 install numpy --user
Collecting numpy
  Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at
7f5f725b9320>: Failed to establish a new connection: [Errno 101] Network is unreachable',)': /simple/numpy/
  Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at
7f5f725cd908>: Failed to establish a new connection: [Errno 101] Network is unreachable',)': /simple/numpy/
  Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at
7f5f725cd9e8>: Failed to establish a new connection: [Errno 101] Network is unreachable',)': /simple/numpy/
  Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at
7f5f725cdac0>: Failed to establish a new connection: [Errno 101] Network is unreachable',)': /simple/numpy/
  Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('<urllib3.connection.VerifiedHTTPSConnection object at
7f5f725cdb38>: Failed to establish a new connection: [Errno 101] Network is unreachable',)': /simple/numpy/
  Could not find a version that satisfies the requirement numpy (from versions: )
```

可能原因

没有配置pip源。

解决方法

配置pip源，配置方法如下：

步骤1 使用软件包的安装用户，执行如下命令：

```
cd ~/.pip
```

如果提示目录不存在，则执行如下命令创建：

```
mkdir ~/.pip
cd ~/.pip
```

在.pip目录下创建pip.conf 文件，命令为：

```
touch pip.conf
```

步骤2 编辑pip.conf文件。

使用**vi pip.conf**命令打开pip.conf文件，写入如下内容：

```
[install]
#可信主机，请根据实际情况进行替换。
trusted-host=cmc-cd-mirror.rnd.huawei.com
[global]
#可用的源，请根据实际情况进行替换。
index-url=http://cmc-cd-mirror.rnd.huawei.com/pypi/simple/
```

步骤3 执行**:wq!**命令保存文件。

----结束

6.4 SD 卡制作过程中人为拔卡导致系统存在冗余挂载盘

SD卡制作过程中人为拔卡导致系统出现冗余临时挂载盘，可使用如下步骤卸除。

步骤1 使用普通用户登录Ubuntu服务器，并执行**su - root**命令切换到root用户。

步骤2 输入命令**df -h**，查看到/dev/loop0临时挂载盘。

```
root@kickseed:~# df -h
Filesystem      Size    Used    Avail   Use% Mounted on
/dev/loop0      745M  745M    0       100% /home/ubuntu/studio/scripts/180919002200
/dev/sdc1       118G   60M     112G  1%    /home/ubuntu/studio/scripts/sd_mount_dir
```

步骤3 使用**umount**命令卸除挂载盘，命令中的/dev/loop0、/dev/sdc1设备请用户根据**步骤2**实际查询结果调整。

```
root@kickseed:~# umount /dev/loop0
root@kickseed:~# umount /dev/sdc1
```

若命令提示**target is busy**，请尝试重启Ubuntu服务器，再重新**步骤1**至**步骤3**。

----结束

6.5 Ubuntu 服务器连接开发者板无法建立信任关系

问题描述

在Ubuntu服务器中，执行如下命令使用SSH方式连接Atlas 200 DK开发者板，提示无信任关系。

在Ubuntu服务器中执行如下命令重新建立信任关系：

ssh-keygen -f "$HOME/.ssh/known_hosts" -R *192.168.1.2*

其中192.168.1.2为Atlas 200 DK开发者板的IP地址。

报如下错误：

```
ECDSA host key for 192.168.1.2 has changed and you have requested strict checking.
```

解决方法

出现此错误是本地保存的SSH信息已失效导致，所以需要清空当前保存的SSH信息，然后重新建立连接。

步骤1 清空Ubuntu服务器中当前用户连接*192.168.1.2*主机的公钥信息。

ssh-keygen -R *192.168.1.2*

步骤2 重新以SSH方式连接Atlas 200 DK开发者板。

ssh HwHiAiUser@192.168.1.2

当提示如下信息时，输入**yes**重新建立SSH连接。

```
The authenticity of host '192.168.1.2' can't be established.
ECDSA key fingerprint is 53:b9:f9:30:67:ec:34:88:e8:bc:2a:a4:6f:3e:97:95.
Are you sure you want to continue connecting (yes/no)?
```

----结束

6.6 开发者板无法正常连接 Ubuntu 服务器如何处理

问题描述

有以下几种现象：

- 将制作好的SD卡插入开发者板并上电后，开发者板LED1与LED2灯状态信息异常。
- 将制作好的SD卡插入开发者板，并通过USB方式连接Ubuntu服务器，上电、开发者板启动完成后，Ubuntu服务器无虚拟网卡信息。
- 将制作好的SD卡插入开发者板，并通过NIC方式连接Ubuntu服务器，上电、开发者板启动完成，配置Ubuntu服务器网卡信息后，Ubuntu服务器无法与开发者板通信。

解决思路

按照如图6-2所示流程进行排查。

图 6-2 Atlas 200 DK 无法连接 Ubuntu 服务器问题排查思路

解决方法

步骤1 确认SD卡制作正确并成功。

请查看制卡脚本所在目录下的sd_card_making_log查看是否制卡成功，若制卡失败，请参考**4.1.1 制作SD卡**重新制作SD卡。

步骤2 将成功制作的SD卡插入Atlas 200 DK开发者板，并上电。

- 若Atlas 200 DK开发者板的LED1与LED2状态正常，即启动成功后，LDE1与LDE2同时处于亮的状态，则执行**步骤3**。
- 若Atlas 200 DK开发者板的LED1与LED2状态异常，即启动很长一段时间后（15分钟以后），LED1与LED2不同时为亮的状态，首先请参考**异常处理**查看系统软件安装日志及Atlas 200 DK启动日志。

  若仍无法解决，则执行**步骤4**。

步骤3 连接Atlas 200 DK开发者板与Ubuntu服务器。

- 若通过USB方式连接开发者板，但Ubuntu服务器中不显示虚拟USB网卡。

  首先检查USB网线，确保USB网线两端口连接正常。

  若Ubuntu服务器中仍然不显示USB虚拟网卡，请尝试使用NIC方式连接。
- 若通过NIC方式连接开发者板，配置好Ubuntu服务器的IP信息后，Ubuntu服务器无法与开发者板通信。

  首先检查网线，确保网线两端口正常，然后重新配置Ubuntu服务器的IP地址。

  若Ubuntu服务器仍然无法与开发者板通信，请尝试使用USB方式连接开发者板。

若USB方式与NIC方式，都无法正常连接Ubuntu服务器与开发者板，请执行步骤**步骤4**。

步骤4 参考**5.4 通过串口连接Atlas 200 DK**将Atlas 200 DK开发者板的Atlas 200 AI加速模块通过串口线与Ubuntu服务器相连。

步骤5 在Ubuntu服务器中安装网络调试工具与USB转串口驱动软件。

- 网络调试工具推荐使用IPOP工具。
- USB转串口驱动软件请使用PL2303驱动软件。

步骤6 打开网络工程调试工具，以IPOP工具为例，进入串口窗口。

1. 单击“终端工具”页签。
2. 在菜单栏选择 ，进入“设置”窗口。
3. 进行连接配置。
   - 连接名称：自定义连接名称。
   - 类型：选择COMX，可以通过计算机的设备管理器查看可用的COM端口，拔插Ubuntu服务器上的串口连接线，判断Atlas 200 DK使用的哪个COM口，如**图6-3**所示。

图 6-3 查看 COM 端口

– 设置波特率为115200。

4. 单击确定。

步骤7 上电Atlas 200 DK开发者板，在IPOP的COM连接窗口中查看Atlas 200 DK启动信息。

由于启动日志较多，单击菜单栏中的将启动日志保存到IPOP工具的安装目录中，当此按钮变为，IPOP工具底部会出现文件已保存的提示信息，可以根据提示信息在IPOP安装目录中获取以当前时间命名的日志文件。

步骤8 在**Ascend论坛**上发求助帖，并将启动日志信息作为帖子的附件上传，将会有华为工程师为您解答。

----结束