

# Linux 内核配置单讲解

我们说的 Linux 其实指的就是核心（kernel）而已。这个核心控制你主机的所有硬件并提供系统所有的功能。我们开机的时候其实就是利用开机管理程序载入这个核心文件来侦测硬件，在核心载入适当的驱动程序后，你的系统才能够顺利的运行。

## kernel 内核

1. Linux 内核（英语：Linux kernel），是一种计算机操作系统内核，Linux 是一个宏内核，设备驱动程序可以完全访问硬件。Linux 内的设备驱动程序可以方便地以模块化(modularize)的形式设置，并在系统运行期间可直接装载或卸载。
2. 从技术上说 Linux 只是一个内核。“内核”指的是一个提供硬件抽象层、磁盘及文件系统控制、多任务等功能的系统软件。一个内核并不是一套完整的操作系统。有一套基于 Linux 内核的完整操作系统叫作 Linux 操作系统，或是 GNU/Linux（在该系统中包含了很多 GNU 计划的系统组件）。
3. 核心源代码下的次目录，在上述核心目录下含有哪些重要数据呢？基本上有下面这些东西：

arch：与硬件平台有关的项目，大部分指的是 CPU 的类别，例如 x86, x86\_64, Xen 虚拟支持等；

block：与区块设备较相关的设置数据，区块数据通常指的是大量储存媒体！还包括类似 ext3 等文件系统的支持是否允许等。

crypto：核心所支持的加密的技术，例如 md5 或者是 des 等等；

Documentation：与核心有关的一堆说明文档，若对核心有极大的兴趣，要瞧瞧这里！

drivers：一些硬件的驱动程序，例如显卡、网卡、PCI 相关硬件等等；

firmware：一些旧式硬件的微指令码（固件）数据；

fs：核心所支持的 filesystems，例如 vfat, reiserfs, nfs 等等；

include：一些可让其他程序调用的标头（header）定义数据；

init：一些核心初始化的定义功能，包括挂载与 init 程序的调用等；

ipc：定义 Linux 操作系统内各程序的沟通；

kernel：定义核心的程序、核心状态、线程、程序的调度（schedule）、程序的讯号（signle）等

lib：一些函数库；

mm：与内存单元有关的各项数据，包括 swap 与虚拟内存等；

net：与网络有关的各项协定数据，还有防火墙模块（net/ipv4/netfilter/\*）等等；

security：包括 selinux 等在内的安全性设置；

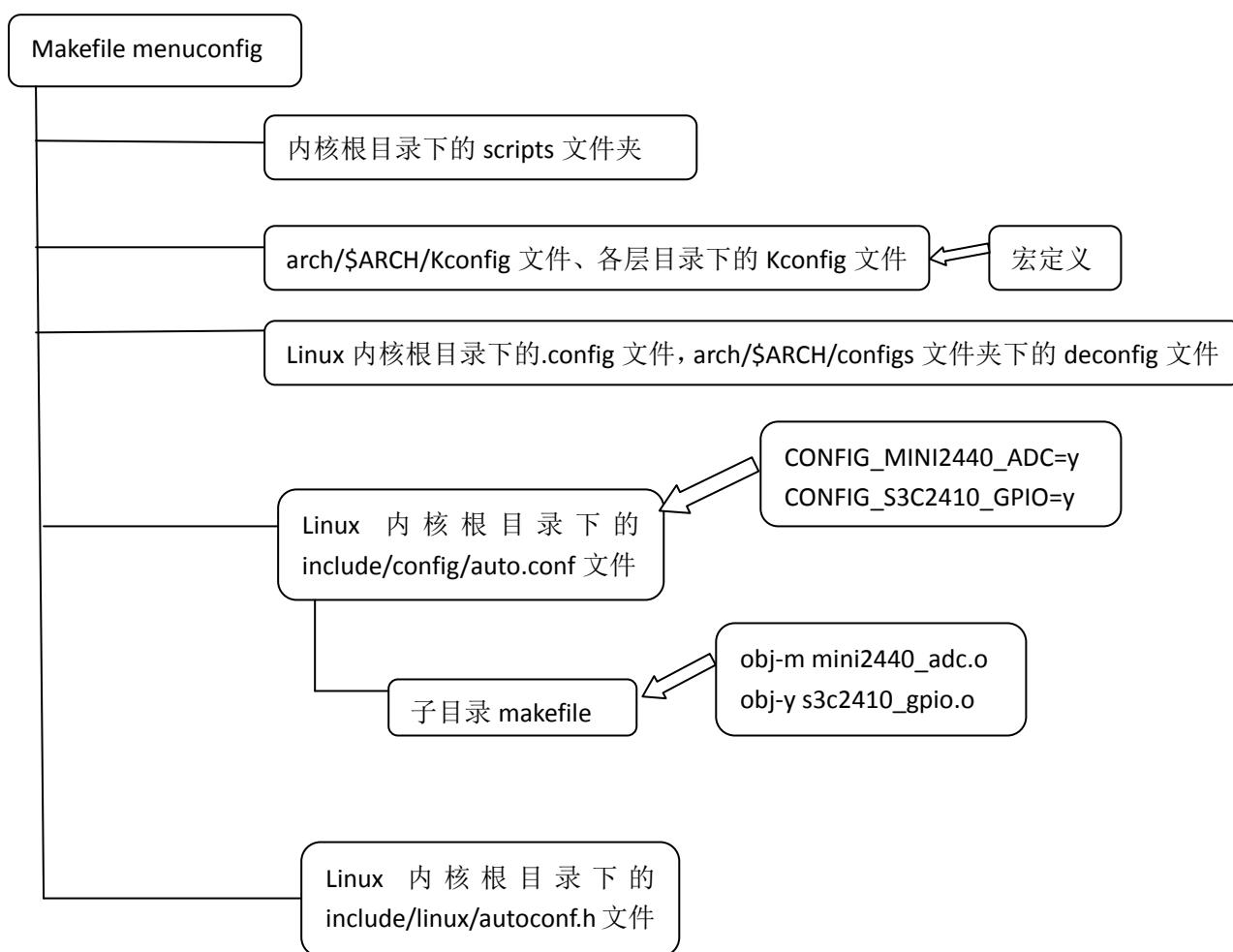
sound：与音效有关的各项模块；

virt：与虚拟化机器有关的信息，目前核心支持的是 KVM（Kernel base Virtual Machine）。

Linux 内核官网：<http://www.kernel.org/>

## Linux 内核的配置系统由三个部分组成，分别是：

1. **Makefile**：分布在 Linux 内核源代码中的 **Makefile**，定义 Linux 内核的编译规则；
2. 配置文件（**config.in**）：给用户提供配置选择的功能；
3. 配置工具：包括配置命令解释器（对配置脚本中使用的配置命令进行解释）和配置用户界面（提供基于字符界面、基于 **Ncurses** 图形界面以及基于 **Xwindows** 图形界面的用户配置界面，各自对应于 **Make config**、**Make menuconfig** 和 **make xconfig**）。



1、scripts 文件夹存放的是跟 **make menuconfig** 配置界面的图形绘制相关的文件，我们作为使用者无需关心这个文件夹的内容。

2、读取 arch/arch/\$ARCH/Kconfig 以及各子目录下的 Kconfig 文件，生成配置条目。

\$ARCH 由 linux 内核根目录下的 makefile 文件决定

ARCH       ?= arm

CROSS\_COMPILE   ?= arm-linux-

Kconfig 文件中为配置信息的宏定义，与我们在 **make menuconfig** 图形界面看到的信息一致。

例如：

```

config CPU_S3C2410_DMA
    bool
    depends on S3C2410_DMA && (CPU_S3C2410 || CPU_S3C2442)
    default y if CPU_S3C2410 || CPU_S3C2442
    help
        DMA device selection for S3C2410 and compatible CPUs

```

因此，Kconfig 文件很重要的作用就是：定义配置宏、相关依赖关系、帮助信息。

- 3、读取内核根目录下.config 文件，生成配置选项:[\*]编译进内核 [M]编译为模块 []不编译  
arch/arm/configs/文件夹下存放了一些配置模板

我们可以通过 `cp /arch/arm/configs/xx_defconfig .config` 来使用这些配置模板

通过图形界面变更配置选项会自动更新到.config 文件中

`make disclean` 会删除.config

- 4、编译过程根据.config 生成 Linux 内核根目录下的 include/config/auto.conf 文件

```
CONFIG_EEPROM_93CX6=m
```

```
CONFIG_DM9000=y
```

根目录 Makefile 以及子目录的 Makefile 根据 auto.conf 生成编译条件

```
obj-$(CONFIG_DM9000) += dm9000.o //obj-m += dm9000.o
```

- 5、编译过程根据.config 生成 Linux 内核根目录下的 include/linux/autoconf.h 文件

.config 或 auto.conf 中定义要编译为 m 模块的项，如：

```
CONFIG_DEBUG_NX_TEST=m
```

在 autoconf.h 中会被定义为：

```
#define CONFIG_DEBUG_NX_TEST_MODULE 1
```

.config 或 auto.conf 中定义为编译为 y 的选项,如：

```
CONFIG_DM9000= y
```

在 autoconf.h 中会被定义为：

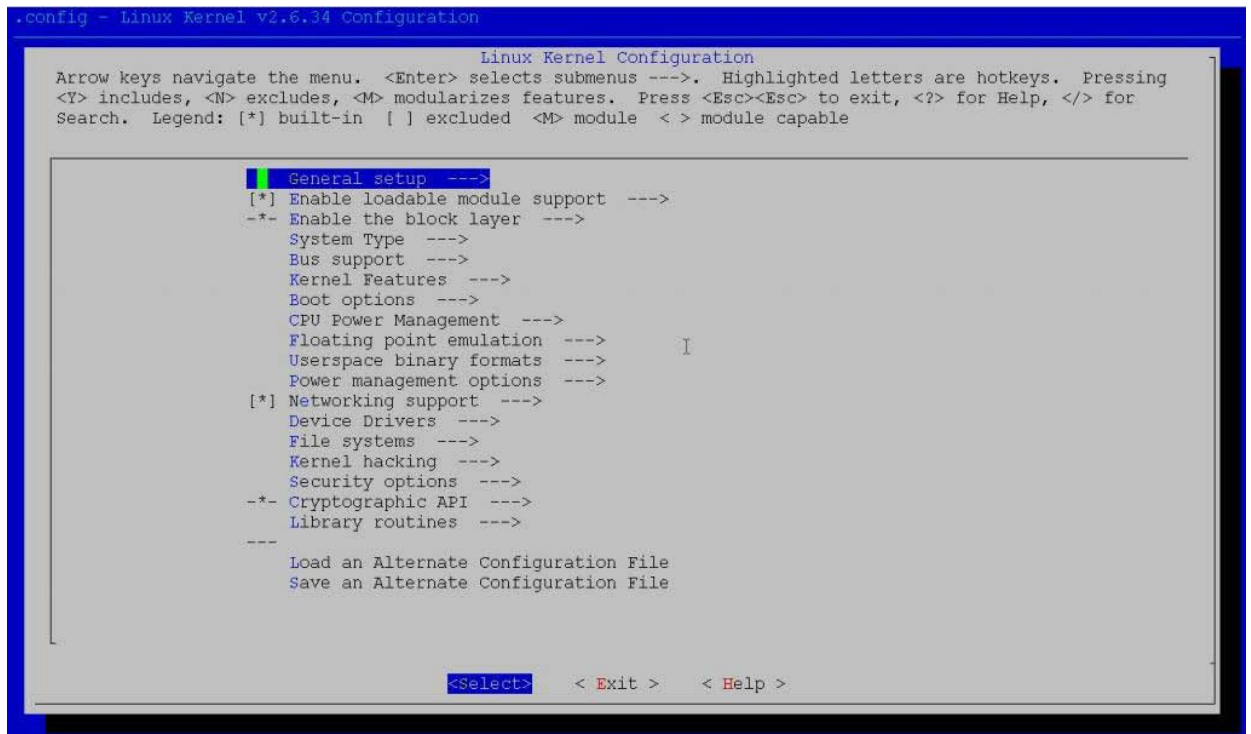
```
#define CONFIG_DM9000 1
```

autoconf.h 中是.config 或者 auto.conf 中配置信息的另一种体现形式，它是站在源码的角度，供源码使用的 C 语言宏定义。

## 6、总结

我们在使用 `make menuconfig` 时，首先会确定架构 arch，然后读取 arch 目录的 Kconfig 中的配置宏定义，生成编译条目，然后读取 Linux 内核根目录下的.config 选项，将.config 中的配置信息显示在图形界面上[\*] [M] or []。我们在图形界面中更改配置选项会自动保存到.config 文件中。编译过程根据.config 随后生成 auto.conf 文件，它决定了 makefile 中各个文件的编译类型，静态编译进内核、编译成模块、不编译;同时生成 autoconf.h，它以 C 语言宏定义的形式表达了 各个文件是否被编译，源码中会判断某文件是否被编译进行不同的处理。

我们在进行 linux 内核配置的时候经常会执行 `make menuconfig` 这个命令，然后屏幕上会出现以下界面：



```
.config - Linux Kernel v2.6.34 Configuration

Linux Kernel Configuration

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for
Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

General setup --->
[*] Enable loadable module support --->
--*- Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
Device Drivers --->
File systems --->
Kernel hacking --->
Security options --->
--*- Cryptographic API --->
Library routines --->
Load an Alternate Configuration File
Save an Alternate Configuration File

<Select> < Exit > < Help >
```

这几个元件的大致用法如下：

- “左右方向键”：可以移动最下面的 <Select>, <Exit>, <Help>项目；
  - “上下方向键”：可以移动上面大框框部分的反白光柱，若该行有箭头（--->）则表示该行内部还有其他细项需要来设置的意思；
  - 选定项目：以“上下键”选择好想要设置的项目之后，并以“左右键”选择 <Select> 之后，按下“Enter”就可以进入该项目去作更进一步的细部设置；
  - 可挑选之功能：在细部项目的设置当中，如果前面有 [ ] 或 < > 符号时，该项目才可以选择，而选择可以使用“空白键”来选择；
  - 若为 [ ] < > 则表示编译进核心；若为 <M> 则表示编译成模块！尽量在不知道该项目为何时，且有模块可以选，那么就可以直接选择为模块！
  - 当在细项目选择 <Exit> 后，并按下 Enter，那么就可以离开该细部项目！
- 基本上建议只要“上下左右的方向键、空白键、Enter”这六个按键就好了！不要使用 Esc，否则一不小心就有可能按错的！另外，关于整个核心功能的选择上面，建议你可以这样思考：
- “肯定”核心一定要的功能，直接编译进核心内；
  - “可能在未来会用到”的功能，那么尽量编译成为模块；
  - “不知道那个东西要干嘛的，看 help 也看不懂”的话，那么就保留默认值，或者将他编译成为模块；

下面只列出几个鸟哥认为比较重要的设置项目。其他更详细的核心功能项目，自行参考 help 的说明。

### ➤ **General setup**

与 Linux 最相关的程序互动、核心版本说明、是否使用发展中程序码等信息都在这里设置的。这里的项目主要都是针对核心与程序之间的相关性来设计的，基本上，保留默认值即可！不要随便取消下面的任何一个项目，因为可能会造成某些程序无法被同时执行的困境！不过下面有非常多新的功能，如果你有不清楚的地方，可以按 <Help> 进入查阅，里面会有一些建议！你可以依据 Help 的建议来选择新功能的启动与否！

### ➤ **loadable module + block layer**

要让你的核心能够支持动态的核心模块，那么下面的第一个设置就得要启动才行！至于第二个 block layer 则默认是启动的，你也可以进入该项目的细项设置，选择其中你认为需要的功能即可！

### ➤ **CPU 的类型与功能选择**

进入 “Processor type and features” 后，请挑选你主机的实际 CPU 形式。

### ➤ **电源管理功能**

如果选择了 “Power management and ACPI options” 之后，就会进入系统的电源管理机制中。其实电源管理机制还需要搭配主板以及 CPU 的相关省电功能，才能够实际达到省电的效率啦！不论是 Server 还是 Desktop 的使用，在目前电力不足的情况下，能省电就加以省电吧！

### ➤ **一些总线（bus）的选项**

这个 “Bus options (PCI etc.)” 项目则与总线有关啦！分为最常见的 PCI 与 PCI-express 的支持，还有笔记本电脑常见的 PCMCIA 插卡啊！要记住的是，那个 PCI-E 的接口务必要选取！不然你的新显卡可能会捉不到！

### ➤ **编译后可执行文件的格式**

选择 “Executable file formats / Emulations” 会见到如下选项。下面的选项必须要勾选才行喔！因为是给 Linux 核心运行可执行文件之用的数据。通常是与编译行为有关啦！

### ➤ **核心的网络功能**

这个 “Networking support” 项目是相当重要的选项，因为他还包含了防火墙相关的项目！就是未来在服务器篇会谈到的防火墙 iptables 这个数据啊！所以，千万注意了！在这个设置项目当中，很多东西其实我们在基础篇还没有讲到，因为大部分的参数都与网络、防火墙有关！由于防火墙是在启动网络之后再设置即可，所以绝大部分的内容都可以被编译成为模块，而且也建议你编成模块！有用到再载入到核心即可啊！

### ➤ 各项设备的驱动程序

进入“Device Drivers”这个是所有硬件设备的驱动程序库！因为较符合一般状态，所以核心额外的编译进来很多跟你的主机系统不符合的数据，例如网卡设备～你可以针对你的主板与相关硬件来进行编译。不过，还是要记得有“未来扩充性”的考虑！

### ➤ 文件系统的支持

文件系统的支持也是很重要的一项核心功能！因为如果不支持某个文件系统，那么我们的Linux kernel 就无法认识，当然也就无法使用啦！例如 Quota, NTFS 等等特殊的 filesystem。这部份也是有够麻烦～因为涉及核心是否能够支持某些文件系统，以及某些操作系统支持的 partition table 项目。我们常常用到的网络操作系统（NFS/Samba 等等），以及基础篇谈到的 Quota 等，你都得要勾选，否则是无法被支持的。如果你有兴趣，也可以将 NTFS 的文件系统设置为可读写看看啰！

### ➤ 核心骇客、信息安全、密码应用

“Kernel hacking”项目，那是与核心开发者比较有关的部分，这部分建议保留默认值即可，应该不需要去修改他！然后下面有个“Security Options”，那是属于信息安全方面的设置，包括 SELinux 这个细部权限强化模块也在这里编入核心的！这个部份只要记得 SELinux 作为默认值，且务必要将 NSA SELinux 编进核心即可，其他的细部请保留默认值。另外还有“Cryptographic API”这个密码应用程序接口工具选项，以前的默认加密机制为MD5，近年来则改用了 SHA 这种机制。不过，反正默认已经将所有的加密机制编译进来了，所以也是可以保留默认值啦！都不需要额外修改就是了！

### ➤ 虚拟化与函数库

虚拟化是近年来非常热门的一个议题，因为计算机的能力太强，所以时常闲置在那边，此时，我们可以通过虚拟化技术在一部主机上面同时启动多个操作系统来运行，这就是所谓的虚拟化。Linux 核心已经主动的纳入虚拟化功能喔！而 Linux 认可的虚拟化使用的机制为KVM（Kernel base Virtual Machine）。至于常用的核心函数库也可以全部编为模块！

现在请回到画面中，在下方设置处移动到“Save”的选项，点选该项目，在出现的窗口中确认文件名为 .config 之后，直接按下“OK”按钮，这样就将刚刚处理完毕的选项给记录下来了。

总结一下我们的安装步骤：

- |  |
|--|
| <pre>1、获取内核源码，解压至/usr/src # tar xf linux-3.13.5.tar.xz -C /usr/src # ln -sv /usr/src/linux-3.13.5 /usr/src/linux  2、配置内核特性(选择一种方法就可以了) make config: 遍历选择所要编译的内核特性 make allyesconfig: 配置所有可编译的内核特性 make allnoconfig: 并不是所有的都不编译</pre> |
|--|

```
make menuconfig: 这种就是打开一个文件窗口选择菜单  
make kconfig(KDE 桌面环境下, 并且安装了 qt 开发环境)  
make gconfig(Gnome 桌面环境, 并且安装 gtk 开发环境)
```

### 3、编译内核

```
# make [-j #] : #号最多为 CPU 物理核心总数的两倍, 加上核心数会加快编译速度
```

### 4、安装内核模块

```
# make modules_install
```

### 5、安装内核

```
# make install
```

### 6、验证并测试

```
# cat /boot/grub/grub.conf
```