

Docker编译环境搭建

立创·泰山派介绍

泰山派是一款开源的卡片电脑，提供全面开放的软硬件资料，愿与志同道合的伙伴们共同推动技术的发展和创新。

小巧的板子搭载了高配的处理器的、引出丰富的外部资源、多样性SDK、赋予创意无限可能。

软硬件结合、项目式学习、解决项目落地难问题、让每一个想法变成现实。



共建开源生态

共建开源生态是我们的目标。为此我们努力把性价比和颜值做到极致。

同时全面开放SDK、原理图、PCB等所有软硬件资料。

提供交流平台，免费训练营、开发者扶持计划等，让您能够尽情发挥您的才华，

我们坚信自由和定制化是开源生态的核心，

泰山派开发板不仅仅是立创开发板的产品，它更是我们共同努力的成果。

我们只是抛砖引玉，期望能够激发出更多的创意和想法，

我们渴望与志同道合的伙伴们携手合作，共同探索无限可能。

让我们各展所长，共同建设一个强大的开源生态系统。

泰山派开发板等待着您的加入。

让我们一起推动技术的前进，一起创未来！



LC@SC 立创开发板
WWW.LCKFB.COM

扫码购买



立创·泰山派开发板1G+0G版本



立创·泰山派开发板2G+16G版本

问题反馈

问题反馈：

- 如果您在阅读过程中有任何疑问或建议，欢迎随时与我们联系。
我们随时为您解答问题，提供帮助。您的反馈将帮助我们更好提升我们的产品质量，至此诚挚感谢。
- 在反馈问题时，请尽可能提供以下信息以便我们更快速地为为您提供解决方案：
问题描述：简述您遇到的问题，包括触发问题的操作步骤和结果。
错误信息：附上任何错误信息或截图以协助我们的分析工作。
设备信息：您使用的开发板型号、固件版本和操作系统。

联系方式



文档更新说明

日期	版本	作者	修改说明
2023-09-11	v1.0	立创开发板	初稿

免责声明

本文档虽力求准确，但可能存在错误或遗漏，用户需谨慎评估适用性；我们保留随时修订的权利，且不能保证旧版本内容的有效性，为了得到最新版本的产品信息，请定时访问立创开发板官网进行资料的更新下载。本文档版权及最终解释权归深圳嘉立创科技集团股份有限公司所有。请务必注意以上声明，自行评估风险并决策，我们对使用本文档产生的任何问题概不负责。感谢您的包容与支持。

前言

在我们编译泰山派SDK时候，可能会因为大家的电脑环境不同导致编译失败，如果使用docker就能很好的避免环境不同意带来的编译问题，极大提供高编译成功率，这种方法也是我们主推的编译方法，下面是我测试过并长期使用的docker环境，建议大家编译泰山派SDK时候使用此方法。

Docker 的安装

卸载旧版本

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

使用存储库安装

更新 apt 包索引并安装相关软件包以允许 apt 通过 HTTPS 使用 docker 存储库：

```
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg lsb-release
```

添加 Docker 的官方 GPG 密钥：

```
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

设置存储库：

```
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
$(lsb_release -cs) stable" | sudo tee -a /etc/apt/sources.list.d/docker.list >
/dev/null
```

安装 Docker engine

```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

验证安装完成

```
sudo docker -v
# 打印出版本号
Docker version 24.0.2, build cb74dfc
```

创建镜像

当我们从 docker 镜像仓库中下载的镜像不能满足我们的需求时，我们可以通过以下两种方式对镜像进行更改。

- 1、从已经创建的容器中更新镜像，并且提交这个镜像
- 2、使用 Dockerfile 指令来创建一个新的镜像

```
#创建目录
root@filex-virtual-machine:/home/LCKFB_RK3566# mkdir docker_lckfb_android_sdk/
#进入目录
root@filex-virtual-machine:/home/LCKFB_RK3566# cd docker_lckfb_android_sdk/
编写Dockerfile文件
root@filex-virtual-machine:/home/LCKFB_RK3566/docker_lckfb_android_sdk# vim
dockerfile
```

deckerfile文件内容

```
# 设置基础镜像为Ubuntu 18.04
FROM ubuntu:18.04

# 设置作者信息
MAINTAINER lckfb "service@lckfb.com"

# 设置环境变量，用于非交互式安装
ENV DEBIAN_FRONTEND=noninteractive
```

```

# 备份源列表文件
RUN cp -a /etc/apt/sources.list /etc/apt/sources.list.bak

# 将源列表中的 http://.*ubuntu.com 替换为 http://repo.huaweicloud.com
RUN sed -i 's@http://.*ubuntu.com@http://repo.huaweicloud.com@g'
/etc/apt/sources.list

# 更新包列表
RUN apt update

# 安装基本的编译工具和依赖
RUN apt install -y build-essential crossbuild-essential-arm64 \
    bash-completion vim sudo locales time rsync bc python

# 安装其他依赖包，这里编译android11sdk需要的环境
RUN apt install -y repo git ssh libssl-dev liblz4-tool lib32stdc++6 \
    expect patchelf chrpath gawk texinfo diffstat binfmt-support \
    qemu-user-static live-build bison flex fakeroot cmake \
    unzip device-tree-compiler python-pip ncurses-dev python-pyelftools \
    subversion asciidoc w3m dblatex graphviz python-matplotlib cpio \
    libparse-yapp-perl default-jre patchutils swig expect-dev u-boot-tools

RUN apt install -y bear

# 再次更新包列表并安装任何未安装的依赖
RUN apt update && apt install -y -f

# 生成本地化语言支持
RUN locale-gen en_US.UTF-8
ENV LANG en_US.UTF-8

# 创建开发板用户
RUN useradd -c 'lckfb user' -m -d /home/lckfb -s /bin/bash lckfb

# sudo免密登录
RUN sed -i -e '/\%sudo/ c \\\%sudo ALL=(ALL) NOPASSWD: ALL' /etc/sudoers
RUN usermod -a -G sudo lckfb

USER lckfb
#设置docker工作目录为/home/lckfb
WORKDIR /home/lckfb

#容器使用这个内核方法
#docker run --privileged --mount
type=bind,source=/home/LCKFB_RK3566/ROCKCHIP_ANDROID11.0_SDK_RELEASE/ROCKCHIP_AND
ROID11.0_SDK_RELEASE,target=/home/lckfb/android11 --name="lckfb_android11_sdk" -h
lckfb -it lckfb_android11_sdk_cmp

```

创建镜像

在 Dockerfile 文件的存放目录下，执行构建动作

```

docker build -t lckfb_android11_sdk_cmp .
# lckfb_android11_sdk_cmp是镜像名称，可随意更改，注意命令最后有一个‘.’
# 此过程需要一段时间，请耐心等待

```

如果创建镜像时候出现了命令错误可以通过先运行下面进行测试

```
docker run --rm -it ubuntu 18.04 bash
```

- `docker run` 是用于创建并运行容器的命令。
- `--rm` 参数指示 Docker 在容器停止后自动删除该容器。这样可以确保不会在系统中留下不再使用的停止状态容器。
- `-it` 参数结合了 `-i` (交互式) 和 `-t` (终端) 两个选项, 使得容器中的命令可以接收终端输入并显示输出。
- `<镜像名称或ID>` 是指要基于哪个镜像创建容器。您可以提供镜像的名称或ID来唯一标识一个镜像。
- `<要运行的命令>` 是您希望在容器中运行的命令或指令。

查看镜像

```
root@filex-virtual-machine:/home/EX_DISK_4T_20230726/lckfb_tspi/android11/docker#  
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tsp_android11_cmp_img	latest	998f0ecd6d5d	About a minute ago	1.93GB
lckfb_android11_sdk_cmp	latest	d8e470a39b2b	About a minute ago	1.93GB

基于镜像创建容器

```
docker run --privileged --mount  
type=bind,source=/home/wucaicheng/gw_work/EX_DISK_A,target=/home/lckfb --  
name="lckfb_android11_sdk" -h lckfb -it lckfb_android11_sdk_cmp
```

- `--privileged`: 在容器内启用特权模式, 允许容器内的进程访问所有主机设备。
- `--mount`
`type=bind,source=/home/wucaicheng/gw_work/EX_DISK_A,target=/home/lckfb`: 创建绑定类型的挂载点, 将宿主机上的 `/home/wucaicheng/gw_work/EX_DISK_A` 目录挂载到容器内的 `/home/lckfb` 目录。这使得容器内的应用程序可以访问并操作宿主机上的该目录。
- `--name="lckfb_android11_sdk"`: 为容器指定一个名称, 这样可以在后续的Docker命令中使用该名称引用容器。
- `-h lckfb`: 设置容器的主机名为 `lckfb`。
- `-it`: 以交互式 and 终端的方式运行容器。
- `lckfb_android11_sdk_cmp`: 使用名为 `lckfb_android11_sdk_cmp` 的镜像来创建容器。

列出所有的容器

`docker ps -a` 包括正在运行的和已经停止的容器

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
7258412811df	lckfb_android11_sdk_cmp	<code>"/bin/bash"</code>	29 minutes ago
Exited (0) 26 minutes ago		lckfb_android11_sdk	

- CONTAINER ID：容器的唯一ID。
- IMAGE：容器所使用的镜像。
- COMMAND：容器启动时运行的命令。
- CREATED：容器的创建时间。
- STATUS：容器的当前状态（正在运行、已停止等）。
- PORTS：容器的端口映射信息。
- NAMES：容器的名称。

启动容器

```
docker start lckfb_android11_sdk #启动容器
docker attach lckfb_android11_sdk #附加到正在运行的名为lckfb_android11_sdk的容器的终端上，以便与容器进行交互
```

退出容器

```
lckfb@lckfb:~$ exit
```