

## 实验 03 Menuconfig\_Kconfig

### 3.1 本章导读

Linux 驱动工程师一定要掌握 Linux 内核的编译方法，也就是将 Linux 内核源码，编译成针对特定硬件的二进制镜像。

在前面入门视频“01-烧写、编译以及基础知识视频”→“实验 10-搭建编译环境 uboot\_linux\_Android”中，简单的介绍过如何将 Linux 源码编译生成二进制 zImage。

在本章中，将更加详细的介绍这部分内容，然后介绍 Kconfig 配置文件，Kconfig 文件是和编译的 Makemenuconfig 工具配合使用的。最后还需要掌握“.config”文件的作用。

#### 3.1.1 工具

##### 3.1.1.1 硬件工具

- 1) PC 机

##### 3.1.1.2 软件工具

- 1) 虚拟机 Vmware
- 2) Ubuntu12.04.2
- 3) Ubuntu 系统下解压生成的 Linux 源码

#### 3.1.2 预备课程

入门视频 “01-烧写、编译以及基础知识视频” → “实验 10-搭建编译环境 uboot\_linux\_Android” 或者使用手册 “五 Android 开发环境搭建以及编译”

### 3.1.3 视频资源

本节配套视频为 “视频 03\_Menuconfig\_Kconfig”

## 3.2 学习目标

本章需要学习以下内容：

掌握 Menuconfig 的用法

理解 Kconfig 文件并掌握修改 Kconfig 的方法

理解配置文件 “.config”

Linux 内核配置裁减

## 3.3 Linux 内核配置系统

Linux 内核配置系统由三个部分组成。

Makefile 文件：分布在 Linux 内核源码中的 Makefile 文件，定义了 Linux 内核的编译规则。

Kconfig 文件：给用户提供配置选择的功能。

配置工具：这里使用的是 menuconfig，相比其它工具，这个工具使用的比较多，也比较容易上手，无论哪个机构发布的 Linux 版本应该都是支持 menuconfig 的。

## 3.4 Menuconfig 的操作

Linux 的裁减配置是通过 menuconfig 工具来实现的，本节介绍如何使用这个工具。

### 3.4.1 Menuconfig 发展历史简介

在 Linux 发展过程中，配置内核可以使用以下工具。

```
#make config
```

这是基于文本的最为传统的配置界面，不推荐使用

```
#make menuconfig
```

基于文本菜单的配置界面，现在大部分都是使用这个工具来裁减配置内核的，本章节也是介绍这种方法。

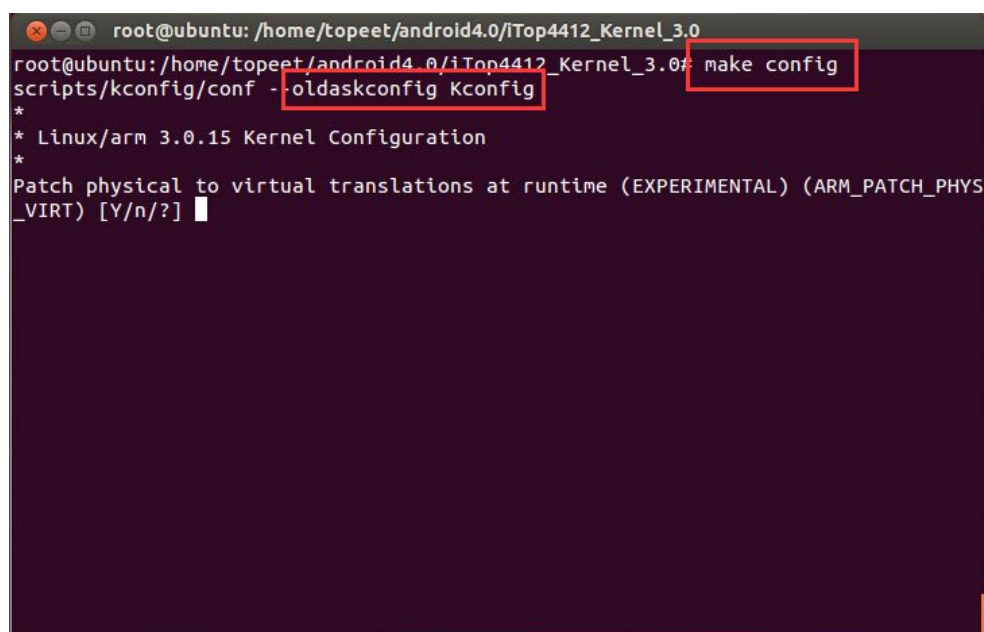
```
#make xconfig
```

要求 QT 被安装，用的比较少。

```
#make gconfig
```

要求 GTK，用的比较少。

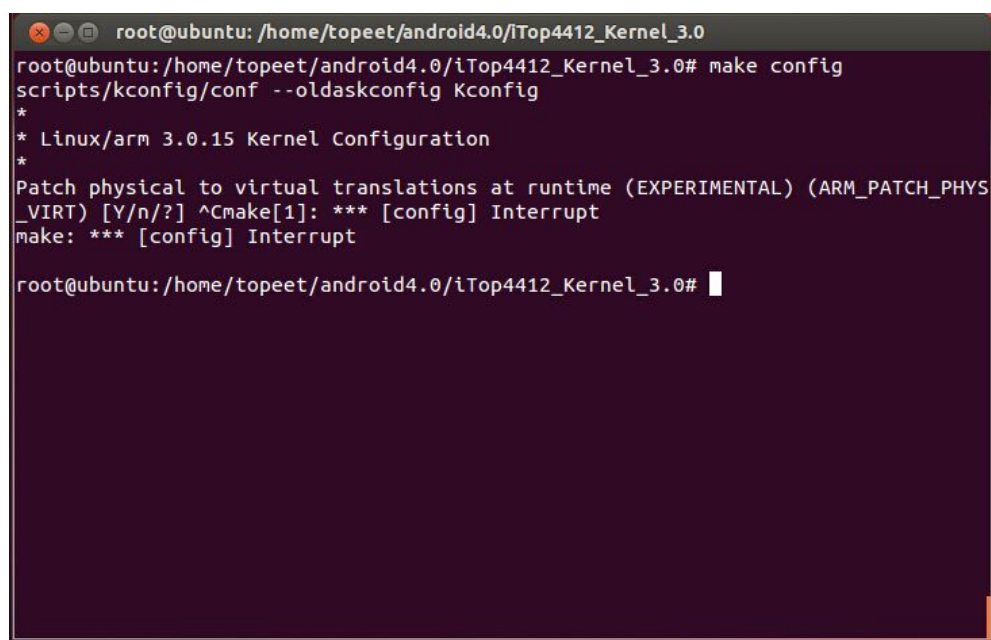
上面不同的命令代表使用不同的工具，如下图所示，在源码目录下，输入命令 “make config” 。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# make config
scripts/kconfig/conf --oldaskconfig Kconfig
*
* Linux/arm 3.0.15 Kernel Configuration
*
Patch physical to virtual translations at runtime (EXPERIMENTAL) (ARM_PATCH_PHYS_VIRT) [Y/n/?]
```

如上图所示，这是一个文本类型的配置工具，根据提示 “scripts/kconfig/conf --oldaskconfig Kconfig” 可以知道，这种方法是旧的配置方法，虽然 Linux 内核可能会长时间的支持，但是不人性化的操作方式，会降低效率，所以现在几乎淘汰了。

使用 “Ctrl+c” 可以退出配置界面，退出后，如下图所示。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# make config
scripts/kconfig/conf --oldaskconfig Kconfig
*
* Linux/arm 3.0.15 Kernel Configuration
*
Patch physical to virtual translations at runtime (EXPERIMENTAL) (ARM_PATCH_PHYS_VIRT) [Y/n/?] ^Cmake[1]: *** [config] Interrupt
make: *** [config] Interrupt

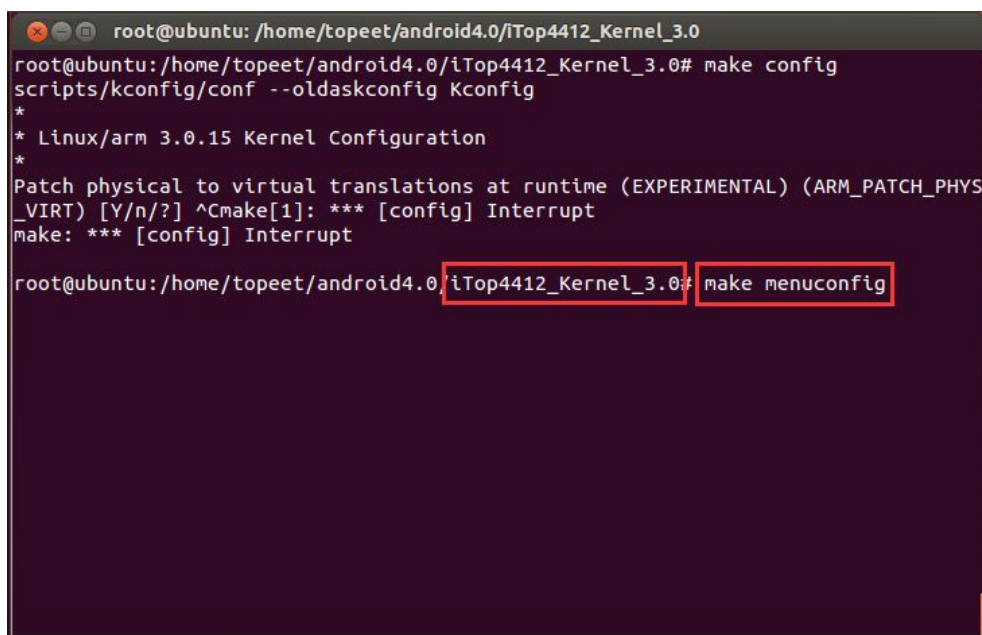
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0#
```

### 3.4.2 Menuconfig 操作方法

前面实验提到过 menuconfig 实现的代码在源码 “scripts” 目录下，不过这里根本不用关心它是怎么实现的，只需要掌握怎么操作即可，就像学习开车，要知道怎么打方向盘，而不需要知道方向盘和轮子之间是怎么传动的。

下面介绍 menuconfig 的操作方法。

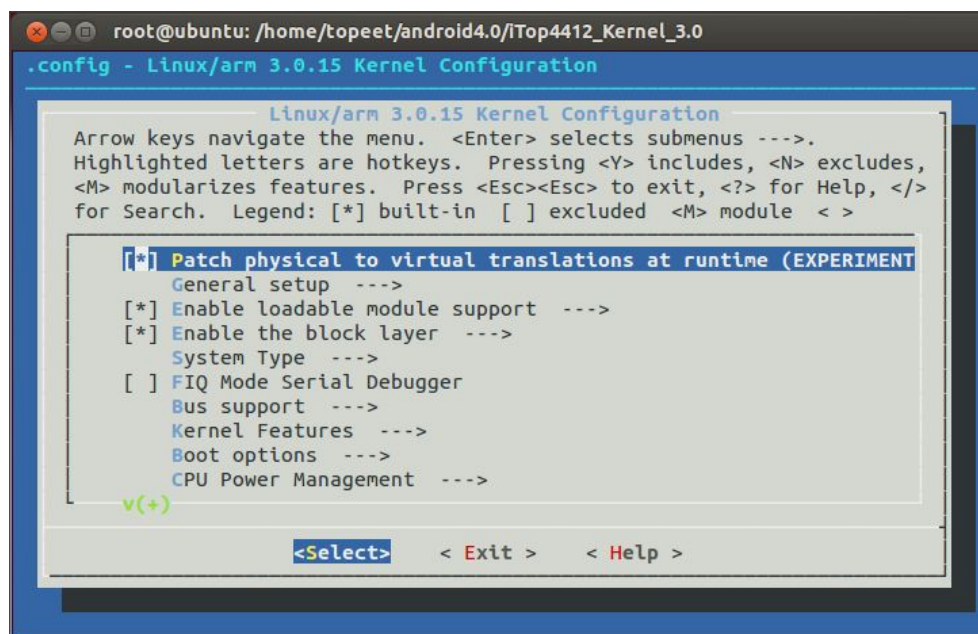
如下图所示，在源码目录下，输入命令 “make Menuconfig” 。



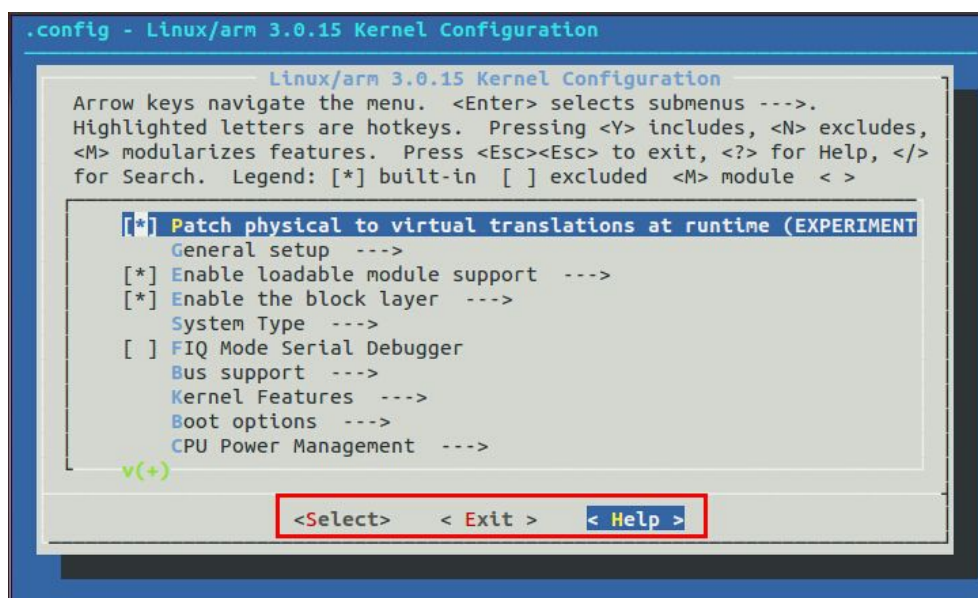
```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# make config
scripts/kconfig/conf --oldaskconfig Kconfig
*
* Linux/arm 3.0.15 Kernel Configuration
*
Patch physical to virtual translations at runtime (EXPERIMENTAL) (ARM_PATCH_PHYS_VIRT) [Y/n/?] ^Cmake[1]: *** [config] Interrupt
make: *** [config] Interrupt

root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# make menuconfig
```

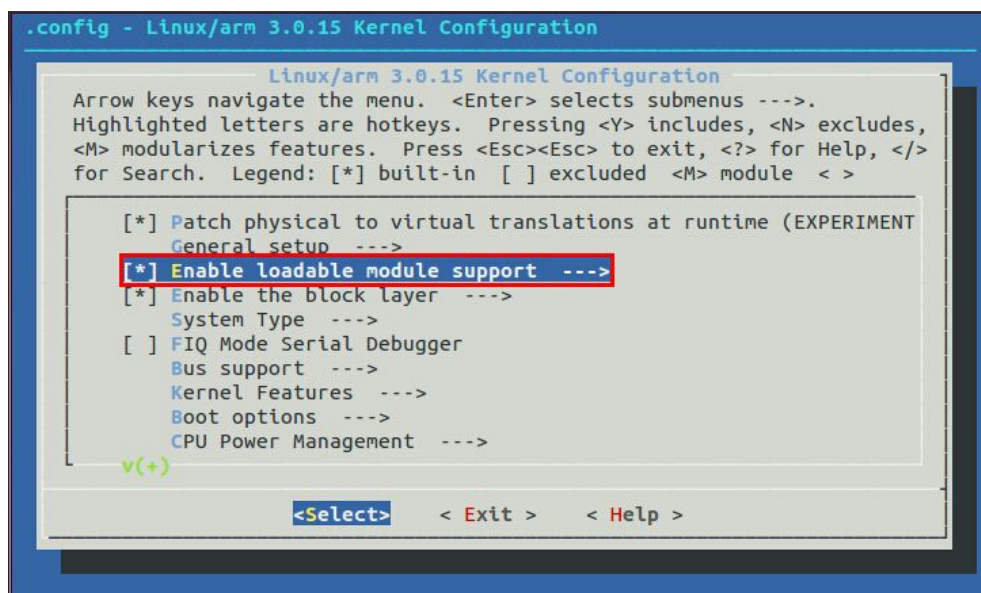
如下图所示，进入配置界面。这个功能界面对应配置工具，它包含配置命令解释器，对配置脚本中使用的命令进行解释；还包含了配置用户界面，用来提供字符界面和图形界面。这些配置工具都是使用脚本语言编写的，不过只用关心怎么使用。



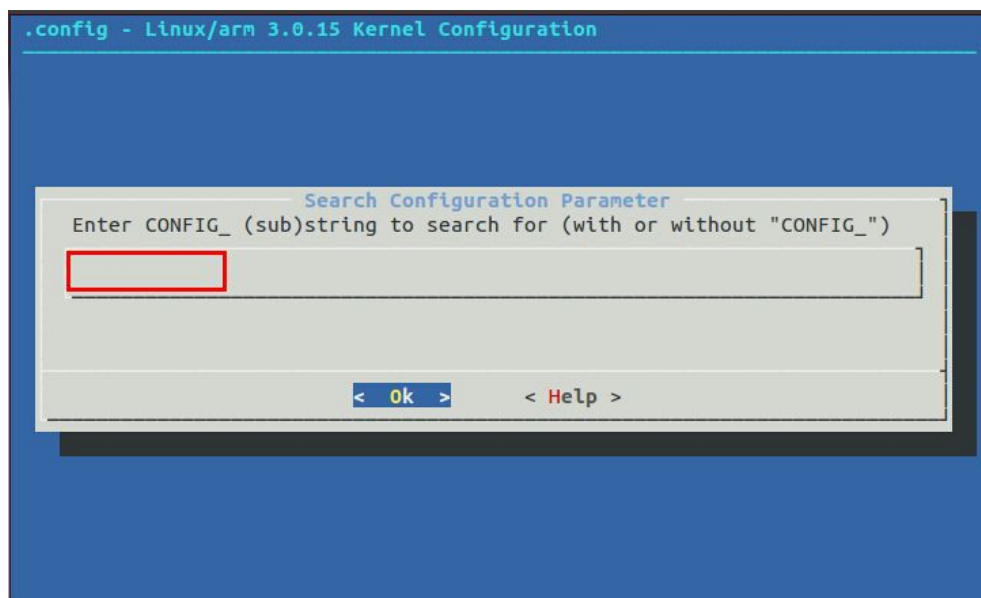
如下图所示，“方向按键”中的“左右”可以选择你需要的操作。“<Select>”表示进入选择的配置界面，“< Exit >”表示返回，“< Help >”可以阅读帮助文档。



如下图所示，“方向按键”中的“上下”可以选择配置的选项。

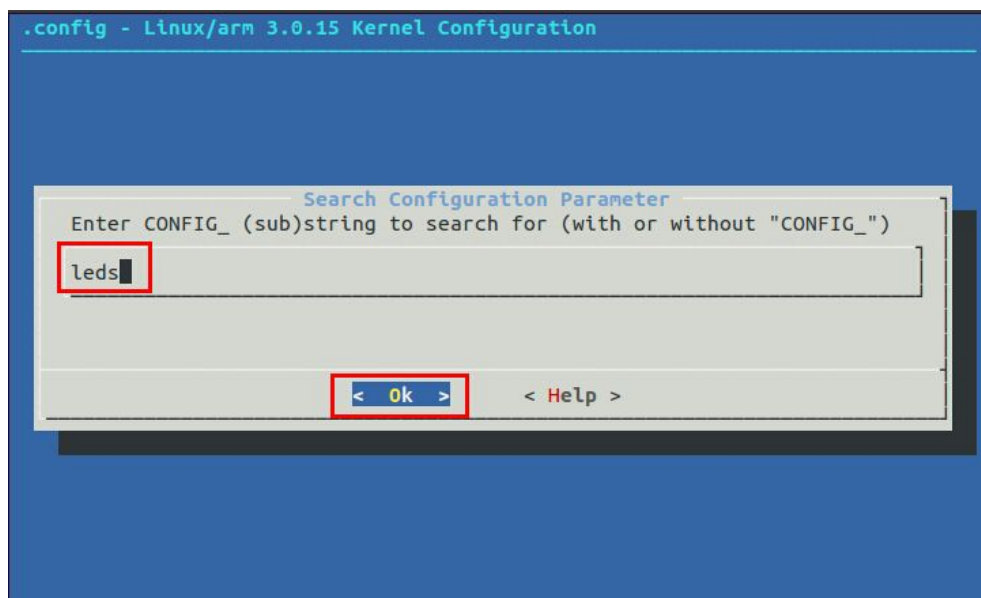


如下图所示，输入 “/” ，可以进入搜索界面。

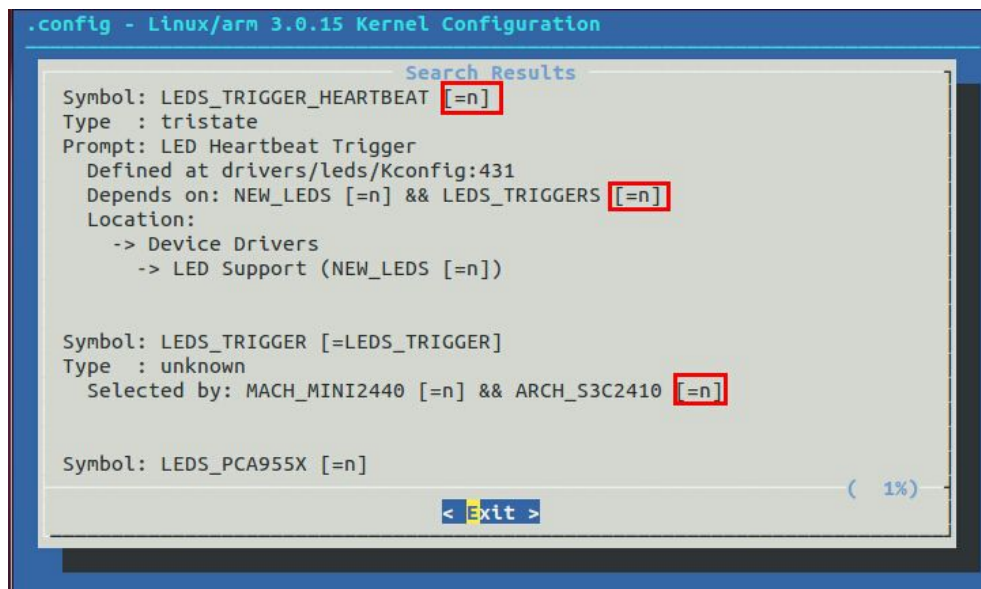


如下图所示，这里来查找一下 “leds” 的驱动，输入 “leds” ，然后按 “回车” 。



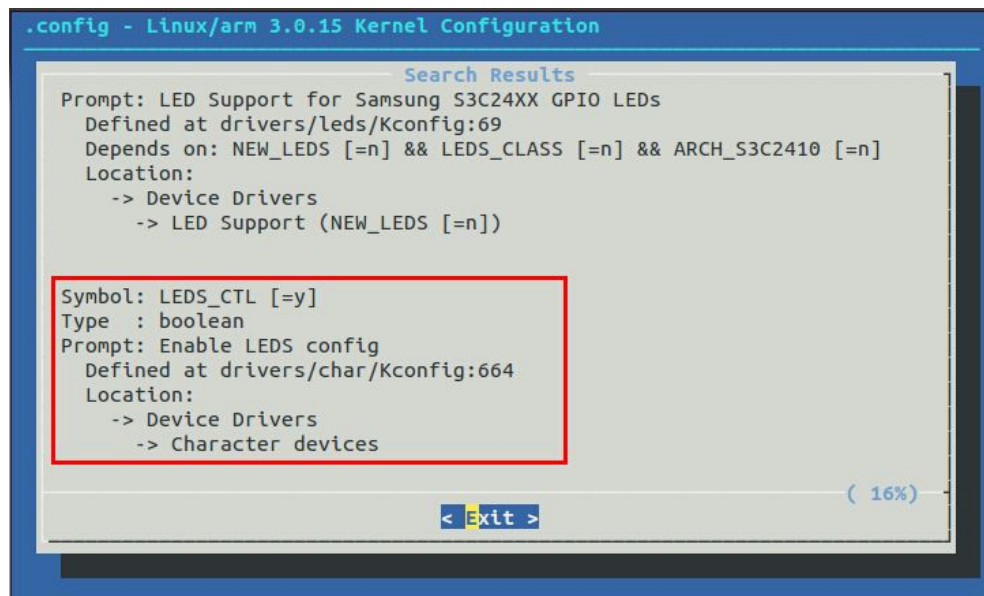


如下图所示，发现很多配置都是“=n”，通过方向按键，控制向下翻页，然后观察那个选项配置成了“=y”。

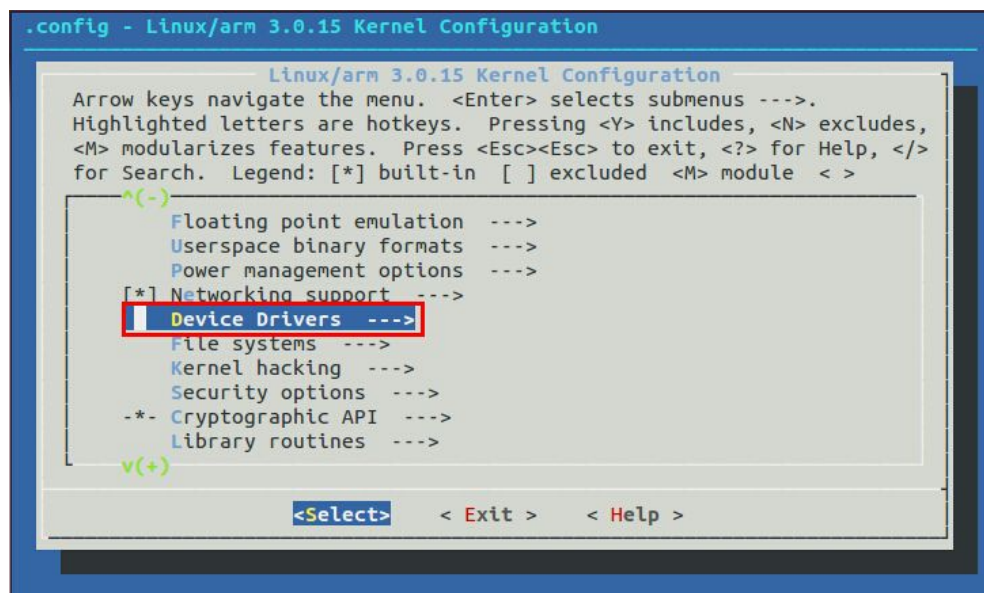


如下图所示，这里可以看到这个 leds 驱动的目录“Device Drivers”“Character devices”。

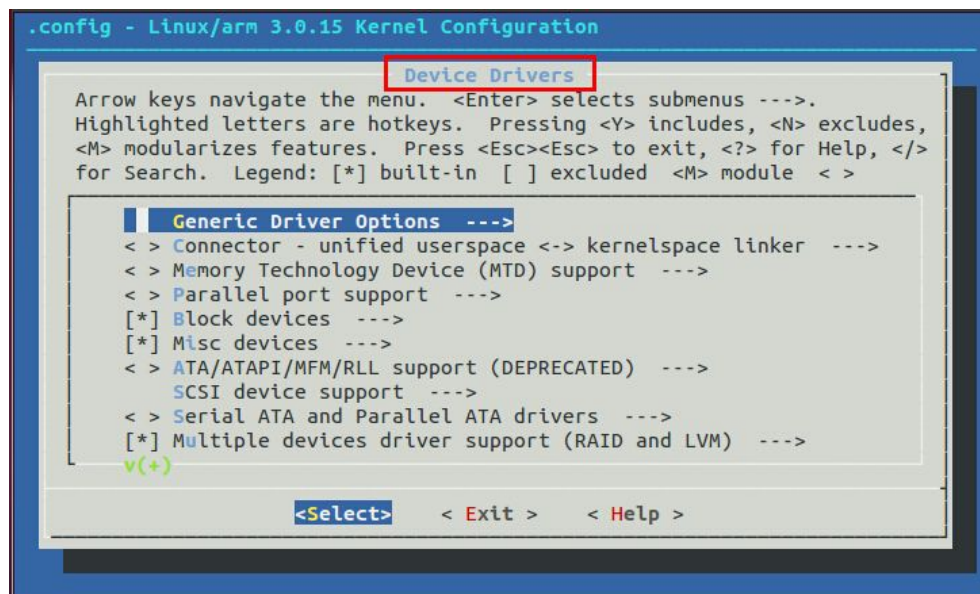




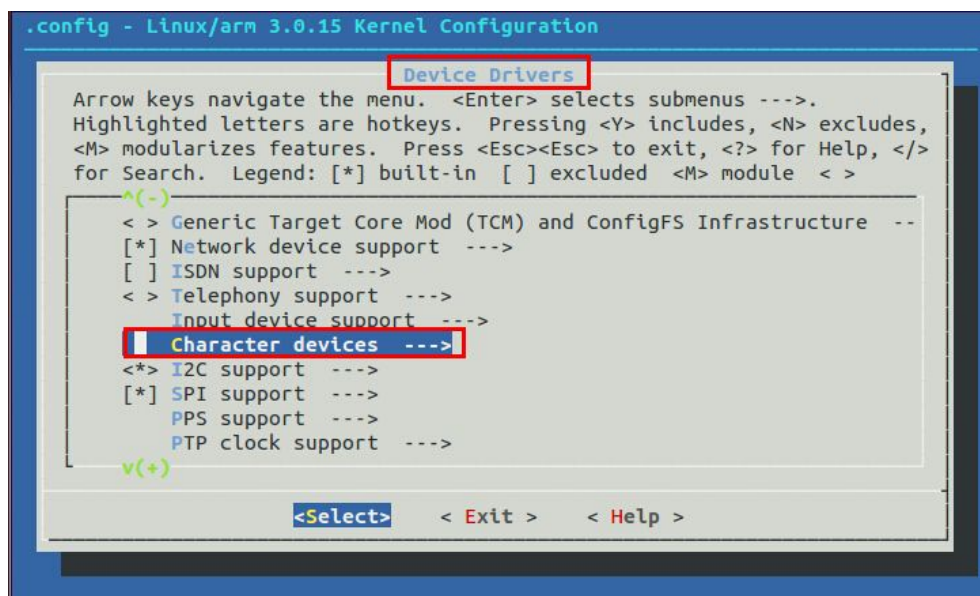
然后,根据查找出来的信息,找到对应的 leds 驱动。如下图,返回配置界面。找到“Device Drivers”目录。输入“回车”。



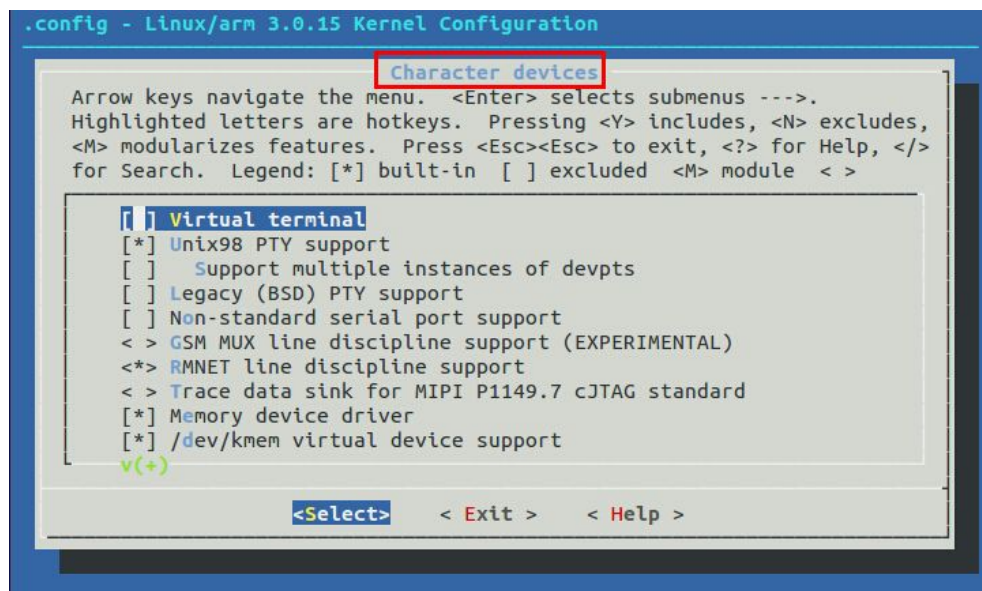
如下图所示,进入“Device Drivers”对应的配置界面。



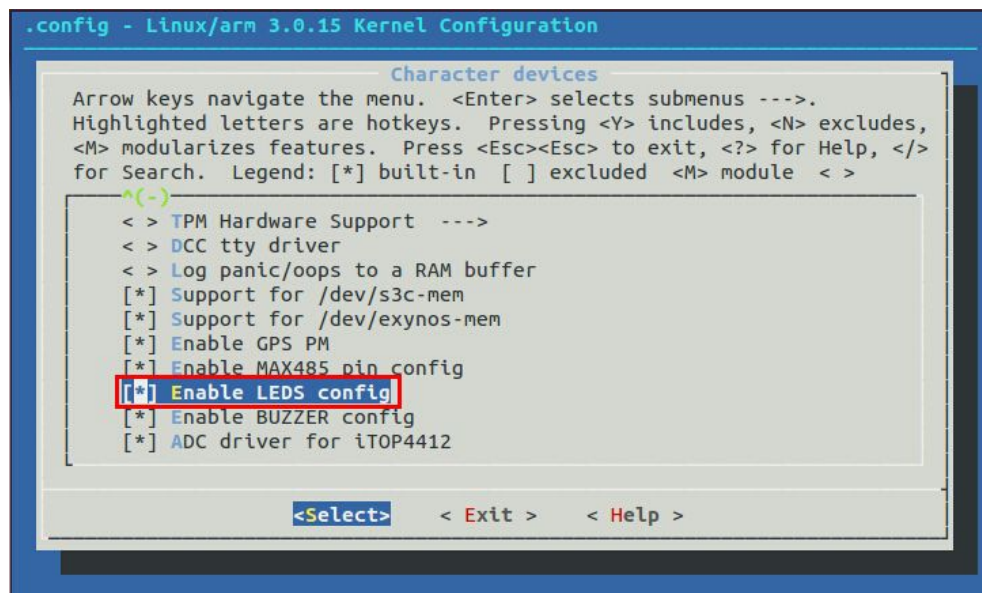
然后，如下图所示，找到“Character devices”，输入“回车”。



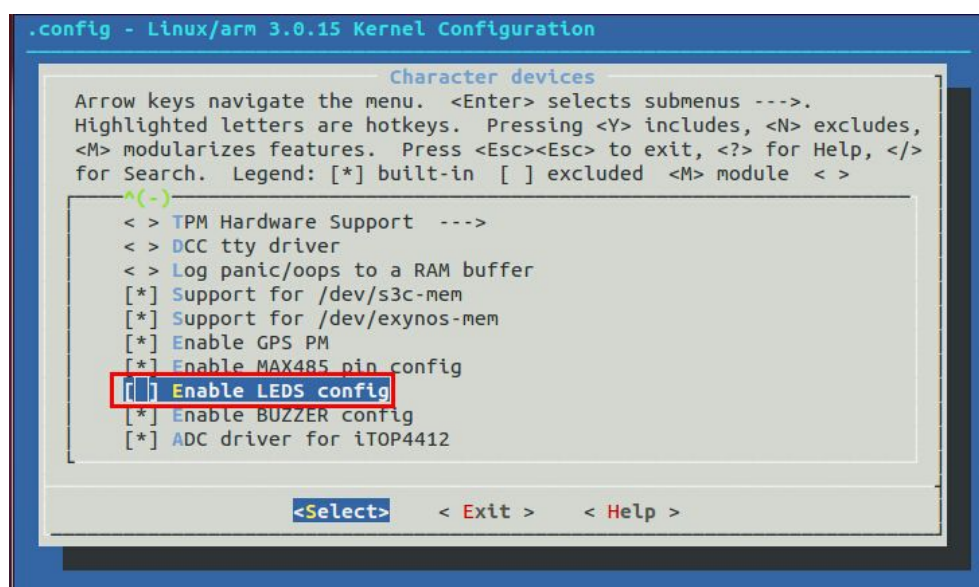
如下图所示，进入“Character devices”配置界面。



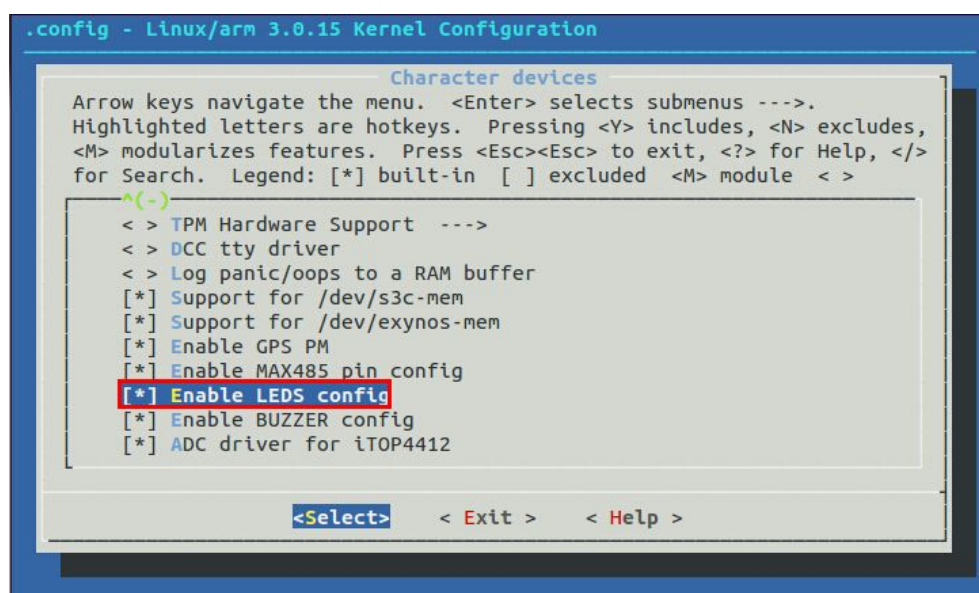
如下图所示，“Enable LEDS config”找到对应的leds 驱动配置选项。缺省配置文件里，这个已经选上了。



单击“空格”键后，去掉leds 驱动选项。

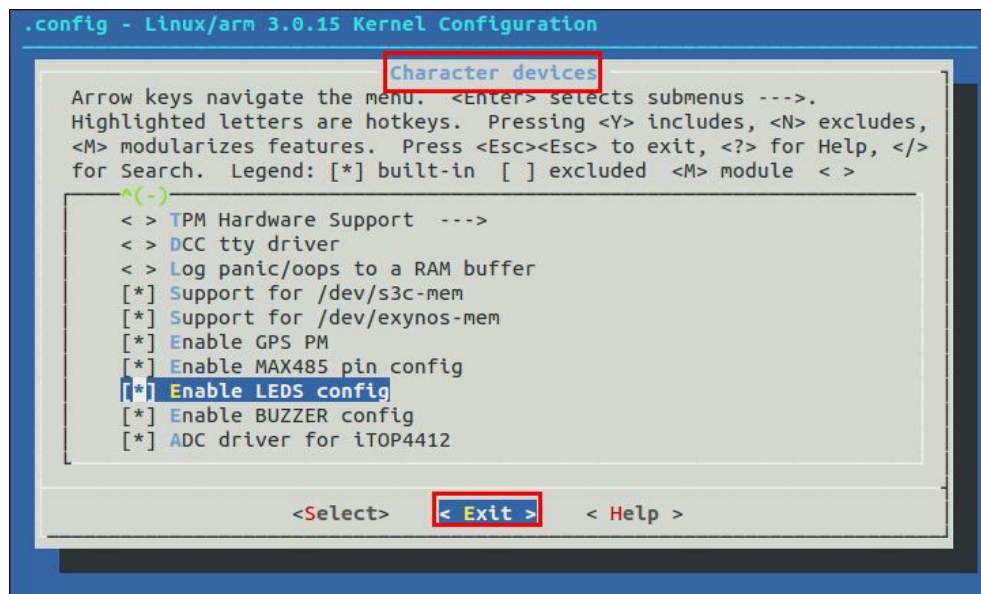


再次敲击“空格”，选上leds驱动的选项。

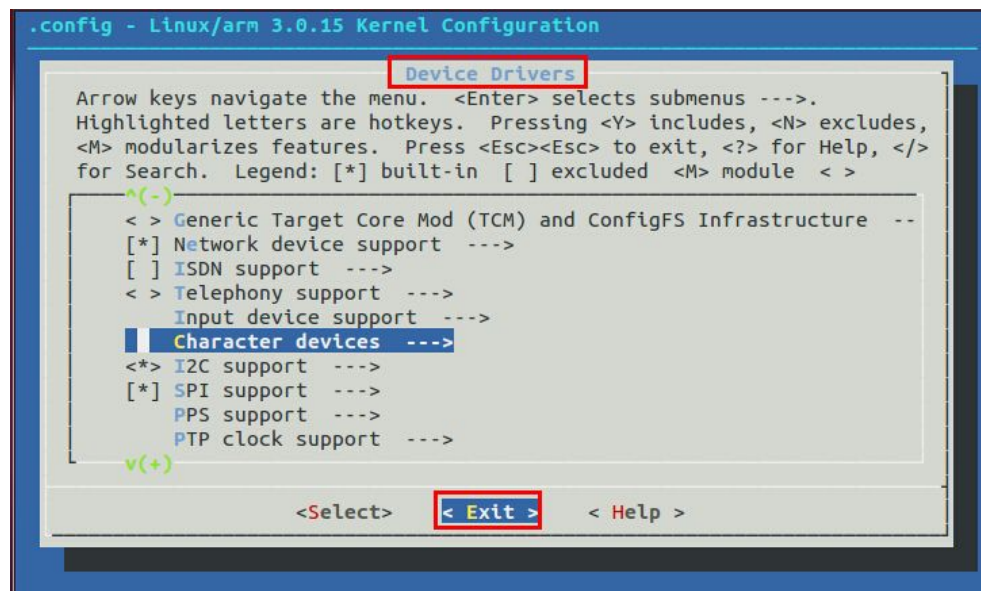


然后，选上“Exit”，如下图所示，输入“回车”。

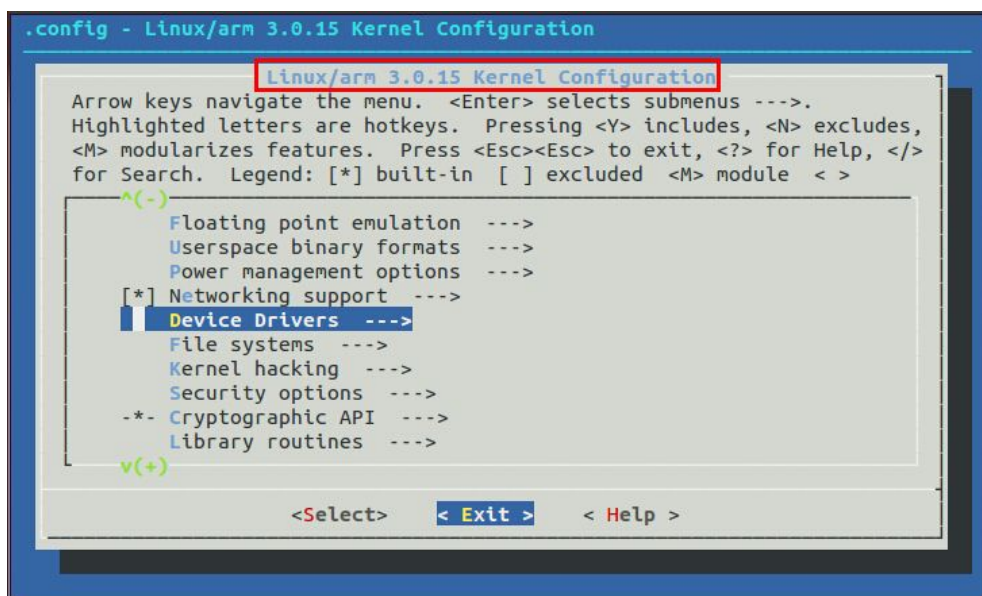




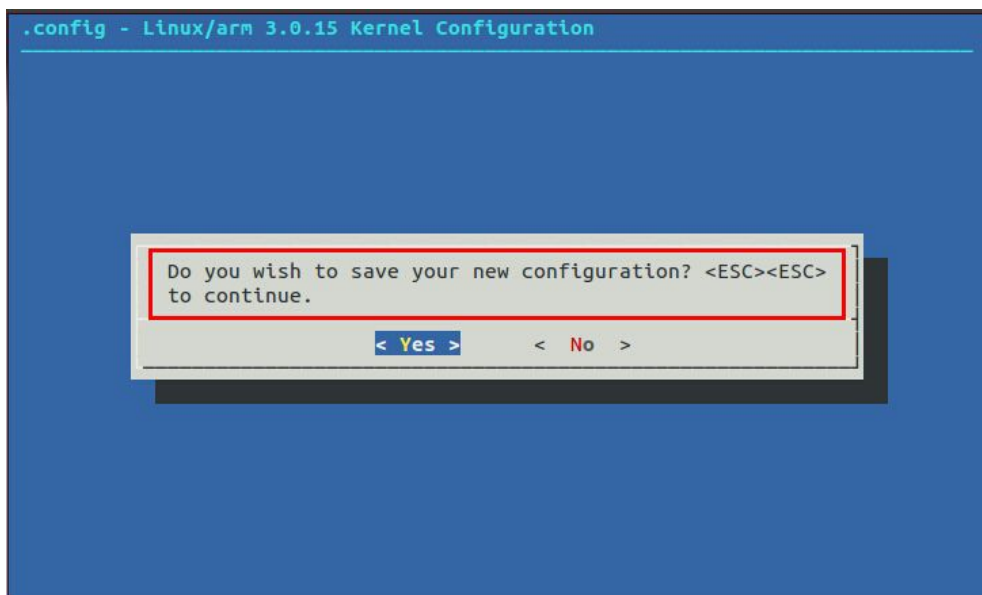
如下图所示，继续退出。



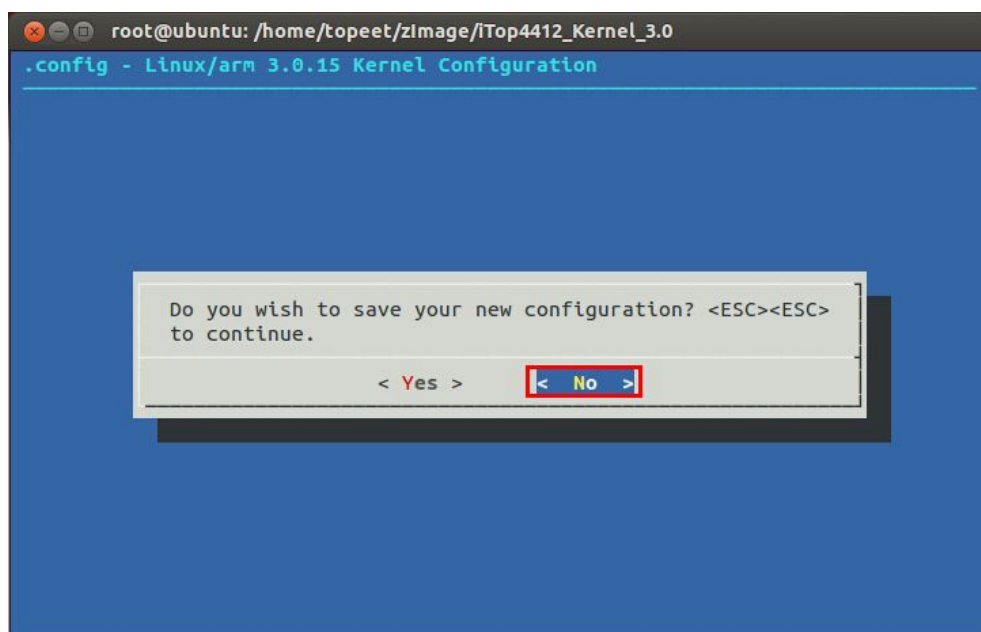
如下图所示，继续退出。



如下图所示，因为修改过配置选项，所以退出的时候会提醒“是否保存新的配置”。



如下图所示，因为第一次操作，担心用户在无意间动了某个配置选项，编译后无法启动，建议选择“No”，不保存退出。



到这里，整个 Menuconfig 配置的操作以及流程就完全介绍完了。如果修改了配置文件，如下图所示的 “.config” 文件就会被修改。再次编译内核的时候，系统会根据新的 config 文件来编译整个内核。

内核的配置非常多，大家可以看一下使用手册 9.4 小节，里面有详细的介绍。

### 3.5 .config 文件和 menuconfig 的关系

menuconfig 最终是为了生成一个.config 文件，这么说大家可能不是很理解，下面给大家举个例子，大家理解了这个例子，对.config 和 menuconfig 的关系就清楚了。

.config 文件在 linux 源码顶层目录，默认是隐藏的，使用查看命令 “ls -a” ，如下图所示。



```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls -a
.                Kbuild                .tmp_kallsyms1.o
..               Kconfig                ..tmp_kallsyms1.o.cmd
arch             kernel                .tmp_kallsyms1.S
binary           kernel_readme.txt     .tmp_kallsyms2.o
block           lib                ..tmp_kallsyms2.o.cmd
.config         MAINTAINERS          .tmp_kallsyms2.S
config_for_android Makefile             .tmp_System.map
config_for_android.2M missing-syscalls.d  tmp_versions
```

以 leds 小灯的驱动配置为例，介绍 menuconfig 具体执行过程。

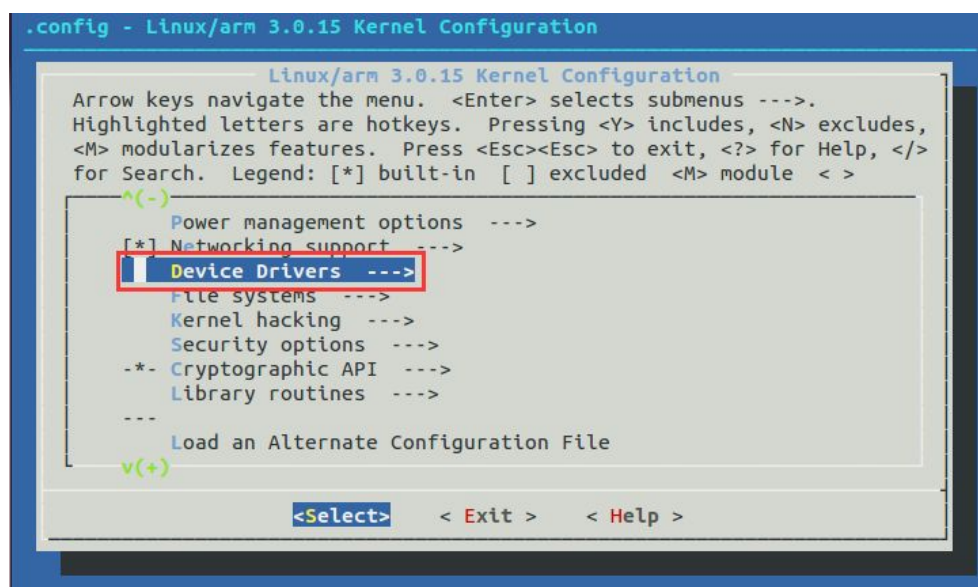
使用命令 “vim .config”，打开文件 “.config” 配置文件，在配置间中搜索

“CONFIG\_LEDS\_CTL”，这里注意要大写，配置文件里面都是一些宏定义，宏定义一般使用英文大写。

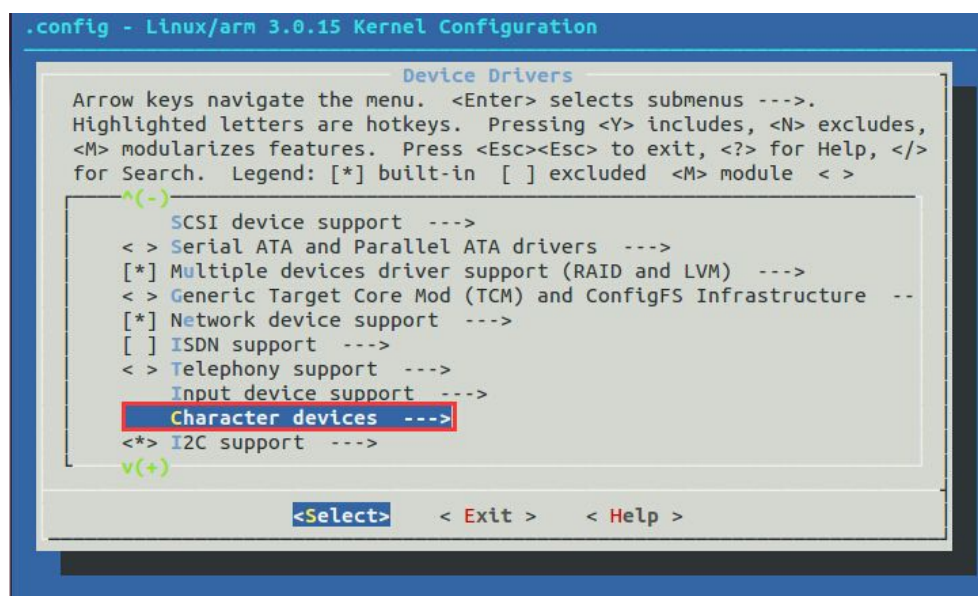
```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0
CONFIG_HW_RANDOM=y
# CONFIG_HW_RANDOM_TIMERIOMEM is not set
# CONFIG_R3964 is not set
# CONFIG_RAW_DRIVER is not set
# CONFIG_TCG_TPM is not set
# CONFIG_DCC_TTY is not set
# CONFIG_RAMoops is not set
CONFIG_S3C_MEM=y
CONFIG_EXYNOS_MEM=y
CONFIG_GPS_PM=y
CONFIG_MAX485_CTL=y
CONFIG_LEDS_CTL=y
# CONFIG_BUZZER_CTL is not set
CONFIG_ADC_CTL=y
CONFIG_RELAY_CTL=y
CONFIG_HELLO_CTL=y
CONFIG_I2C=y
CONFIG_I2C_BOARDINFO=y
CONFIG_I2C_COMPAT=y
CONFIG_I2C_CHARDEV=y
# CONFIG_I2C_MUX is not set
# CONFIG_I2C_HELPER_AUTO is not set
# CONFIG_I2C_SMBUS is not set

1516,8      50%
```

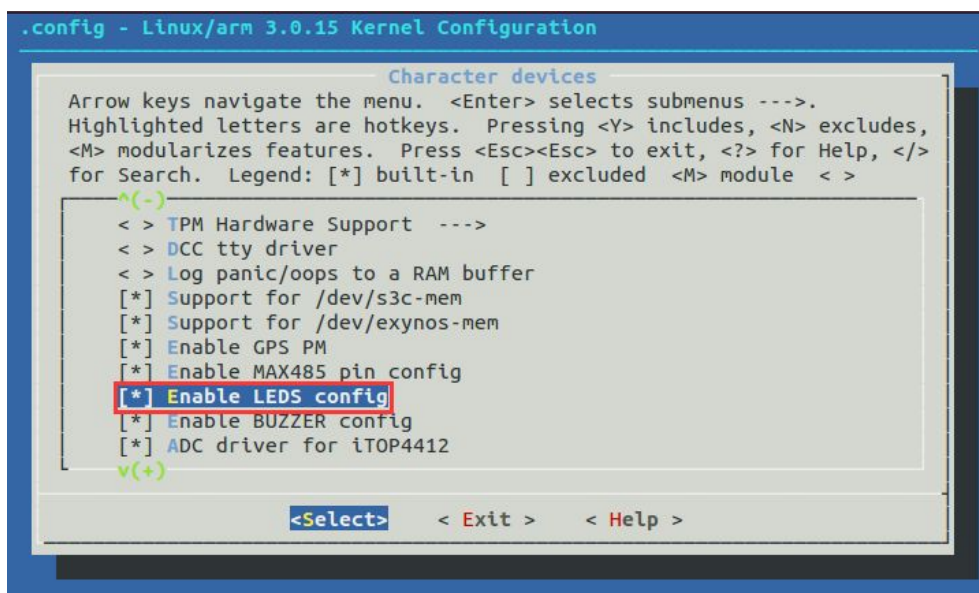
然后来看一下 menuconfig 中要怎么操作。如下图所示，进入 menuconfig 配置界面，找到 “Device Drivers”，输入回车进入 “Device Drivers” 的配置界面。



如下图所示，找到“Character devices”，并进入下一级配置目录。

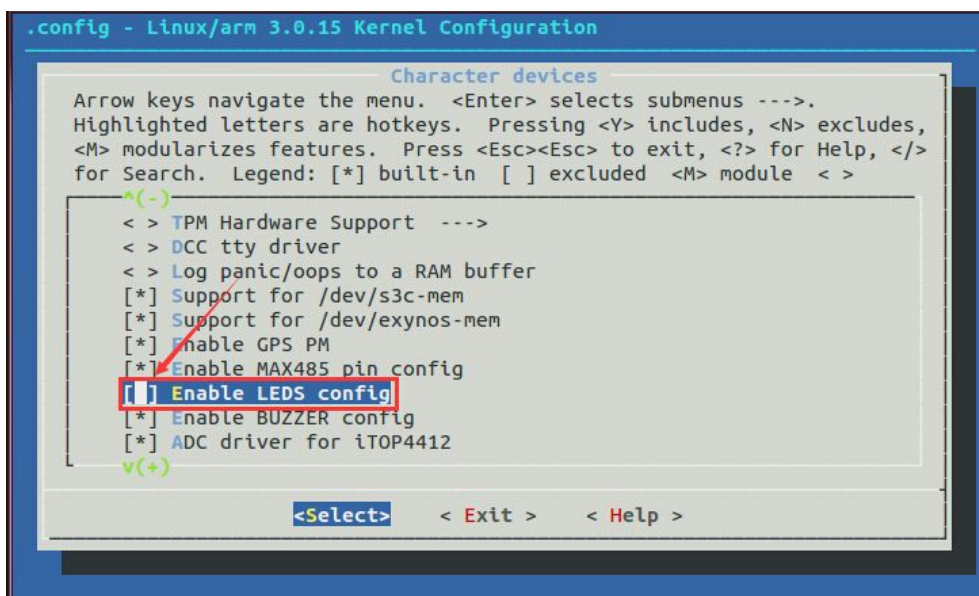


如下图所示，找到“Enable LEDS config”配置界面。

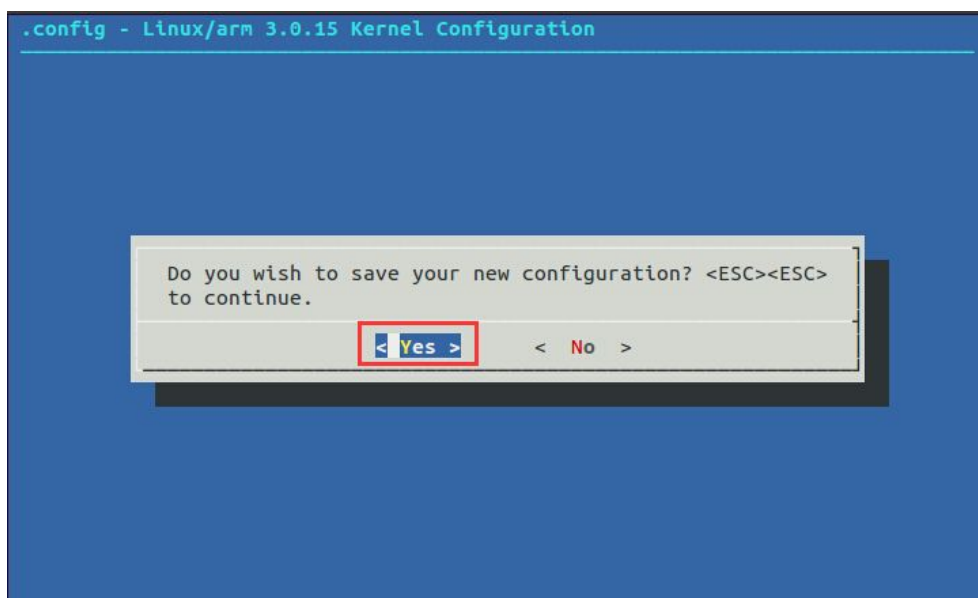


如上图所示,可以看到这个选项选为了“\*”, (通过空格选择,前面已经介绍了 menuconfig 如何操作)。这里选择为“\*”,那么对应“.config”配置文件中的宏定义“CONFIG\_LEDS\_CTL”。

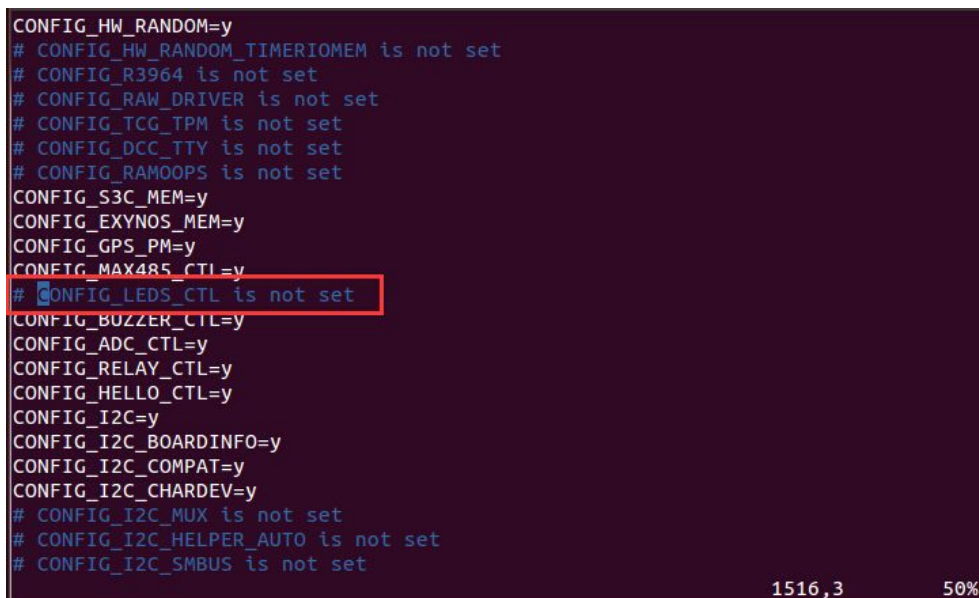
如下图,把默认配置的“\*”去掉,改成如下图所示的“空白”。



如下图所示,保存退出。



这个时候在看一下 “.config” 配置文件中的宏定义 “CONFIG\_LEDS\_CTL” 。



如上图所示，这个宏定义已经被注释掉了。也就是无法在编译过程中，需要用这个宏才能编译的 LEDS 小灯驱动，已经被裁减掉了。

最后大家还是要把 LEDS 的驱动配置上，后面还会用到。



### 3.6 Kconfig 和 menuconfig

这一小节来探讨一下 Kconfig 和 menuconfig 之间的关系。

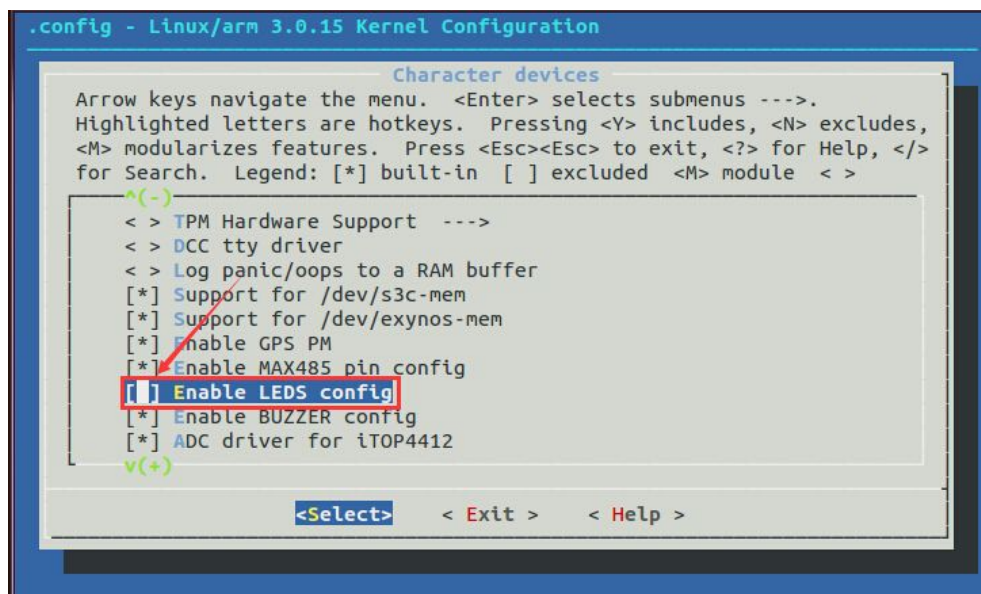
打开源码下的 “drivers/char/Kconfig ” 文件，如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# vim drivers/char/Kconfig
```

在这个文件里面搜索 “LEDS\_CTL” ，如下图所示。

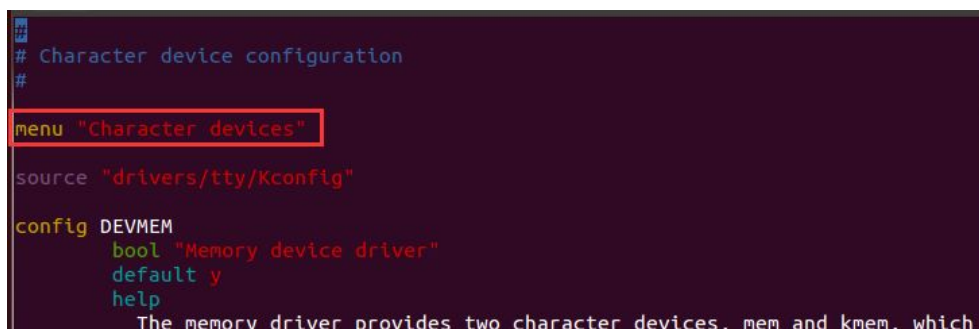
```
config LEDS_CTL
    bool "Enable LEDS config"
    default y
    help
        Enable LEDS config
```

这里的 config LEDS\_CTL 就和 menuconfig 图形配置界面中 “Enable LEDS config” 对应，如下图所示。



当然，讲到这里大家可能还有一个疑问，在 menuconfig 里面还有选择好几项才进入这个配置界面，其实那些选项是和 LEDS config 类似的，也有对应的配置。

进入 “drivers/char/Kconfig ” 文件的第一行，可以看到 “menu "Character devices"”。如下图所示。

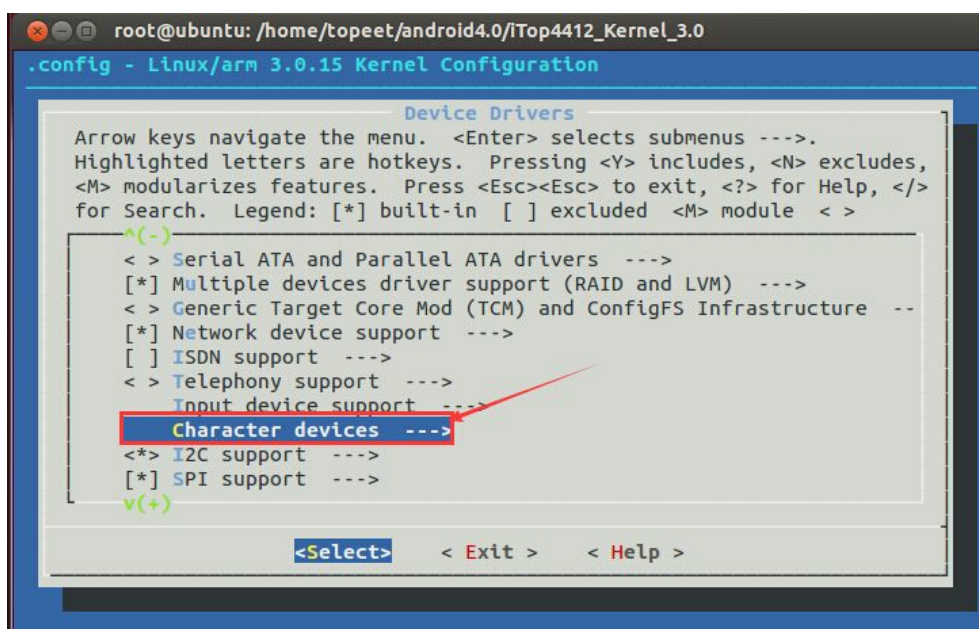


```
# Character device configuration
#
menu "Character devices"

source "drivers/tty/Kconfig"

config DEVMEM
    bool "Memory device driver"
    default y
    help
        The memory driver provides two character devices, mem and kmem, which
```

如上图所示，这个 “Character devices” 就和 menuconfig 中的菜单对应，如下图所示。



然后打开 “drivers/Kconfig” 文件，如下图所示是 “Character devices” 的上一级菜单 “Device Drivers”。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
menu "Device Drivers"

source "drivers/base/Kconfig"

source "drivers/connector/Kconfig"

source "drivers/mtd/Kconfig"

source "drivers/nf/Kconfig"
```

如下图所示，对应“Character devices”中的配置界面。

```
source "drivers/char/Kconfig"

source "drivers/i2c/Kconfig"

source "drivers/spi/Kconfig"

source "drivers/pps/Kconfig"

/char
```

这里特别提醒：无论是 Kconfig 最底层的配置，还是这里菜单的语法，用户只需要仿照者写就可以了。以为无论将来你从哪里拿到一份内核源码，这些框架都已经搭建好了，这一块都是学会仿写即可。千万不要花费太多的时间去学习这个脚本语法，到使用的时候简单的看一下就可以理解的，即使不能理解，依葫芦画瓢就可以了。

最后还有最顶层的 Kconfig 文件， 打开来看一下，如下图所示。

```
# For a description of the syntax of this configuration file,
# see Documentation/kbuild/kconfig-language.txt.
#
mainmenu "Linux/$ARCH $KERNELVERSION Kernel Configuration"

config SRCARCH
    string
    option env="SRCARCH"

source "arch/$SRCARCH/Kconfig"

~
```

如上图所示，这里简单的介绍一下具体含义。

注释主要说的是配置文件的帮助文档。



mainmenu "Linux/\$ARCH \$KERNELVERSION Kernel Configuration"

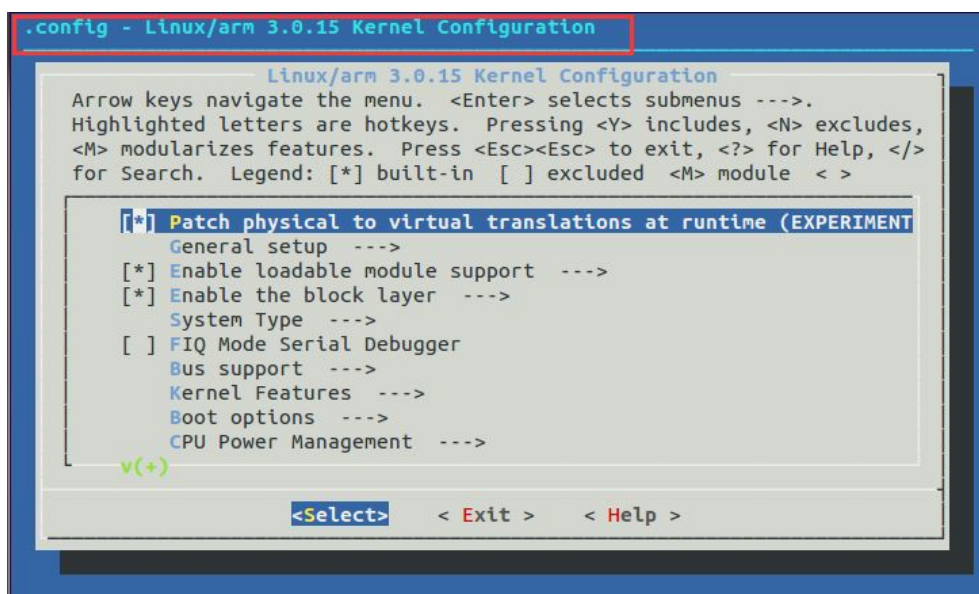
是主菜单的标题

\$ARCH 是定义在 Makefile 中的一个变量。指的是代码运行的具体环境，linux 运行在开发板上，毫无疑问是 arm。

\$KERNELVERSION 是内核版本号

\$SRCARCH 也是 arm，在 Makefile 中可以看到定义，值和 ARCH 一样。

和下图 menuconfig 顶层菜单对应。



Menuconfig 会读取 Makefile 中的参数，其中之一是内核版本号，如果修改一下参数，menuconfig 下主菜单标题就会改变，如下图所示。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
VERSION = 3
PATCHLEVEL = 0
SUBLEVEL = 15
EXTRAVERSION =
NAME = Sneaky Weasel

# *DOCUMENTATION*
# To see a list of typical targets execute "make help"
# More info can be located in ./README
# Comments in this file are targeted only to the developer, do not
# expect to learn how to build the kernel reading this file.

# Do not:
# o use make's built-in rules and variables
#   (this increases performance and avoids hard-to-debug behaviour);
# o print "Entering directory ...";
MAKEFLAGS += -rR --no-print-directory

# Avoid funny character set dependencies
unexport LC_ALL
LC_COLLATE=C
LC_NUMERIC=C
export LC_COLLATE LC_NUMERIC
"Makefile" 1575L, 53911C 1,1 Top
```

在 Makefile 文件中，ARCH 变量定义如下，在 ARCH 下定义七行的地方 SRCARCH 指定为 ARCH,也就是 ARM，如下图所示。

```
# Note: Some architectures assign CROSS_COMPILE in their arch/*/Makefile
export KBUILD_BUILDHOST := $(SUBARCH)
ARCH ?= arm
CROSS_COMPILE := /usr/local/arm/arm-2009q3/bin/arm-none-linux-gnueabi-
#CROSS_COMPILE ?= /usr/local/arm/4.5.1/bin/arm-none-linux-gnueabi-
CROSS_COMPILE ?= $(CONFIG_CROSS_COMPILE: "%"=%)

# Architecture as present in compile.h
UTS_MACHINE := $(ARCH)
SRCARCH := $(ARCH)

# Additional ARCH settings for x86
ifeq ($(ARCH),i386)
    SRCARCH := x86
endif
ifeq ($(ARCH),x86_64)
    SRCARCH := x86
endif

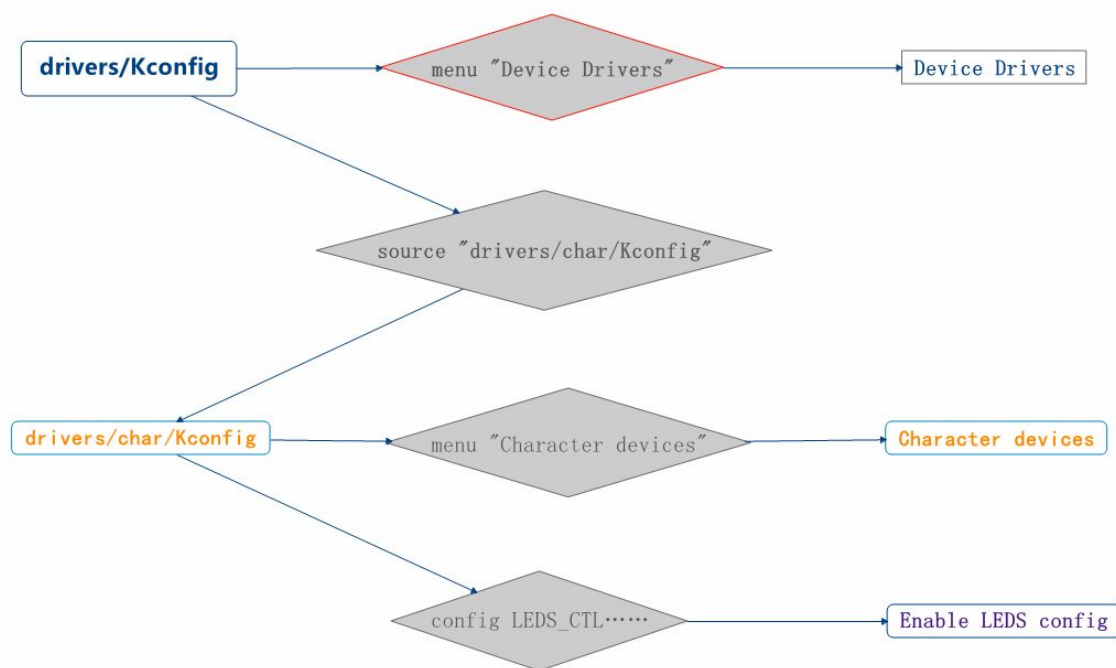
/ARCH 200,22 12%
```

上图中的路径就是前面搭建环境时候新建文件夹的路径，将来碰到什么版本的 Linux 内核，搭建环境的时候，这两个地方都需要对应起来，不然刚刚编译就会报一个错误，提示找到不编译器。

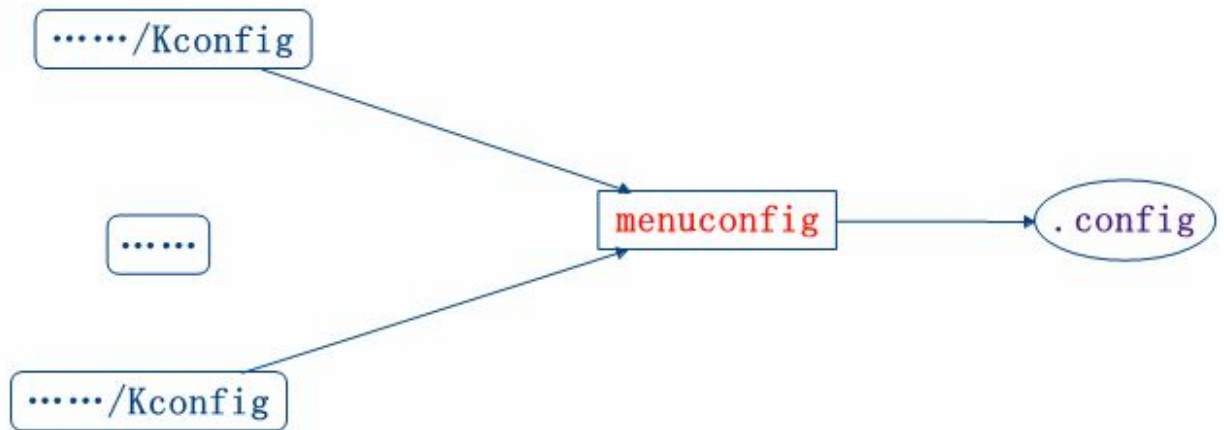
配置里面还有其它的一些知识点，例如依赖、反向依赖、三态变量等，到后面用到的时候再介绍。

### 3.7 图解 Kconfig 和 menuconfig 的关系

下图是以 LEDS 小灯驱动为例，图解 Kconfig 和 menuconfig 的关系。



如下图所示，最终得到配置需要的.config 文件。



### 3.8 其它配置文件

现在看一下提供源码中的，除了 “.config” 文件以外，还有其它的 config\_for\_xxx,如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls
arch          Documentation  lib           REPORTING-BUGS
binary        drivers       MAINTAINERS  samples
block         firmware     Makefile     scripts
config_for_android fs           mm           security
config_for_android_2M include      modem.patch  sound
config_for_linux init         modules.builtin System.map
config_for_ubuntu ipc          modules.order tools
config_for_ubuntu_hdmi Kbuild      Module.symvers usr
COPYING       Kconfig     net          virt
CREDITS       kernel      pull_log.bat vmlinux
crypto        kernel_readme.txt README       vmlinux.o
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

这些都是为了不同的文件系统准备的，有 Android 的配置文件有 Qt 的配置文件等等，或者特殊功能的 config 文件。这些 config 文件都是通过 menuconfig 生成，然后改成易识别的名称。它们公用一套代码，通过 menuconfig 裁减组合成不同功能.config，下一小节介绍 Make 命令，通过 Make 命令生成不同的内核。

注意用户实际源码下的 config\_for\_xxx 可能和上图有区别，以用户实际解压后的为准。