Phytium飞腾

FT2004 安全 CPU 系统搭建手册

v1.0

飞腾信息技术有限公司

2021 年 10 月

更新记录

版本号	发布部门	作者	发布日期	备　注
1.00	飞腾通用软件部	田伟	2021-11-11	初稿

目录

1 介绍

TEE 全称为 Trusted execute environment，也就是信任执行环境。TEE 是基于 trustzone 技术搭建的安全执行环境，trustzone 技术由 Arm 公司提出，用一根安全总线（称为 NS 位）来判断当前处于 secure world 还是 non-secure world 状态，状态的切换由 ATF(arm trusted firmware)来完成，当处于 secure world 状态，执行 TEE OS 部分的代码，当处于 non-secure world 状态时，就执行 linux kernel 部分的代码，Linux kernel 不能直接访问 TEE 部分的资源，Linux kernel 能通过特定的 TA（Trust Appliaction）和 CA（Client Application）来访问 TEE 部分特定的资源，TA 主要处理保密信息，如信用卡 pin 码，私有密码，客户数据，受 DRM（Digital Rights Management，数字版权管理）保护的媒体，TEE 需要硬件软件的协同支持，目前飞腾芯片只有 FT2000/4 和 D2000 完整地支持 TEE，本文档以 FT2000/4 为例讲述 TEE 环境的搭建过程。

2 软硬件环境

2.1 硬件环境

汉为 FT2000/4 桌面机。

2.2 软件环境

Ubuntu20.04 系统。

3 编译 pbtee

3.1 配置编译环境

编译 pbtee 过程中需要 Python3 的 pyelftools 和 pycryptodome 库支持，执行如下命令安装：

```
# sudo apt-get install python3-pyelftools
# sudo apt-get install python3-pycryptodome
```

3.2 获取 pbtee 源码

以 PBTEE v3.0 为例，从 SDK 中获取 pbtee-v3.0-release.tgz 文件，并上传至系统任意目录，进入此目录，执行如下命令解压 pbtee 源码：

```
# sudo tar -zxvf pbtee-v3.0-release.tgz
```

解压后是一个 tar 包文件，文件名的后六位是 MD5 值，pbtee 的 v3.0 版本的 MD5 值后六位是 901590，不会有任何其他不同的值，可以进行验证是否被篡改。

3.3 编译 pbtee

解压 pbtee 源码后执行如下命令进入进入 pbtee 主目录：

```
# sudo cd pbtee
```

接着执行如下命令一键编译 pbtee：

```
# sudo ./auto-compile-all ft2004r
```

编译成功后，会自动在主目录生成一个 out 目录，out 目录中的 data-xxxx.tgz 和 tee-phytium-xxxx.bin 文件分别就是 REE 侧的支持包和 TEE 侧的映像。（如果编译过程中出现错误，可以采用错误出现目录中的编译脚本先进行手动编译，等待问题解决后，可以执行

```
# sudo ./auto-clean-all
```

命令清理冗余文件，再进行整体编译。）

4 固件打包

固件打包的目的是将所有需要固化在 spi flash 中的软件通过一定规则打包成一个可以烧录的二进制文件，方便烧录器一次烧录到 spi flash 芯片中去。SDK 提供了自动化打包环境 image_fix 和最新的 pbf 映像。相关文件存放在“固件打包”目录下。

4.1 解压打包工具（image_fix_xxx.tar）

上传 image_fix_xxx.tar 至系统任意目录后，进入此目录，执行如下命令解压打包工具（以 v14 版本为例）：

```
# sudo tar -zxvf image_fix_v14.tar
```

4.2 更新 PBF 映像

解压后会生成一个 image_fix 的目录，SDK 中和 image_fix_xxx.tar 同一级目录还有一个名为 firmwarm_xxx.tar 文件，把 firmwarm_xxx.tar 解压后可以看到有一个 firmwarm_xxx.bin 文件，即最新的 pbf 映像，将该文件拷贝到 image_fix 目录下，执行如下命令进入此目录：

```
# sudo cd image_fix
```

并执行以下命令完成 pbf 更新映像：

```
# sudo ./my_scripts/pbf_update.sh firmwarm_xxx.bin
```

4.3 设置软连接

工具 image_fix 解压后的目录中有两个软链接，分别是 bl32_new.bin 和 bl33_new.bin，应分别指向 PBTEE 的映像和 uefi 的映像，把 PBTEE（3.3 中的 tee-phytium-xxxx.bin）和 uefi（uefi-xxx.bin，由固件部门提供，这里作为案例）上传至 image_fix 目录下，接着在此目录下执行如下命令建立软链接：

```
# sudo ln -sf tee-phytium-xxxx.bin bl32_new.bin
# sudo ln -sf uefi-xxx.bin bl33_new.bin
```

注意：如果是打包 uboot 固件，只需要上传 uboot 的映像（uboot-xxx.bin）至 image_fix 目录下，并将 bl33_new.bin 指向它。

4.4 参数配置

执行如下命令配置参数：

```
# sudo ./my_scripts/fix_parameter.sh
```

这里选择将“dev interface”中的 qspi 频率调整到 4.6MHz，另外将“enable pcie”中的 PEU0 和 PEU1 的 split mode 修改成“x8x8”（与硬件的 pcie 拆分模式保持一致），如果已经是该配置，则不需要修改。

4.5 打包

执行以下命令进行打包：

```
# sudo ./my_scripts/image-fix.sh bl32
```

命令执行完成后，目录下新生成的 fip-all.bin 就是最终烧录到 spi flash 中的二进制 bin 文件。

注意：默认是 ARM 的打包环境，如果是 x86 的打包环境，my_scripts 目录中的对应工具链接需要改成 x86 的版本

5 升级内核

内核版本最好高于 4.12.0，否则需要手动添加 tee 驱动模块。

目前飞腾补丁基于 4.19.x 系列内核开发，下面以 4.19.7（最新版本）内核为例来说明升级内核的方法。

FT2000/4 桌面机安装完 Ubuntu20.04 系统后缺少网卡驱动，无法联网，而编译内核需要联网安装相应的工具包，因此编译内核最好在其他有网的环境下进行，可以把升级的内核打成 deb 包，方便移植。

编译内核需要准备内核源码、补丁、配置文件。

内核源码可以从内核官网或者国内各大开源镜像站下载。内核官网下载地址：https://mirrors.edge.kernel.org/pub/linux/kernel/v4.x/linux-4.19.7.tar.xz 。清华源内核下载地址：

https://mirrors.tuna.tsinghua.edu.cn/kernel/v4.x/linux-4.19.7.tar.xz。

patch-phytium-4.19.7

内核补丁：patch-phytium-4.19.7

ft2004-config

配置文件：ft2004-config

5.1 Uefi 启动方式

Uefi 启动方式需要 ACPI 表对 optee 进行描述，如下图 5.1-1（类似）所示：

```
Scope (_SB)
{
    Device (OPTE)
    {
        Name (_HID, "PHYT8003")  // _HID: Hardware ID
        Name (_CID, "FTOP0001")  // _CID: Compatible ID
        Name (_UID, Zero)  // _UID: Unique ID
        Name (_DSD, Package (0x02)  // _DSD: Device-Specific Data
        {
            ToUUID ("daffd814-6eba-4d8c-8a91-bc9bbf4aa301") /* Device Properties for _DSD */,
            Package (0x02)
            {
                Package (0x02)
                {
                    "compatible",
                    "linaro,optee-tz"
                },

                Package (0x02)
                {
                    "method",
                    "smc"
                }
            }
        })
    }
}
```

图 5.1-1 ACPI 表对 optee 的描述

ACPI 表（二进制文件）是 /sys/firmware/acpi/tables/DSDT 文件，将其拷贝到其他任意目录，接着反编

译查看源文件，执行如下命令:

```
# sudo apt install acpica-tools
# sudo cp -a /sys/firmware/acpi/tables/DSDT /
# sudo iasl -da -dl /DSDT
```

命令执行完后，会在根目录生成 DSDT.dsl 源文件，查看 DSDT.dsl 文件验证 ACPI 表是否对 optee 作了描述。

5.1.1 安装打补丁所需的工具包

```
# apt install patch
```

5.1.2 安装编译内核所需的工具包

```
# apt install gcc
# apt install libncurses-dev
# apt install bison
# apt install flex
# apt install libssl-dev
# apt install openssl
# apt install make
```

5.1.3 上传内核源码、补丁、配置文件

如下图 5.1.3-1 所示，上传内核源码、补丁、配置文件，并置于同一目录。

```
root@test:/opt# ls
ft2004-config  linux-4.19.7.tar.xz  patch-phytium-4.19.7
```

图 5.1.3-1 内核源码、补丁和配置文件

5.1.4 解压内核源码

```
# tar -xvf linux-4.19.7.tar.xz
```

解压后当前目录文件如下图 5.1.4-1 所示

```
root@test:/opt# ls
ft2004-config  linux-4.19.7  linux-4.19.7.tar.xz  patch-phytium-4.19.7
```

图 5.1.4-1 当前目录文件

5.1.5 进入内核源码目录

```
# cd linux-4.19.7
```

5.1.6 打内核补丁

```
# patch -p1 < ../patch-phytium-4.19.7
```

5.1.7 生成配置文件

```
# cp -a ../ft2004-config .config
```

5.1.8 打开 tee 配置并解决内核板块依赖

```
# make menuconfig
```

命令执行后，如下图 5.1.8-1 所示

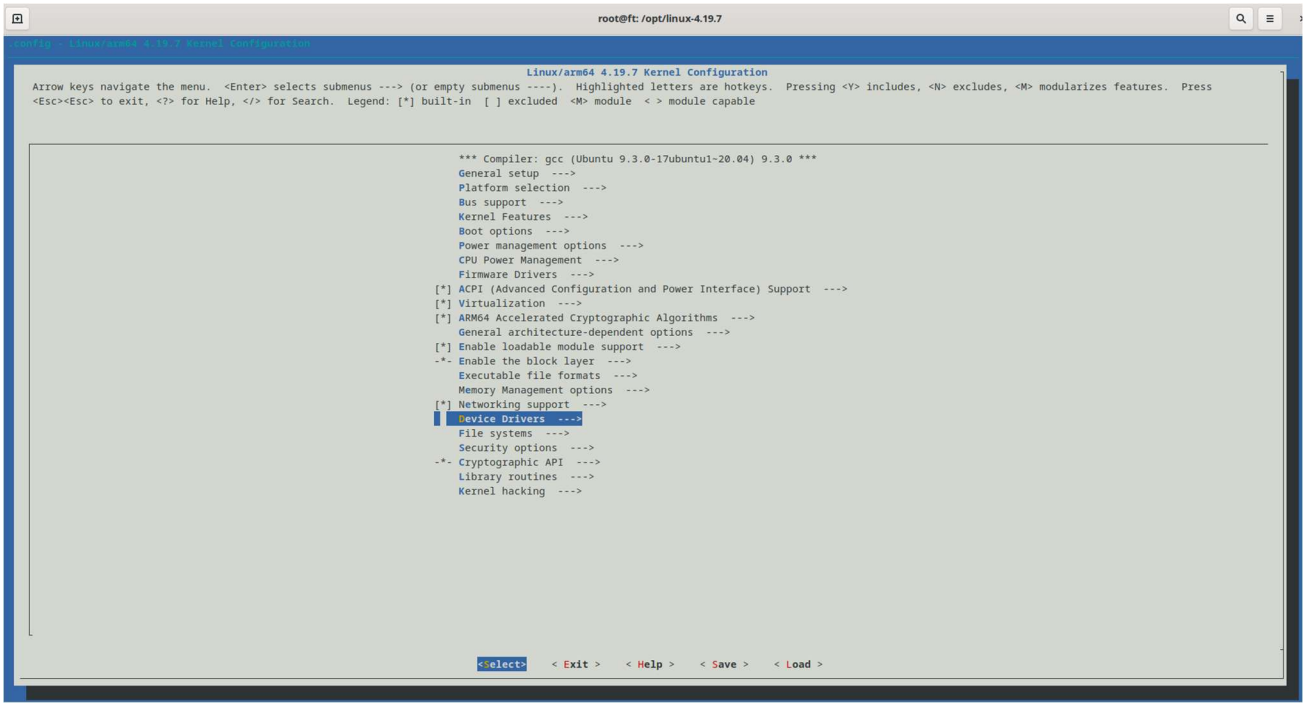


图 5.1.8-1　图形化设置配置文件界面

选择“Device Drivers --->”后回车进入到设备驱动配置页面，如下图 5.1.8-2 所示：

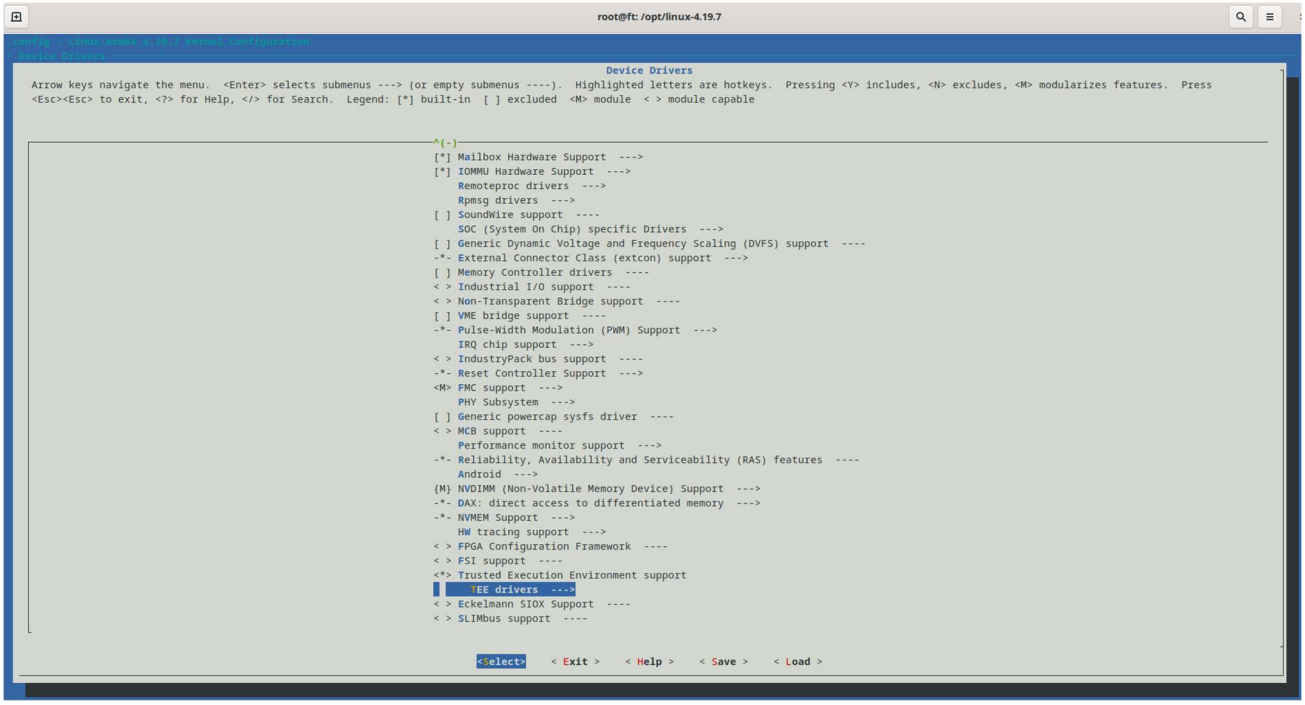


图 5.1.8-2 内核设备驱动配置页面

接着选择”TEE drivers --->“后回车进入到 TEE 驱动的配置页面，修改后的配置如下图 5.1.8-3 所示：

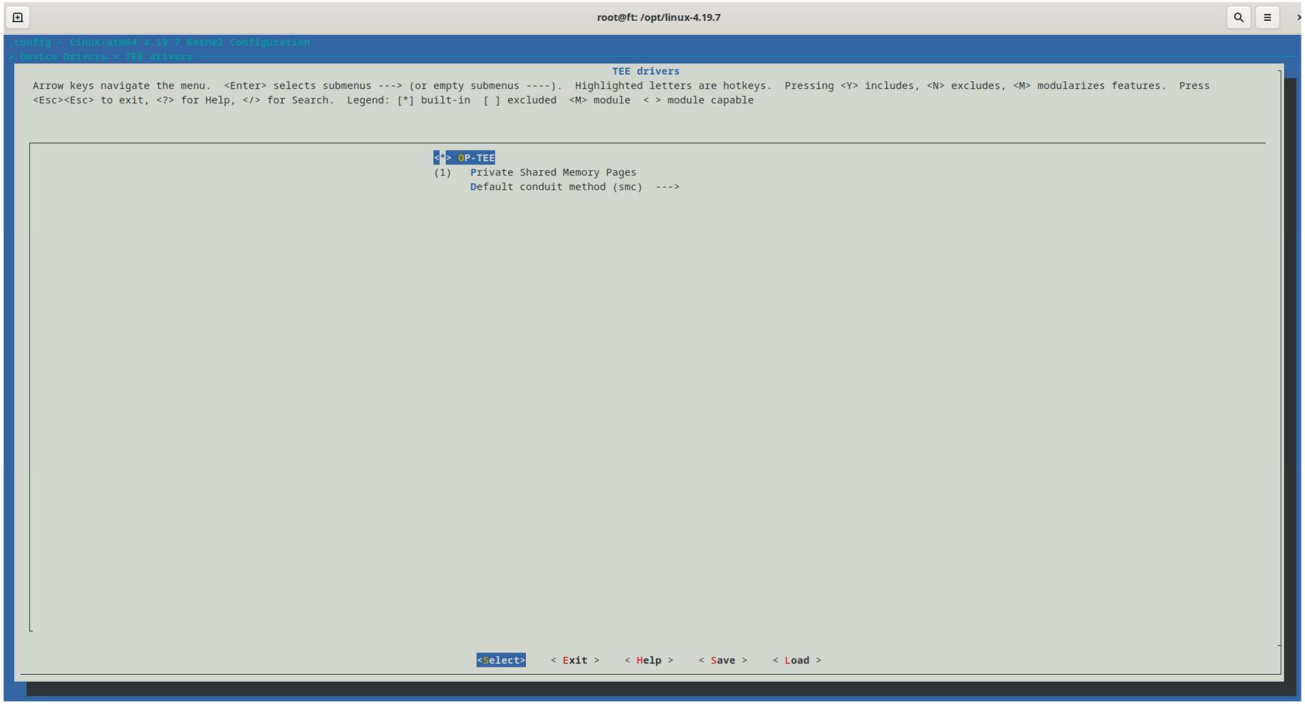


图 5.1.8-3 TEE 驱动配置页面

选择”< Save >”保存配置后，再选择“<Exit>”退出。

5.1.9 编译内核

```
# make bindeb-pkg -j4
```

执行完成之后，执行如下命令退回到上一级目录（内核源码目录）。

```
# sudo cd ..
```

执行如下命令

```
# sudo ls
```

可以查看当前目录下的文件列表，如下图 5.1.9-1 所示：

```
root@ft:/opt# ls
ft2004-config  linux-4.19.7_4.19.7-1_arm64.buildinfo  linux-4.19.7.tar.xz                  linux-image-4.19.7_4.19.7-1_arm64.deb      linux-libc-dev_4.19.7-1_arm64.deb
linux-4.19.7   linux-4.19.7_4.19.7-1_arm64.changes    linux-headers-4.19.7_4.19.7-1_arm64.deb  linux-image-4.19.7-dbg_4.19.7-1_arm64.deb  patch-phytium-4.19.7
```

图 5.1.9-1 当前目录文件列表

其中 linux-image-4.19.7_4.19.7-1_arm64.deb 和 linux-headers-4.19.7_4.19.7-1_arm64.deb 是升级内核需要用到的 deb 包，将这两个 deb 包上传到 FT2000/4 上的 ubuntu20.04 系统任意目录下，并在此目录下执行如下命令升级内核：

```
# sudo dpkg -i linux-image-4.19.7_4.19.7-1_arm64.deb
# sudo dpkg -i linux-headers-4.19.7_4.19.7-1_arm64.deb
```

5.1.10 修改默认启动内核

升级内核后，FT2000/4 上的 ubuntu20.04 系统默认启动内核仍是系统自带内核，需要做如下配置：

执行如下命令修改 grub 文件：

```
# sudo vim /etc/default/grub
```

只需修改 GRUB_DEFAULT（默认启动内核）、注释 GRUB_TIMEOUT_STYLE=hidden（取消隐藏 grub 菜单）、GRUB_TIMEOUT（grub 菜单显示时间[s]），如下图 5.1.10-1 所示：

```
GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 4.19.7"
# GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=10
```

图 5.1.10-1 /etc/default/grub 文件需要改动的部分（修改后）

再执行如下命令更新 grub：

```
# sudo update-grub
```

5.2 Uboot 启动方式

Uboot 启动方式需要设备树对 optee 进行描述，如下图 5.2-1（类似）所示：

```
firmware{
        optee{
            compatible = "linaro,optee-tz";
            method = "smc";
        };
};
```

图 5.2-1 设备树对 optee 的描述

设备树（二进制文件）一般在系统下/boot/dtb 目录下（这里假设为/boot/dtb/test.dtb）,可以通过反编译来查看设备树源文件，执行如下命令:

```
# sudo cd /boot/dtb
# sudo dtc -I dtb -O dts test.dtb > test.dts
```

命令执行完后，会在当前目录生成 test.dts 设备树源文件，查看 test.dts 文件验证设备树是否对 optee 作了描述。

5.2.1 安装打补丁所需的工具包（同 5.1.1）

5.2.2 安装编译内核所需的工具包（同 5.1.2）

5.2.3 上传内核源码、补丁、配置文件（同 5.1.3）

5.2.4 解压内核源码（同 5.1.4）

5.2.5 进入内核源码目录（同 5.1.5）

5.2.6 打内核补丁（同 5.1.6）

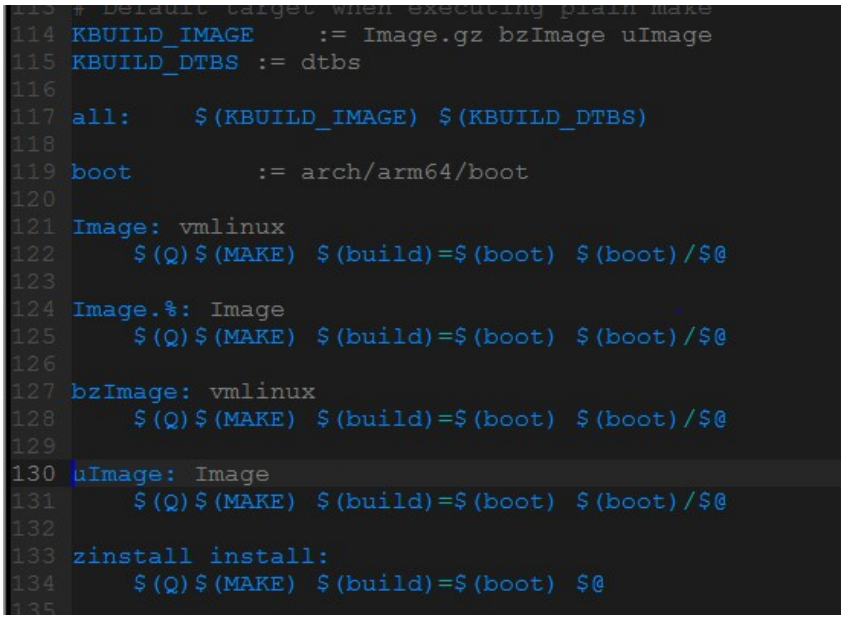
5.2.7 内核修改

修改 ulmage 加载地址，修改 arch/arm64/boot/Makefile，如下图 5.2.7-1 所示: （文件图示以外部分保持不变）

```
OBJCOPYFLAGS_Image :=-O binary -R .note -R .note.gnu.build-id -R .comment -S

UIMAGE_LOADADDR=0x80080000

check_for_multiple_loadaddr = \
if [ $(words $(UIMAGE_LOADADDR)) -ne 1 ]; then \
        echo 'multiple (or no) load addresses: $(UIMAGE_LOADADDR)'; \
        echo 'This is incompatible with uImages'; \
        echo 'Specify LOADADDR on the commandline to build an uImage'; \
        false; \
fi

targets := Image Image.gz bzImage uImage

$(obj)/Image: vmlinux FORCE
    $(call if_changed,objcopy)

$(obj)/bzImage: vmlinux FORCE
    $(call if_changed,objcopy)

$(obj)/Image.bz2: $(obj)/Image FORCE
    $(call if_changed,bzip2)

$(obj)/Image.gz: $(obj)/Image FORCE
    $(call if_changed,gzip)

$(obj)/Image.lz4: $(obj)/Image FORCE
    $(call if_changed,lz4)

$(obj)/Image.lzma: $(obj)/Image FORCE
    $(call if_changed,lzma)

$(obj)/Image.lzo: $(obj)/Image FORCE
    $(call if_changed,lzo)

$(obj)/uImage: $(obj)/Image FORCE
    @$(check_for_multiple_loadaddr)
    $(call if_changed,uimage)
```

图 5.2.7-1 arch/arm64/boot/Makefile 修正图

修改 arch/arm64/Makefie 如下图 5.2.7-2 所示，修改完这两个 Makefile 就可以生成 uboot 头的 ulmage 文件了。



```
114 KBUILD_IMAGE     := Image.gz bzImage uImage
115 KBUILD_DTBS := dtbs
116
117 all:     $(KBUILD_IMAGE) $(KBUILD_DTBS)
118
119 boot         := arch/arm64/boot
120
121 Image: vmlinux
122     $(Q)$(MAKE) $(build)=$(boot) $(boot)/$@
123
124 Image.%: Image
125     $(Q)$(MAKE) $(build)=$(boot) $(boot)/$@
126
127 bzImage: vmlinux
128     $(Q)$(MAKE) $(build)=$(boot) $(boot)/$@
129
130 uImage: Image
131     $(Q)$(MAKE) $(build)=$(boot) $(boot)/$@
132
133 zinstall install:
134     $(Q)$(MAKE) $(build)=$(boot) $@
```

图 5.2.7-2 arch/arm64/Makefie 修正图

5.2.8 打开 tee 配置并解决内核板块依赖（同 5.1.8）

5.2.9 编译内核

执行如下命令编译内核:

```
# make -j4
```

4 是代表 4 线程编译，make 可以一次性编译内核、设备树和模块。（最好执行两次，确认编译没有报错）

若单独编译内核 uImage，则执行如下命令:

```
# make uImage -j4
```

若单独编译设备树，则执行如下命令:

```
# make dtbs -j4
```

编译好的 uImage 在 arch/arm64/boot 目录下，编译好的 dtb 文件在 arch/arm64/boot/dts/phytium/目录下。（ft2004-devboard-d4-dsk.dtb）

5.2.10 安装内核模块

```
# make modules_install -j4
```

是把编译好的模块拷贝到本地系统目录下（一般是/lib/modules/）。

5.2.11 系统盘分区

假设本地磁盘为/dev/sda（编译内核的盘），系统盘为/dev/sdb，先对系统盘进行分区，执行如下命令:

```
# fdisk /dev/sdb
```

效果如下图 5.2.11-1 所示

```
[root@localhost ~]# fdisk /dev/sdb

欢迎使用 fdisk (util-linux 2.32.1)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。


命令(输入 m 获取帮助): o
创建了一个磁盘标识符为 0x95fd15b9 的新 DOS 磁盘标签。

命令(输入 m 获取帮助): n
分区类型
   p   主分区 (0个主分区，0个扩展分区，4空闲)
   e   扩展分区 (逻辑分区容器)
选择 (默认 p):

将使用默认回应 p。
分区号 (1-4，默认  1):
第一个扇区 (2048-3907029167，默认 2048):
上个扇区，+sectors 或 +size{K,M,G,T,P} (2048-3907029167，默认 3907029167): +5G

创建了一个新分区 1，类型为 “Linux”，大小为 5 GiB。
分区 #1 包含一个 ext4 签名。

您想移除该签名吗？是[Y]/否[N]: y

写入命令将移除该签名。

命令(输入 m 获取帮助): n
分区类型
   p   主分区 (1个主分区，0个扩展分区，3空闲)
   e   扩展分区 (逻辑分区容器)
选择 (默认 p):

将使用默认回应 p。
分区号 (2-4，默认  2):
第一个扇区 (10487808-3907029167，默认 10487808):
上个扇区，+sectors 或 +size{K,M,G,T,P} (10487808-3907029167，默认 3907029167): +500G

创建了一个新分区 2，类型为 “Linux”，大小为 500 GiB。

命令(输入 m 获取帮助): w
分区表已调整。
正在同步磁盘。
```

图 5.2.11-1 使用 fdisk 命令对系统盘分区

5.2.12 创建文件系统

分别给系统盘的两个分区建立文件系统（一般为 ext4），执行如下命令：

```
# mkfs.ext4 /dev/sdb1
```

```
# mkfs.ext4 /dev/sdb2
```

5.2.13 boot 分区制作

创建两个文件夹，挂载系统盘的两个分区，执行如下命令:

```
# mkdir /mnt/sdb1
# mkdir /mnt/sdb2
# mount /dev/sdb1 /mnt/sdb1
# mount /dev/sdb2 /mnt/sdb2
```

创建用于存放设备树的文件夹，执行如下命令:

```
# mkdir /mnt/sdb1/dtb
```

拷贝设备树到系统盘，执行如下命令:

```
# cp /opt/linux-4.19.5/arch/arm64/boot/dts/phytium/d2000-devboard-dsk.dtb /mnt/sdb1/dtb/
```

拷贝 uImage 镜像到系统盘，执行如下命令:

```
# cp /opt/linux-4.19.5/arch/arm64/boot/uImage /mnt/sdb1/dtb/
```

5.2.14 拷贝生成系统盘的根文件系统

执行如下命令:

```
# cp -a /opt/squashfs-root/LiveOS/rootfs/* /mnt/sdb2/
```

5.2.15 内核模块移植

在本地系统的/lib/modules 目录下会生成名为 4.19.7 的子目录，拷贝该子目录到系统盘 root 分区中的/lib/modules 目录下（如果没有 modules 目录请手动创建），执行如下命令:

```
# cp -a /lib/modules/4.19.7 /mnt/sdb2/lib/modules/
```

因为此 rootfs 默认启动方式为 anaconda 启动，uboot 引导进入 anaconda 模式后会直接卡死，需要删除/etc/systemd/system/default.target，执行如下命令:

```
# rm -rf /mnt/sdb2/etc/systemd/system/default.target
```

创建软连接，执行如下命令:

```
# ln -s /mnt/sdb2/usr/lib/systemd/system/multi-user.target
/mnt/sdb2/etc/systemd/system/default.target
```

5.2.16 修改默认启动内核（同 5.1.10）

至此，系统盘已经做好，把 FT2000/4 主板上的固件换成带有 tee 模块的 uboot 固件（注意固件描述的设备树与内核信息，可能要改动设备树与内核的名字与位置，否则可能会提示找不到设备树或内核），系统盘就能正常启动。

6 验证

6.1 检查系统设备文件

如果前面步骤都操作正确，则系统下一定会有/dev/tee0 和/dev/teepriv0 两个设备文件。

6.2 运行 helloworld 演示程序

将 3.3 生成的 data-xxxx.tgz 上传至系统根目录，执行如下命令解压并进入/data 目录：

```
# sudo tar -zxvf data-xxxx.tgz
# sudo cd data
```

接着执行如下命令设置环境：

```
# sudo source ./env.source
```

进入 bin 子目录：

```
# sudo cd bin
```

执行 hello_world 程序：

```
# sudo ./hello_world
```

程序执行结果如下图 6.2-1 所示：

```
D/TC:? 0 tee_ta_init_pseudo_ta_session:299 Lookup pseudo TA 8aaaf200-2450-11e4-abe2-0002a5d5c51b
D/TC:? 0 ldelf_load_ldelf:91 ldelf load address 0x40006000
D/LD:  ldelf:134 Loading TA 8aaaf200-2450-11e4-abe2-0002a5d5c51b
F/TC:? 0 trace_syscall:151 syscall #3 (syscall_get_property)
F/TC:? 0 trace_syscall:151 syscall #5 (syscall_open_ta_session)
D/TC:? 0 ldelf_syscall_open_bin:140 Lookup user TA ELF 8aaaf200-2450-11e4-abe2-0002a5d5c51b (Secure Storage TA)
```

图 6.2-1 helloworld 运行结果截图（部分）

如果程序运行没有报错，并有上图 6.2-1 类似显示，就说明 TEE 环境搭建成功了。

补充说明：

这个演示程序是 REE 环境下发了一个值为 42 的变量给了 TEE 环境，TEE 环境收到后，进行了自增操作，然后返回给 REE 环境。具体可以参考源代码 hello_world.c（REE 环境运行）和 hello_world_ta.c（TEE 环境运行）,代码在 pbtee/ree_ca/hello_world 目录下。用户可以参考 helloworld 的模式开发客户自己的 CA（Trusted Application）和 TA（Client Application）程序。也可以参考 ree_ca 目录下提供的其他开发例子。

7 常见问题

7.1 TEE 环境搭建失败（系统下没有生成/dev/tee0 和/dev/teepriv0 两个文件）

首先排查固件的问题，通过串口分析启动日志，出现以下日志则说明固件环境是正常的，如下图 7.1-1 所示：

```
E/TC:0   console_init:273 enter c code(pl011)
I/TC: PBTEE 3.0.0 (10:58:40 Sep 23 2021)
NOTICE:  Phytium System Service Call: 0x82000001
E/TC:0   console_init:280 PBF version : 10054
D/TC:0   get_aslr_seed:1380 Warning: no ASLR seed
D/TC:0   add_phys_mem:557 VCORE_UNPG_RX_PA type TEE_RAM_RX 0xfc000000 size 0x00096000
D/TC:0   add_phys_mem:557 VCORE_UNPG_RW_PA type TEE_RAM_RW 0xfc096000 size 0x0036a000
D/TC:0   add_phys_mem:557 TA_RAM_START type TA_RAM 0xfc400000 size 0x01e00000
D/TC:0   add_phys_mem:557 TEE_SHMEM_START type NSEC_SHM 0xfe200000 size 0x00c00000
D/TC:0   add_phys_mem:557 ROUNDDOWN(0x28001000, CORE_MMU_PGDIR_SIZE) type IO_NSEC 0x28000000 size 0x00200000
D/TC:0   add_phys_mem:557 ROUNDDOWN(0x29900000, CORE_MMU_PGDIR_SIZE) type IO_SEC 0x29800000 size 0x00600000
D/TC:0   add_phys_mem:557 ROUNDDOWN(0x28220000, CORE_MMU_PGDIR_SIZE) type IO_SEC 0x28200000 size 0x00200000
D/TC:0   add_phys_mem:557 ROUNDDOWN(((0x29800000) + 0x5000UL), CORE_MMU_PGDIR_SIZE) type IO_SEC 0x29800000 size 0x00200000
D/TC:0   add_phys_mem:571 Physical mem map overlaps 0x29800000
D/TC:0   add_va_space:597 type RES_VASPACE size 0x00a00000
D/TC:0   add_va_space:597 type SHM_VASPACE size 0x02000000
D/TC:0   dump_mmap_table:704 type NSEC_SHM     va 0xf5800000..0xf63fffff pa 0xfe200000..0xfedfffff size 0x00c00000 (pgdir)
D/TC:0   dump_mmap_table:704 type TA_RAM       va 0xf6600000..0xf83fffff pa 0xfc400000..0xfe1fffff size 0x01e00000 (pgdir)
D/TC:0   dump_mmap_table:704 type IO_SEC       va 0xf8600000..0xf8bfffff pa 0x29800000..0x29dfffff size 0x00600000 (pgdir)
D/TC:0   dump_mmap_table:704 type IO_SEC       va 0xf8e00000..0xf8ffffff pa 0x28200000..0x283fffff size 0x00200000 (pgdir)
D/TC:0   dump_mmap_table:704 type IO_NSEC      va 0xf9000000..0xf91fffff pa 0x28000000..0x281fffff size 0x00200000 (pgdir)
D/TC:0   dump_mmap_table:704 type RES_VASPACE  va 0xf9200000..0xf9bfffff pa 0x00000000..0x009fffff size 0x00a00000 (pgdir)
```

图 7.1-1 串口有关 tee 的信息截图

日志内容可能会有些差异，但只要有类似“E/TC”，“I/TC”这样标志性信息即可。

如果没有类似图 7.1-1 的打印，则可以从 pbtee 的编译（编译是否有报错）、ACPI 或设备树（是否有 TEE 模块的描述）、固件打包（pbf 是否需要更新）几方面入手。

其次再排查内核的问题（内核配置是否正确、编译有没有报错）。