

实验七 Linux 内核移植

【实验目的】

掌握 Linux 内核配置和编译的基本方法

【实验环境】

- 1、ubuntu 14.04 发行版
- 2、FS4412 实验平台
- 3、交叉编译工具：arm-none-linux-gnueabi-

【注意事项】

- 1、实验步骤中以“\$”开头的命令表示在 ubuntu 环境下执行，以“#”开头的命令表示在开发板下执行

【实验步骤】

- 1、在 Linux 官网下载 Linux 内核源码（这里我们下载 linux-3.14.tar.xz）

<https://mirrors.edge.kernel.org/pub/linux/kernel/v3.x/>

- 2、拷贝内核源码包到 ubuntu 的家目录下，解压并进入其顶层目录

```
$ tar xvf linux-3.14.tar.xz
```

```
$ cd linux-3.14
```

- 3、源码并不知道我们的处理器架构及交叉编译工具是什么，我们自己在 Makefile 中指定

```
$ vi Makefile
```

将

```
ARCH      ?= $(SUBARCH)
```

```
CROSS_COMPILE ?= $(CONFIG_CROSS_COMPILE:"%"=%)
```

修改为以下内容（注意后边不要有多余空格），然后保存退出

```
ARCH      ?= arm
```

```
CROSS_COMPILE ?= arm-none-linux-gnueabi-
```

- 4、指定使用的处理器

```
$ make exynos_defconfig
```

显示如下信息表示配置成功

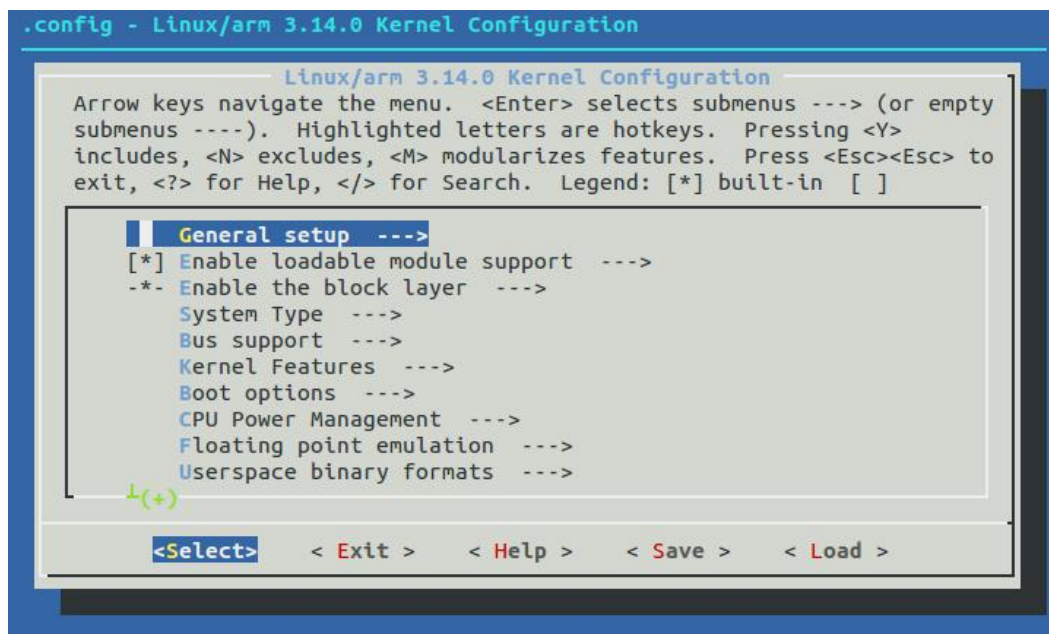
```
linux@linux:~/linux-3.14$ make exynos_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/zconf.lex.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
#
# configuration written to .config
#
```

5、进入内核配置界面

```
$ make menuconfig
```

弹出如下图形化配置界面，在该界面下我们可以对 linux 进行进一步的修改和配置

方向键可选择不同的选项，‘Enter’键进入子菜单，‘Y’键选中某项功能，‘N’键去除某项功能，‘M’键将该功能编译成内核模块，两次‘Esc’键退出界面，‘?’键为帮助选项，‘/’键为搜索选项



注 1：若显示如下信息，是因为 ubuntu 上没有安装对应的图形库

```
linux@linux:~/linux-3.14$ make menuconfig
*** Unable to find the ncurses libraries or the
*** required header files.
*** 'make menuconfig' requires the ncurses libraries.
***
*** Install ncurses (ncurses-devel) and try again.
***
make[1]: *** [scripts/kconfig/dochecklxdialog] Error 1
make: *** [menuconfig] Error 2
```

执行如下命令安装对应的图形库，然后重新执行 make menuconfig 即可

```
$ sudo apt-get install libncurses5-dev
```

注 2: 若显示如下信息, 是因为终端窗口太小, 需将终端最大化后再执行

```
linux@linux:~/linux-3.14$ make menuconfig
scripts/kconfig/mconf Kconfig
Your display is too small to run Menuconfig
!
It must be at least 19 lines by 80 columns.
make[1]: *** [menuconfig] Error 1
make: *** [menuconfig] Error 2
```

6、配置内核

将 ‘System Type’ 菜单下的 ‘S3C UART...’ 修改为 2 (即使用 UART2)

System Type --->

(2) S3C UART to use for low-level messages

设置完成后通过方向键选择 ‘Save’ 保存即可, 然后选择 ‘Exit’ 退出该配置界面

7、编译内核 (该过程可能需要二十分钟左右)

```
$ make uImage
```

显示如下信息表示编译成功, 即在源码的 arch/arm/boot/目录下生成了 uImage 镜像

```
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
UIMAGE arch/arm/boot/uImage
Image Name: Linux-3.14.0
Created: Sun Apr 12 11:43:59 2020
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 2756512 Bytes = 2691.91 kB = 2.63 MB
Load Address: 40008000
Entry Point: 40008000
```

注: 如图所示, 第一次在 ubuntu 上编译 Linux 内核会提示缺少一个 mkimage 命令

```
"mkimage" command not found - U-Boot images will not be built
make[1]: *** [arch/arm/boot/uImage] Error 1
make: *** [uImage] Error 2
```

该命令可在 uboot 源码中 u-boot-2013.01/tools/目录下获取 (必须是编译后的 uboot)

将该命令拷贝到 ubuntu 的 /usr/bin 目录下即可正确编译内核

```
$ sudo cp u-boot-2013.01/tools/mkimage /usr/bin/
```

给该命令添加可执行权限

```
$ sudo chmod 777 /usr/bin/mkimage
```

完成后回到内核的顶层目录下重新编译内核即可

8、编译设备树

内核源码中并没有 fs4412 平台的设备树文件, 这里我们从源码支持的平台中找一个硬件与我们最类似的, 在其基础上进行修改, 这里我们参考的是 samsung 公司的 origen

拷贝 origen 的设备树并将其重命名

```
$ cp arch/arm/boot/dts/exynos4412-origen.dts arch/arm/boot/dts/exynos4412-fs4412.dts
```

因为添加的设备树文件也要编译，所以对应的 Makefile 也要修改

```
$ vi arch/arm/boot/dts/Makefile
```

在

```
exynos4412-origen.dtb \
```

后添加如下内容，然后保存退出

```
exynos4412-fs4412.dtb \
```

回到源码的顶层目录下编译设备树

```
$ make dtbs
```

显示如下信息表示编译成功，即在 arch/arm/boot/dts/目录下生成了 exynos4412-fs4412.dtb

```
linux@linux:~/linux-3.14$ make dtbs
DTC  arch/arm/boot/dts/exynos4210-origen.dtb
DTC  arch/arm/boot/dts/exynos4210-smdkv310.dtb
DTC  arch/arm/boot/dts/exynos4210-trats.dtb
DTC  arch/arm/boot/dts/exynos4210-universal_c210.dtb
DTC  arch/arm/boot/dts/exynos4412-odroidx.dtb
DTC  arch/arm/boot/dts/exynos4412-origen.dtb
DTC  arch/arm/boot/dts/exynos4412-fs4412.dtb
DTC  arch/arm/boot/dts/exynos4412-smdk4412.dtb
```

9、测试内核和设备树

将编译生成的内核和设备树拷贝到 tftp 的工作目录

```
$ sudo cp arch/arm/boot/uImage /tftpboot
```

```
$ sudo cp arch/arm/boot/dts/exynos4412-fs4412.dtb /tftpboot/
```

```
$ sudo chmod 777 /tftpboot/*
```

设置 uboot 的启动参数并保存

(参之前)

```
# setenv ipaddr ***.***.***.***
```

```
# setenv serverip xxx.xxx.xxx.xxx
```

```
# setenv bootcmd tftp 41000000 uImage;tftp 42000000 exynos4412-fs4412.dtb;bootm 41000000 - 42000000
```

```
# setenv bootargs root=/dev/nfs nfsroot=xxx.xxx.xxx.xxx:/opt/4412/rootfs/ rw console=ttySAC2,115200 init=/linuxrc ip=***.***.***.***
```

```
# saveenv
```

注 1: xxx.xxx.xxx.xxx 为 ubuntu 主机的 ip, ***.***.***.*** 为开发板的 ip, 必须和 ubuntu 主机的 ip 在同一个网段 (根据自己电脑情况进行设置)

注 2: 以上设置手动输入, 命令粘贴可能会有中文符号

回到 ubuntu 重启 tftp 和 nfs 服务器

```
$ sudo service tftpd-hpa restart
```

```
$ sudo service nfs-kernel-server restart
```

重启开发板查看现象, 如图所示, 内核在启动到一半时会崩溃卡死, 原因在于我们在该实验中只是对 UART 进行了配置, 而其他功能都保持默认选项, 内核默认配置中没选配我们使用的网卡驱动、nfs 等功能, 所以在挂载根文件系统时导致内核崩溃, 所以后续我们还需要配置网卡驱动、nfs 等

```
rne1+0x368/0x3c4)
[ 1.630000] [<c04bdaf4>] (start_kernel) from [<40008074>] (0x40008074)
[ 1.630000] CPU1: stopping
[ 1.630000] CPU: 1 PID: 0 Comm: swapper/1 Not tainted 3.14.0 #1
[ 1.630000] [<c0013e10>] (unwind_backtrace) from [<c0011240>] (show_stack+0x10/0x14)
[ 1.630000] [<c0011240>] (show_stack) from [<c037dedc>] (dump_stack+0x64/0xb4)
[ 1.630000] [<c037dedc>] (dump_stack) from [<c00131bc>] (handle_IPI+0x154/0x180)
[ 1.630000] [<c00131bc>] (handle_IPI) from [<c0008568>] (gic_handle_irq+0x60/0x68)
[ 1.630000] [<c0008568>] (gic_handle_irq) from [<c0011d40>] (__irq_svc+0x40/0x70)
[ 1.630000] Exception stack(0xee8c9fa0 to 0xee8c9fe8)
[ 1.630000] 9fa0: 00000001 00000000 0000112a 00000000 ee8c8000 c04fe494 c0385b24 c052ed19
[ 1.630000] 9fc0: c052ed19 413fc090 00000001 00000000 0000001b ee8c9fe8 c000eff4 c000eff8
[ 1.630000] 9fe0: 60000153 ffffffff
[ 1.630000] [<c0011d40>] (__irq_svc) from [<c000eff8>] (arch_cpu_idle+0x28/0x30)
[ 1.630000] [<c000eff8>] (arch_cpu_idle) from [<c00589b0>] (cpu_startup_entry+0x9c/0x138)
[ 1.630000] [<c00589b0>] (cpu_startup_entry) from [<40008604>] (0x40008604)
```