20211212-系统领域-龙芯固件-调研报告-王金龙

相关术语

缩写	全称	描述
PMON	**P**rom **MON**itor	固件名称 - “程序监视器”

一 、问题

基于龙芯 CPU 的计算机主板，绝大多数使用的固件是由 **PMON** 演化而来，然而由于 **PMON** 的市场占有率以及普及率较低，导致很多工程技术人员对该固件比较陌生，本篇文章主要介绍 **PMON** 的由来原理以及使用方法和策略。

- 软硬件环境

硬件环境	Arch	Mem	Hdd
	loongarch64	8 GB	256 GB

软件环境	OS	Compiler
	UnionTech OS 20	gcc-8.3.0(8.3.0-6-lnd.vec.23)

- 期望结果

  通过本文的介绍，可以使读者对 **PMON** 有一个更为完整的认识。

二 、现状

2.1 PMON 简介

PMON (**P**rom **MON**itor)：它是一个开源的全功能固件。全功能是指它除了在加电的时候初始化硬件，也可以直接载入内核和文件系统以及直接通过命令读写物理内存地址，调试程序等。类似 **BIOS** 兼 **Bootloader** 和 **Debugger** 的部分功能，**PMON** 具有强大丰富的功能，包括硬件初始化、操作系统引导和硬件测试、程序调试等功能。它提供多种加载操作系统的方式，可以从 U盘、光盘、TFTP服务器和硬盘等媒介加载；它提供对内存、串口、显示、网络、硬盘等的基础测试工具，此外，它还支持空中升级。

2.1.1 PMON

1988 年 6月 MIPS 公司推出32位芯片组 R3000, 实现了 MIPS I 指令集架构，是 MIPS 旗舰处理器的第二个 MIPS 实现。随后 LSI 公司使用 R3000 芯片组推出了 LSI Logic MIPS R3000 开发板。Phil Bunce 为这块开发板，开发了 **PMON** 的最初版本。而随后 **PMON** 被 NEC 等 MIPS 设备广泛使用，甚至在 Linksys WRT54G 以及 Asus WL-300g 这样的路由器上使用。不幸的是 **PMON** 的官方主线版本自 2000 年之后就不再更新，旧有的代码停留在 1999 年的版本。并且代码的网址 http://www.PhilBunce.com/pmon 也已经失效。然而幸运的是 2001 年英国公司 Algorithmics 在旧版

PMON 的基础上加入了 R4000-style 扩展和 GNU Make system，以至于我们还能找到最接近于 1999 年版本的 PMON 源码 pmonsrc-20011116.tar.gz.

2.1.2 PMON-2000

鉴于 **PMON** 的主要代码自 2000 年以来就再也没有官方更新维护。一家瑞典的商业公司 Opsycon，基于 **FreeBSD** 对 **PMON** 进行了开发维护，并且添加了许多 **BSD** 的条款，渐渐的变成了非官方的后继版本，以至于后期有人会误以为 **PMON** 的初始版本来源于早期的 **BSD** 内核。同样也由于年代久远，应用场景较少，**PMON-2000** 的 主页 PMON-2000 也已经失效。

2.1.3 PMON-Loongson

由于龙芯和 **MIPS** 的紧密关系，龙芯家族的大部分设备均使用在 **PMON-2000** 的基础上再开发的固件，龙芯在接手 **PMON-2000** 后引入了自己的版本编号。现在 **PMON** 与开始时的代码已经有较大不同了，看代码注释可以知道里面有一些中科院计算所和中科龙梦的人往里面添加了代码，就目前而言仅仅只有龙芯一家在使用 **PMON** 的路上继续坚持。

龙芯版本	版本简介	源码地址
初始	只知道基于 PMON-2000，具体代码网址不可考证。	
1.x	遗产版本，目前可以找到曾经由龙芯梦兰维护的版本，但二进制发布文件大都已经丢失。	https://github.com/kisom/pmon
2.x	遗产版本，短暂支持过，仅比 1.x 增加了部分 3A/3B 设备的支持。	
3.x	应该是属于主线版本，但目前已经失效。	http://www.loongnix.org/cgit/pmon-loongson3/
3.0	遵循 “固件与内核接口规范” 的基础版本，2011年10月20日发布。	
3.1	支持 smbios，2012年10月30日发布。	
3.2	适配龙芯的2.6.32内核以及主线内核，针对内存映射等部分做了相应的调整。	
3.3	使用新的 PMON 交叉编译器gcc4.4的版本，2014年7月10日发布。	
	龙芯广州分公司为龙芯俱乐部提供的龙芯1C，1B，智龙v2/v3 主板的 PMON。	https://github.com/lshw/loongson1-pmon
	网络资源目前可找到的最新版本。	https://gitee.com/brep/pmon-loongson3
5.0	龙芯提供的 PMON 源码，以下案例均以此为例，版本为：5.0.0-Release 。	龙芯提供

后续进行介绍演示和编译所使用的源码是经龙芯官方授权提供的最新 **PMON** 版本。

2.1.4 PMON 源码结构

```
uos@uos-VB:~/PMON/pmon-loongarch$ ls -al
总用量 9156
drwxr-xr-x 14 uos uos    4096 8月  17 19:40 .
drwxr-xr-x  3 uos uos    4096 9月  23 03:23 ..
-rwxr-xr-x  1 uos uos     177 6月   2 09:19 cmd.sh
drwxr-xr-x  2 uos uos    4096 5月   6 10:09 conf
-rw-r--r--  1 uos uos    1999 5月   6 10:09 Copyright
drwxr-xr-x  5 uos uos    4096 5月   6 10:09 doc
drwxr-xr-x  5 uos uos    4096 5月   6 10:09 examples
drwxr-xr-x  3 uos uos    4096 5月   6 10:09 fb
drwxr-xr-x  5 uos uos    4096 6月   9 10:41 include
drwxr-xr-x  6 uos uos    4096 5月   6 10:09 lib
-rw-r--r--  1 uos uos    2426 5月   6 10:09 Makefile
-rw-r--r--  1 uos uos    6891 5月   6 10:09 Makefile.inc
drwxr-xr-x 10 uos uos    4096 5月   6 10:09 pmon
drwxr-xr-x 12 uos uos    4096 5月   6 10:09 sys
-rw-r--r--  1 uos uos 9296382 6月   2 16:32 tags
drwxr-xr-x  3 uos uos    4096 5月   6 10:09 Targets
drwxr-xr-x  8 uos uos    4096 5月   6 10:09 tools
drwxr-xr-x  4 uos uos    4096 5月   6 10:09 x86emu
drwxr-xr-x  4 uos uos    4096 8月  17 09:37 zloader
lrwxrwxrwx  1 uos uos       7 5月   6 10:09 zloader.3a5000_7a -> zloader
```

从上述的源码目录命名可以简单做如下理解：

路径名	内容简述
cmd.sh	一键编译的脚本文件，手动配置添加不同选项
conf	源代码编译所依赖的配置文件所在目录
Copyright	版权许可信息
doc	内容手册，说明文档
examples	示例程序，如 hello 等
fb	帧缓存目录，存放 PMON 可视界面中的元素，图、表、进度条等
pmon	主体源代码
sys	较低层级的代码
Targets	存放每个开发板的配置文件，配置所有外设
tools	一些工具
x86emu	x86 显卡模拟器，主要是运行显卡的 BIOS，初始化显卡
zloader	生成 bin 文件的环境，zip 格式加载启动代码

- 目录 Targets

  龙芯提供的 **PMON** 源码中只有这一公板的配置信息，所以以这一公板为引，来简介 **PMON** 源码中偏底层的 Targets 目录的结构。此处仅列举部分目录，有兴趣的读者可以自行查看其它目录：

```
uos@uos-VB:~/PMON/pmon-loongarch/Targets/ls3a5000_7a$ tree -L 1
.
├── acpi_tables
├── cache_stage
├── compile
├── conf
├── dev
├── include
├── loongson
└── pci

8 directories, 0 files
```

- dev 目录下一些板子特殊的设备的驱动。
- conf 目录下是一些编译环境建立需要的一些文件。

```
uos@uos-VB:~/PMON/pmon-loongarch/Targets/ls3a5000_7a/loongson$ ls -al
总用量 108
drwxr-xr-x  2 uos uos  4096 7月  12 11:05 .
drwxr-xr-x 10 uos uos  4096 5月   6 10:25 ..
-rw-r--r--  1 uos uos 17083 5月   6 10:09 ht_link.c
-rw-r--r--  1 uos uos  3830 5月   6 10:09 loongson3_clksetting.S
-rw-r--r--  1 uos uos  7488 5月  11 16:47 loongson3_def.h
-rw-r--r--  1 uos uos  7503 5月   6 10:09 loongson3_ht1_32addr_trans.S
-rw-r--r--  1 uos uos 10366 5月   6 10:09 ls3a5000_vctrl.S
-rw-r--r--  1 uos uos   712 5月   6 10:09 resume.c
-rw-r--r--  1 uos uos  1875 5月   6 10:09 resume.S
-rwxr-xr-x  1 uos uos 15224 5月   6 10:09 start.S
-rwxr-xr-x  1 uos uos 20755 6月   9 10:39 tgt_machdep.c
```

- start.S 起点汇编文件
- tgt_machdep.c 一些板子相关的函数。

三、技术方案

3.2 PMON 编译

3.2.1 本地编译

3.2.1.1 环境配置

```
# 系统平台环境配置（Loongnix GNU/Linux 20 Beta9 + loongarch64）
sudo apt install bison flex xutils-dev
```

3.2.1.2 获取源码

```
# 来自龙芯授权提供，版本为：5.0.0-Release。
```

3.2.1.3 编译流程

- 手动编译

```
cd ($PMON)/tools/pmoncfg/
make
sudo cp pmoncfg /usr/local/bin/
cd ($PMON)/zloader.3a5000_7a/
make cfg
make tgt=rom    # 生成用于flash的可烧写的 gzrom.bin
make tgt=ram    # 生成用于可网络加载的 gzram.bin
```

- 自动编译

```
cd ($PMON)/
./cmd.sh
cd ($PMON)/zloader.3a5000_7a/ # 该目录下生成 bin 文件。
```

3.2.1.4 固件烧写

详见 4.1.3 和 4.1.5 小结的内容。

3.2.2 交叉编译

3.2.2.1 环境配置

```
# 系统平台环境配置（UnionTech OS GNU/Linux 20 + amd64）
sudo apt install iasl xutils-dev

# 配置交叉编译工具链
sudo tar xvf loongarch64-linux-gnu-2021-06-19.tar.gz -C /opt
export PATH=/opt/loongarch64-linux-gnu-2021-06-19/bin:$PATH
```

3.2.2.2 获取源码

```
# 来自龙芯提供，版本为：5.0.0-Release。
```

3.2.2.3 编译流程

- 手动编译

```
cd ($PMON)/zloader.3a5000_7a/
make cfg
make tgt=rom    # 生成用于flash的可烧写的 gzrom.bin。
make tgt=ram    # 生成用于可网络加载的 gzram.bin。
```

- 自动编译

```
cd ($PMON)/
./cmd.sh
cd ($PMON)/zloader.3a5000_7a/ # 该目录下生成 bin 文件。
```

3.2.2.4 固件烧写

详见 4.1.3 和 4.1.5 小结的内容。

四、实验验证

4.1 PMON 使用

PMON 在使用操作上提供了多重交互接口：串口，网口，FB 字符界面，还有鼠标参与的图形界面。

4.1.1 图形界面

系统上电，完成基本的初始化后，会有等待进入pmon图形界面设置的提示，一般会等待3秒钟，当用户按下[DEL]键，会再次进入 **PMON** 的图形设置界面。此外，还可以在 **PMON** 的字符控制界面输入 main 命令进入 **PMON** 的图形设置。

- SystemInfo 页

  是启动 PMON 图形界面的显示的第一页。给出了时间、CPU名及其频率、内存大小、基本的指令和数据Cache大小、MAC地址等等信息。右面一栏给出简单提示，屏幕的最下方给出不同按键的功能。在该页中，可以对时间进行调整，点击进入时间和日期调整，进入其调整的二级页面，输入新的时间和日期，保存退出。

- BOOT 页

  在该页中可以指定系统从哪个盘启动，3a系统提供6个接口，可以挂在sata盘、光盘等等不同介质，同时也可以选择从usb启动内核。

- Safe 页

  该页为设置pmon密码，输入密码并保存后，下次启动bios图形界面时需要输入命令才可进入。注意：请谨慎记住该密码，否则请联系生产厂商。

- Exit 页

  该页提供四个选项：保存退出、不保存退出、退回到PMON、自动加载内核。在图形界面里有所改动、需要保存的话，需要在这次进行保存。

4.1.2 字符界面

在进入系统前按 “c” 进入 **PMON** 的命令行。

```
PMON>
```

4.1.3 操作命令

命令	功能
about	显示 **PMON** 的一些信息
boot	启动命令，主要用来初始化 cpu、mem、pci 等
cdinstall	从 CD_ROM 光盘安装系统
compare	对比两块内存
control + c	结束当前进程
copy	从源地址 src 拷贝count大小的内容到目的地址dst；例如：PMON>memcpy 0x8030000 0x80400000 2 即为从0x8030000拷贝2个byte的 内容到0x80400000 的地址上
devls	列出当前所有设备,显示设备标识符
dir	可以列出这个分区的根目录信息，分区号从 0 开始计数
d[1-8]	d1 xxx：在地址 xxx 处读取一个 byte 的内容，d2 读取一个 half word 的内容，d4 读取两个 word 的内容，d8 读取一个 double word 的内容，显示内存信息
erase_dtb	擦除 dtb
eset	编辑环境变量
env	查看环境变量
eval	打印计算结果
fdisk	显示存储器分区信息
flush	刷新 cache
flash	对 flash 进行编程，可以把 size 大小的 data 烧写到给定的设备上或擦除更改 flash 的内容
fill	向内存里填充数据
date	get/set date and time 给出当前时间，后加日期则会系统设置时间，其格式为：date 年月日时分秒，如：date 20110211143308 即为设定当前时间为2011年2月11日14点33分8秒。
g	执行程序
h	列出当前 pmon 可以使用的命令
/h xxx	查看具体命令的用法
halt	挂起待机状态,挂起
hi	查看当前用户的历史命令
load_dtb	烧写dtb
ifaddr	设置网卡地址
initrd	载入初始化内存盘

命令	功能
load	载入文件， 默认不加参数载入内核，加参数载入其他文件， 2016 年后开始支持两种写法
ls	调试器的列出符号（List Symbols）命令
lsdev/devls	列出pmon探测到的所有网络，磁盘，USB存储设备
main	图形界面
m[1-8]	查看内存内容. m1 xxx：在地址 xxx 处写入一个 byte 的内容，m2 写入一个 half word 的内容，m4 写入两个 word 的内容，m8 写入一个 double word 的内容。modify memory 修改内存信息。simple memory test 对内存做一个简单测试
more	paginator 一屏显示不下时，more显示更多的内容，如按enter可以一行一行下翻，‘/’可以匹配字符串，n显示n行
mtdparts	查看 mtd 分区
mtd_erase	擦除 mtd 分区
partls	列分区命令,可以把磁盘中的分区信息打印出来，显示分区号，分区开始和结 束的扇区号，大小，文件系统和分区类型
pcicfg	排除 pci 的空白区间
pciscan	pci 设备扫描，列出总线上所有挂在的设备
ping	测试网络，查看网络是否在线，是否通畅。
poweroff	直接关机
print_dtb	查看 dtb 信息
reboot	重启系统
search	搜寻内存
set	显示已设定的变量或设定变量
sh	shell 相应命令输入
sleep	睡眠
spacescan	测试所有的空余内存
unset	取消环境变量
usbcdinstall	使用 usb 把 CD_ROM 连接到主板上，然后从CD_ROM里面安装系统
usbinstall	从 USB 设备安装系统
usb	查看及设置 USB 设备
vers	打印版本信息

PMON 的命令均可以在源码中进行重映射，增加或删除，这一点在龙芯的后续版本中有体现；龙芯对 PMON 中的命令做了许多扩充，PMON 中的热键键值也可以进行重映射，进入 PMON 命令行模式的热键，还有一些其他功能快捷键。

4.1.4 更新固件

```
# $FILE_LOCATION 随更新介质变化
PMON> load -r -f 0x1c000000 $FILE_LOCATION # 0x1c000000 for loongarch64
```

- 本地更新

  ```
  # 这里的 $FILE_LOCATION 是 gzrom.bin 的分区位置
  # 假设 PMON 更新文件 gzrom.bin 位于该分区的 /boot 目录下:
  PMON> load -r -f 0x1c000000 /dev/fs/ext2@wd0/boot/gzrom.bin     # 硬盘 ext2 格式的分区
  PMON> load -r -f 0x1c000000 /dev/fs/ext2@usb0/gzrom.bin         # U盘 ext2 格式的分区
  PMON> load -r -f 0x1c000000 /dev/fs/fat@usb0/gzrom.bin          # U盘 fat 格式分区
  PMON> load -r -f 0x1c000000 /dev/fs/iso9660@cd0/boot/gzrom.bin  # 光驱
  PMON> load -r -f 0x1c000000 /dev/fs/iso9660@usb0/boot/gzrom.bin # usb 光驱
  ```

- 在线更新

  ```
  # 这里的 $FILE_LOCATION 是网络协议可解析的 URL
  PMON> load -r -f 0x1c000000 http://xxx.xxx.xxx/gzrom.bin
  PMON> load -r -f 0x1c000000 tftp://xxx.xxx.xxx.xxx/gzrom.bin
  ```

4.1.5 固件烧录

- 使用专用的芯片烧录器工具进行固件烧写。

4.1.6 程序调试

- 主板调试

  ```
  # 改变 PMON 源码中关于主板外设的资源，比如 GPIO，USART，IIC，SPI 等。编译成二进制文件，直接载入执行。
  # $FILE_LOCATION 可执行的二进制文件，本地或在线均可。
  # $VIRTUAL_ADDRESS RAM 地址。
  PMON> load $VIRTUAL_ADDRESS $FILE_LOCATION
  PMON> g $VIRTUAL_ADDRESS
  ```

- 内核调试

  ```
  # 由于内核有一定的权限访问硬件资源，所以可以在 PMON 中调试。
  PMON> load $VIRTUAL_ADDRESS $FILE_LOCATION
  PMON> g $VIRTUAL_ADDRESS
  ```

4.2 PMON 启动

4.2.1 启动流程

通用计算机系统启动流程：

01	02	03	04	05	06
上电跳转	启动固件	引导加载	内核映像	文件系统	用户服务

PMON 的位置交叉包含了 阶段2 和 阶段3 。系统上电，待电压部署稳定，CPU 会在第一时间从虚拟地址为 0x1c000000 (loongarch64) 的虚拟地址入口处读取指令执行 ，随后 **PMON** 会完成初始化 cpu，内存，总线等设备及对串口、键盘、鼠标等外设进行基础测试等一些列工作，过程大致为：找到第一块硬盘的第一个分区、读取这个分区的 `boot.cfg` 文件，显示菜单、`load` 选择菜单项的内核和初始化内存盘，直接进操作系统。

注意，MBR 看起来被优雅地绕过了；但是，这个分区表还得是 MBR 格式的。**PMON** 并不认 GPT 硬盘——换句话说，**PMON** 确实读取了 MBR，但是没有直接执行 MBR 的前 448 字节代码，因为它本身就可以充当 Bootloader 的角色。所以，要用 PMON 引导系统，首要的条件是硬盘上的第一个分区有 `boot.cfg`，和别的嵌入式平台没什么两样——第一个分区是 `/boot` 就可以了。但是，PMON 只认两种文件系统，一种是 `vfat`，另一种是 `ext2/3`。注意，**`ext4` 不受支持**。

4.2.1.1 基本初始化

正如前述 2.1.4 小节中所描述的源码结构，现在进入 Targets/ls3a5000_7a/loongson/：

```
uos@uos-VB:~/PMON/pmon-loongarch/Targets/ls3a5000_7a/loongson$ ls -al
总用量 108
drwxr-xr-x  2 uos uos  4096 9月  23 05:35 .
drwxr-xr-x 10 uos uos  4096 5月   6 10:25 ..
-rw-r--r--  1 uos uos 17083 5月   6 10:09 ht_link.c
-rw-r--r--  1 uos uos  3830 5月   6 10:09 loongson3_clksetting.S
-rw-r--r--  1 uos uos  7488 5月  11 16:47 loongson3_def.h
-rw-r--r--  1 uos uos  7503 5月   6 10:09 loongson3_ht1_32addr_trans.S
-rw-r--r--  1 uos uos 10366 5月   6 10:09 ls3a5000_vctrl.S
-rw-r--r--  1 uos uos   712 5月   6 10:09 resume.c
-rw-r--r--  1 uos uos  1875 5月   6 10:09 resume.S
-rwxr-xr-x  1 uos uos 15224 5月   6 10:09 start.S
-rwxr-xr-x  1 uos uos 20755 6月   9 10:39 tgt_machdep.c
```

start.S 是 C 环境建立之前的汇编代码，是整个 **PMON** 运行的起点。

- 在这里定义了子程序的名称 _start 和 start。并定义了堆栈的栈底 stack 值。

```
 53         .globl  _start
 54         .globl  start
 55         .globl  __main
 56 _start:
 57 start:
 58         .globl  stack
 59 stack = start + LOCK_CACHE_SIZE         /* Place PMON stack in the end of
RAM */
```

- 这里配置串口：

```
590 /************************************
591  *used: a0~a1
592  ************************************/
```

```
593 LEAF(initserial)
594         dli     a0, PHYS_TO_UNCACHED(0x1fe001e0)
595
596         dli     a1, 0x80
597         st.b    a1, a0, 3
598
599 #if 1
600 #ifdef BONITO_100M
601         /* divider, highest possible baud rate, for 100M crystal*/
602         dli     a1, 0x36
603 #else
604         /* divider, highest possible baud rate, for 25M crystal*/
605         dli     a1, 0x0d
606 #endif
607 #else
608         /*33M*/
609         dli     a1, 0x12
610 #endif
611         st.b    a1, a0, 0
612         /* divider, highest possible baud rate*/
613         dli     a1, 0x0
614         st.b    a1, a0, 1
615         dli     a1, 3
616         st.b    a1, a0, 3
617
618         dli     a1, 0
619         st.b    a1, a0, 1
620
621         dli     a1, 71
622         st.b    a1, a0, 2
623         jirl    zero, ra, 0
624 END(initserial)
625
```

- 这里拷贝 flash 代码到缓存

```
420         /* copy flash code to scache */
421         dli     a1, PHYS_TO_CACHED(0x1c000000)
422         la      a0, start
423         la      a2, edata
424 1:
425         ld.d    a3, a1, 0
426         st.d    a3, a0, 0
427         addi.d  a0, a0, 8
428         addi.d  a1, a1, 8
429         bne     a2, a0, 1b
430
```

start.S 文件基本包含了整个 **PMON** 启动流程，上面仅仅列举了三个主要的部分，第一段代码引用关于 *堆栈配置*，第二段关于*串口配置*，第三段关于*代码从 flash 到 RAM 的搬运*，这是三个最为经典的过程。

4.2.1.2 加载内核

PMON 从固件层面支持多种引导方式：从硬盘启动，从 USB设备启动，以及从网络启动（也就是说 PMON 是可以初始化网卡并联网的）。对于硬盘上的一个特定分区的特定文件，PMON自2016年底开始支持以上两种语法格式，在 PMON 中有两种表达方式。

- 优盘加载

```
# 旧的表达方式
# (usbX,Y)/path/to/file
# X 表示第 X 块硬盘；Y 表示第 Y 个分区

# 新的表达方式
# /dev/fs/<FSTYPE>@usbX/path/to/file
# X 表示第 X 块硬盘； <FSTYPE> 表示分区类型，比如 ext2，vfat，iso9600

load (usb0,0)/vmlinuxboot
load /dev/fs/ext2@usb0/vmlinuxboot
```

- 光盘加载

```
# 旧的表达方式
# (cdX,Y)/path/to/file
# X 表示第 X 块硬盘；Y 表示第 Y 个分区。

# 新的表达方式
# /dev/fs/<FSTYPE>@cdX/path/to/file
# X 表示第 X 块硬盘； <FSTYPE> 表示分区类型，比如 ext2，vfat，iso9600

load (cd0,0)/boot/vmlinuxboot
load /dev/fs/iso9660@cd0/boot/vmlinuxboot
```

- 硬盘加载

```
# 旧的表达方式
# (wdX,Y)/path/to/file
# X 表示第 X 块硬盘；Y 表示第 Y 个分区。

# 新的表达方式
# /dev/fs/<FSTYPE>@wdX/path/to/file
# X 表示第 X 块硬盘； <FSTYPE> 表示分区类型，比如 ext2，vfat，iso9600

load (wd0,0)/boot/vmlinuxboot
load /dev/fs/iso9660@wd0/boot/vmlinuxboot
```

- 网络加载

```
ifaddr rte0 AD1.AD2.AD3.AD4 # 设置 NIC 地址
devls # 列出 NIC 设备
ping AD1.AD2.AD3.AD4
load tftp://AD1.AD2.AD3.AD4/vmlinuxboot
```

4.2.1.3 加载内存盘

单纯的加载入内核，并不能构成一个完整的系统环境，需要载入基本的内存文件系统镜像，提供最小的“基本系统“。**PMON** 载入的所有文件都通过 load 命令执行。

4.2.2 配置文件

PMON 会优先读取第一个磁盘第一分区根目录下的 boot.cfg 文件，所以要确保它的存在。对于硬盘启动来说配置文件位于第一个硬盘的第一个分 区根目录下，命名为 boot.cfg，且该分区文件系统格式为 ext2。对于光盘启动来说配置文 件则位于光盘根目录下，命名为 boot.cfg。下面就是 boot.cfg 的内容：

```
default 0
timeout 3
showmenu 0



title Loongnix 20 GNU/Linux 4.19.167-rc5.lnd.1-loongson-3
    kernel /dev/fs/ext2@/vmlinuz-4.19.167-rc5.lnd.1-loongson-3
    initrd /dev/fs/ext2@/initrd.img-4.19.167-rc5.lnd.1-loongson-3
    args console=tty loglevel=0 locales=zh_CN.UTF-8 splash quiet
root=UUID=79686f4d-1f55-4987-a705-5651f6030ec4 resume=PARTUUID=853aeb1b-03
```

为了防止引导失败不能进系统，最好加入grub的引导，在上述文件后补充上：

```
title grub

kernel (wd0,0)/grub.efi(此处指向你的/boot/grub.efi文件)

args nil
```

重启后即会看到俩个引导菜单，第一个是使用 PMON 直接引导系统内核，第二个是用 GRUB 间接引导。值得注意的是，不是所有版本的内核都可以直接用PMON 引导的，有的版本确实会出各种问题，所以个人还是推荐用 GRUB 引导。

五、小结

PMON 固件功能强大，但是受众面有限普及率较低，事有利弊，自主可控是一大助力。

六、参考资料

1. https://www.linux-mips.org/wiki/PMON
2. https://www.linux-mips.org/wiki/PMON_2000
3. https://www.linux-mips.org/wiki/Fulong
4. http://www.embeddedlinux.org.cn/BuildLinuxSystem/belinuxsys-CHP-9-SECT-1.html
5. https://wiki.godson.ac.cn/pmon
6. https://github.com/loongson-community/pmon
7. https://github.com/lshw/loongson1-pmon

8. https://zh.wikipedia.org/wiki/R3000
9. https://en.wikipedia.org/wiki/LSI_Corporation
10. https://zh.wikipedia.org/wiki/MIPS%E6%9E%B6%E6%A7%8B
11. https://en.wikipedia.org/wiki/MIPS_architecture
12. https://www.yhi.moe/blog/cn/pmon