黑苹果教程

# 认识功能强大的Clover引导（1）–《手把手教你玩黑苹果》

2019年3月18日 by **Mison**

目录

# 功能强大的Clover引导

## （一）前言

什么是Clover（三叶草）呢？显然它不是指的草地里用来喂牛的草啦。Clover是一个软件，是一个新型的启动器，它能够让普通的PC上用上Mac OS X系统。

苹果公司（Apple）限制Mac OS X系统只能在Apple设备上使用，并且苹果不保证Mac OS X在其它设备上能够正常工作。所以，用户需要承担一定的风险。当然，为了避免其它的法律纠纷，你不应该用作商业用途。装上了Mac OS X的非苹果电脑，就叫做黑苹果(Hackintosh)。

为了开始黑苹果，你需要一个特殊的启动器。现在有很多启动器，它们可以分成两类：仿冒EFI和真实EFI。

#### 启动器类型

##### 仿冒EFI

它是很多年前David Ellion发明的。工作流程如下：它假设EFI已经完成了相应的工作，并在内存里以一种简单的形式保留活动痕迹（boot-arg和tables tree）和EFI运行时程序，然后启动内核*mach_kernel*。举例来说，Chameleon(变色龙)就是这种启动方式，它会产生一些负面影响，比如使启动磁盘选项面板不工作。正是这种没有运行时服务的工作方式让苹果有机会从中作难。2013年1月的情况是，因为缺少SetVariable()函数，iMessage停止运行。

### 真实EFI

对于BIOS主板而言，理论上需要将EFI刷入主板来代替BIOS，但是实际上只需要一个可以加载的EFI即可。EFI启动方式由Intel发明，现在它是TianoCore.org上一个活跃的开源项目，启动器的名字叫*DUET*。那么问题来了，它能够加载EFI但是它并不是用来启动Mac OS X的。所以我们还需要修改DUET，使之能够启动Mac OS X。较新的主板已经包含了EFI，但它并不适合运行黑苹果。

### 名字的来源

启动器的名字*Clover*由一位创建者kabyl命名。他发现了四叶草和Mac键盘上Commmand键的相似之处，由此起了Clover这个名字。

四叶草是三叶草的稀有变种。根据西方传统，发现者四叶草意味的是好运，尤其是偶然发现的，更是祥瑞之兆。另外，第一片叶子代表信仰，第二片叶子代表希望，第三片叶子代表爱情，第四片叶子代表运气。

## （二）技术背景

### EFI

EFI（The Extensible Firmware Interface，可扩展固件接口）是位于操作系统和硬件固件之间的一个软件接口。相对于BIOS那最多64kb的可用空间和运行于16位处理器模式，EFI大小可以达到4MB，运行于32位或64位模式，并且理论上是平台无关的。但实际上想要达到全平台兼容是不可能的。

### Clover

Clover是一个操作系统启动加载器(boot loader)，能够同时运行于支持EFI方式启动的新式电脑和不支持它的老式电脑上。一些操作系统可以支持以EFI方式启动，比如OS X, Windows 7 64-bit, Linux；

也有不支持的，比如Windows XP，它只能通过传统的BIOS方式来启动，也就是通过启动扇区来启动。

EFI不仅存在于操作系统的启动过程中，它还会创建操作系统可访问的表和服务(tables and services)，操作系统的运行依赖于EFI正确的提供功能。从内建的UEFI来启动OS X是不可能的，用原始的DUET来启动OS X也不可能。CloverEFI和CloverGUI做了大量的工作来修正内部表，让运行OS X成为可能。

注：DUET(Developer’s UEFI Emulation)，开发者的UEFI模拟

## Clover的任务

- 设置SMBIOS (DMI)信息来模拟一台真实的Apple电脑 – 这对于运行OS X系统是必需的。序列号是伪造的，但是有效的。
- ACPI表 – 包含在电脑的固件中 – 通常没有被正确的编写而且可能包含bugs，这大多是因为制造商的懒惰：APIC表中包含错误的CPU核心数量，没有NMI数据，FACP表中没有重置寄存器(reset register)，错误的电源配置，SSDT表中没有EIST

  数据，甚至没有DSDT表。Clover试图去修正这些问题。
- 接着OS X会从boot loader获取用来描述附加的设备如显卡、网卡或声卡的数据，这些数据就是所谓的EFI字符串(EFI Strings)。Clover产生这些数据。
- 基于BIOS的电脑在启动初期，USB运行于旧的模式(legacy mode)，当控制权传递给操作系统时，会产生问题。Clover负责改变USB的运行模式。
- OS X使用一块特殊的称为NVRAM的内存来进行信息交换，它被包含在运行时服务(RuntimeServices)中（在启动的初期不

存在，not present in a legacy loader）。Clover提供这种信息交换，使Firewire功能正常，“启动磁盘”设置面板也可正常使用。附加的NVRAM也用于注册iCloud和iMessage服务。

- ConsoleControl协议是必要的并且DUET中没有.
- 通过DataHub协议为EFI/Platform填充某些必要的信息，这些信息DUET中没有，而UEFI中也不一定会有。而且会设置非常重要的FSB频率值(FSBFrequency value)，因为这个值有时是错的或者根本就没有。
- CPU在工作前必须被正确地初始化，但是主板为了兼容大量的不同的CPU，内部表中不包含任何正确的CPU数据。Clover会对安装的CPU做一个完全的检测，修正这些表中的CPU信息。其中的一个效果就是CPU加速模式(turbo mode)能够工作。
- 还有一个小问题：DUET和EDK2源码是为了兼容不同的硬件而编写的，但不同的硬件依赖于不同的常量。这意味着每个编译对应一个特定的平台。Clover提供了自动平台检测。

## （三）基本概念

Clover支持两种启动方式，启动过程如下：

**启动方式A**

**基于BIOS的电脑（老式主板）**

BIOS>MBR>PBR>boot>CLOVERX64.efi>OSLoader

**启动方式B**

**基于UEFI的电脑（新式主板）**

UEFI>CLOVERX64.efi>OSLoader

Mac OS X的操作系统加载器（OSLoader）是boot.efi， Windows的则是bootmgr.efi。

## 两种启动方式所涉及到的文件或目录（即本章的内容）

### 基本概念

- MBR 扇区
- PBR 扇区
- CloverEFI的启动
- CloverGUI与EFI文件
- 目录结构
- EFI驱动程序

### MBR 扇区

扇区位于存储设备（传统硬盘、固态硬盘、USB记忆棒、USB硬盘、DVD）的起始位置。前440个字节可能包含这些不同引导程序的的某一种:

- boot0 – 查找MBR中的活动分区并将控制权交给它的PBR扇区。也有可能是GPT和MBR的混合式布局。如果是纯GTP磁盘分区布局，那么控制权将交给EFI分区。进一步移交 boot0af (首先激活).
- boot0hfs – 查找第一个签名为0xAF的分区。如包含OS X的HFS + 的分区 并将控制权交给它的PBR。按照这种方式系统将

HFS＋的分区,并将控制权交给它的PBR。按照这种方式系统将从以GPT磁盘分区架构设备的HFS＋分区中启动，并且只能从第一个分区启动。进一步移交给 boot0ss (扫描签名).

- boot0ab – 查找签名为0xAB的分区 – Apple启动分区.
- boot0md – 结合所有情况查找多个设备上的HFS+分区, 并非主存储设备。

**PBR 扇区**

在一个存储设备的每个分区的起始位置有一个引导扇区。第二阶段的加载器存储在这里。它知道分区的文件系统并且能够找到 boot 文件，加载该文件并传递控制权。根据文件系统存在不同的引导程序：

- boot1h2 – 支持 HFS+ 文件系统，具有大小高达 472kb 的引导文件。与 “Chameleon” 捆绑在一起的老的 boot1h 引导文件只支持大小为 440kb 的引导文件（需要 472kb）。停顿两秒来提供切换加载器的选项。
- boot1h – 与上面相同，不包含停顿。
- boot1f32alt – 支持 FAT32。这个文件系统具有写入能力并且非常适合于安装启动加载器。可以将它用在一个EFI分区上或者用在 USB 闪存驱动器上，它通常销售时已经预先格式化为 FAT32。此引导程序也提供两秒的停顿。
- boot1f32 – 与上面相同，不包含停顿。

此外，它们还提供另一个有用的功能。在这两秒的停顿中可以按下键盘上的一个数字来启动一个特定的加载器。按下键 1 将启动文件 boot1 ，按下键 3 将启动文件 boot3 而且按下键 6 将启动文件

boot6 。这种方式可以在一个位置保存一组加载器。例如：

- boot – Clover，最新版本或者测试版本 * boot1 – Chameleon
- boot3 – Clover 32位，已测试并且工作的版本 * boot6 – Clover 64位，已测试并且工作的版本 * boot7 – Clover 64位，带有 BiosBlockIO 驱动，可以与 BIOS 支持的任一控制器工作。

除了这些引导程序，PBR 可能包含 Windows 引导管理器（能够处理 NTFS），GRUB （能够处理 EXT4）以及其它与 Clover 不相关引导程序 – 至少目前是这样。

## CloverEFI的启动

Chameleon（变色龙）的启动文件是一个完整的启动加载器，而Clover包含整个EFI系统和一个传递控制权给下一个阶段的启动服务（选项A），或者Clover可以包含在PC的ROM内（选项B）。实际情况中并不总是这样，有些部分需要额外加载。相比于之前的阶段，启动文件有不同的位深，也就是32位和64位的不同。如果CPU支持64位指令集，通常建议选择64位。如果你使用的是32位的操作系统，就只能通过32位的EFI来加载。32位的EFI体积小20%，速度稍快，但不能和Windows 7 EFI兼容，因为Windows 7 EFI只与64位的EFI兼容。通常，启动文件只在DUET的基础上修改了不到1%。尽管如此，这1%发挥了大作用——Clover正是在这一部分实现它的功能。如果有足够的空间把AppleSim加入到DUET，那真是极好的！（译注：难道是引导iOS吗？）但是到现在为止都没有成功。定义了特定PCD常量的DUET可以编译给特定电脑使用，但是Clover需要为大家所用。这个问题已经解决，但我并不想解释其中的细节。后来这个项目变成了我们所说的CloverEFI。

## CloverGUI与EFI文件

CloverGUI存在于CloverIA32.efi和CloverX64.efi中，是Clover的图形化界面。图形化界面里可以选择一个操作系统，更改设置，加载额外的驱动，当然最终要加载操作系统。图像和菜单基于rEFIt项目，相关的目录名称和关于界面中有提到这个项目。当前原始的部分大概占了整个工程的10%。

## 目录结构

Clover启动程序（就是上文提到的CloverGUI）需要一些支持文件，它们的目录结构如下(以Clover r2334为例):

- EFI:
    - BOOT:
        - BOOTX64.efi
    - CLOVER:
        - ACPI:
            - origin:
            - patched:
                - DSDT.aml
            - WINDOWS:
                - SLIC.aml
        - CLOVERIA32.efi
        - CLOVERX64.efi
        - themes:

- black_green:
    - WoB_PTMono_10W_NA.png
    - icons:
        - func_about.png
        - os_clover.icns
        - os_unknown.icns
    - banner.png
    - background.png
    - Selection_big.png
    - Selection_small.png
    - theme.plist
- frame.png
- logo_main.png
- logo_metal.png
- pointer-green.png
- pointer-metal.png
- Selection_64px.bmp
- Selection_144px.bmp

- config.plist
- drivers32:
    - FSInject-32.efi

- drivers64:
    - FSInject-64.efi
    - NTFS.efi
- drivers64UEFI:
    - CsmVideoDxe.efi
    - DataHubDxe.efi
    - HFSPlus.efi
    - NTFS.efi
    - OsxAptioFixDrv-64.efi
    - OsxFatBinaryDrv-64.efi
    - PartitionDxe-64.efi
- kexts:
    - 10.6:
    - 10.7:
    - 10.8:
    - 10.9:
    - Other:
- misc:
- OEM:
    - SystemProductName:
        - ACPI:

- origin:
- patched:

- config.plist
- kexts:
  - 10.6:
  - 10.7:
  - 10.8:
  - Other:
- UEFI:
  - ACPI:
    - origin:
    - patched:
  - config.plist
  - kexts:
    - 10.6:
    - 10.7:
    - 10.8:
    - Other:
- ROM:
- tools:
  - Shell32.efi

- Shell64.efi
- Shell64U.efi

也就是CloverX64.efi要放在/EFI/CLOVER/目录下，字体文件WoB_PTMono_10W_NA.png要放在/EFI/CLOVER/themes/black_green/目录下。通常类似的这些目录中还包含更多其它内容。各个文件的具体用途将在后面说明。

## /EFI/CLOVER/OEM/目录的用途

这个目录用来存储不同的配置信息。典型的情况就是加载一个可启动的USB闪存驱动器。除了通用的配置文件/EFI/CLOVER/config.plist，还可以在OEM目录下包含针对特定机型的其它配置文件，比如：
/EFI/CLOVER/OEM/Inspiron 1525/config.plist，或者
/EFI/CLOVER/OEM/H61M-S1/UEFI/config.plist外加自定义DSDT.aml

OEM目录下的子目录名称是根据SMBIOS得到的，可以在preboot.log日志文件中查到。当你在Clover启动界面按下F2时，日志文件preboot.log就会保存到/EFI/Clover/misc目录下。以下内容就是在Inspiron 1525上运行Clover r1709得到：

```
10:061  0:000  Clover revision: 1709  running on
Inspiron 1525
10:061  0:000  ... with board 0U990C
```

如果是组装的台式电脑，显示信息会有点不同。以下内容是在技嘉

如果是组装的台式电脑，显示信息会有点不同。以下内容是在技嘉Z68MX-UD2H-B3主板上（主板BIOS中没有定义SystemProductName）运行Clover r2334得到的：

```
0:100  0:000  Clover revision: 2334  running on To be filled by O.E.M.
0:100  0:000  ... with board Z68MX-UD2H-B3
```

第一行包含了系统名称(SystemProductName)，第二行包含的是主板型号。笔记本可以由系统名称来识别，组装电脑可以由主板型号来识别。你可以选择任意一种识别方式，只要方便就行。另外可以在配置目录(/EFI/CLOVER/OEM/SystemProductName/)下包含一个UEFI目录来区分UEFI启动（选项A）和常规启动（选项B）。

### EFI驱动程序

BIOS启动过程中（选项A）要用到drivers32或drivers64目录，UEFI启动过程中（选项B）则使用driversUEFI目录。它们的内容会根据配置和BIOS版本而有所不同。

必须要提的一点是这些驱动程序只在bootloader运行时有效，不会影响最终启动的操作系统。

至于到底要使用哪些驱动程序由用户来决定。

#### NTFS.efi

NTFS文件系统驱动程序。（一般不使用，此模块与一些主板的UEFI不兼容）

#### HFSPlus.efi

HFS+文件系统驱动程序。这个驱动对于通过启动方式B来启动Mac OS X是必须的。启动方式A中用到的启动程序(CloverEFI)已经包含了这个驱动。（推荐使用）

**VBoxHFS.efi**

HFSPlus.efi的替代品，性能要差一点。（不推荐使用）

**VBoxExt2.efi**

EXT2/3文件系统驱动。用于启动Linux EFI系统。（需要引导Linux系统的使用）

**VBoxExt4.efi**

EXT4文件系统驱动。用于启动Linux EFI系统。（需要引导Linux系统的使用）

**FSInject.efi**

控制文件系统注入kext到系统的可能性。

**PartitionDxe.efi**

已经存在于在CloverEFI和UEFI中，但没有为Apple分区优化，也没有为GPT/MBR优化。很可能（选项B）需要。

**OsxFatBinaryDrv.efi**

允许加载FAT模块比如boot.efi。（选项B）需要。

**OsxAptioFixDrv.efi**

修复AMI Aptio EFI内存映射。如果没有就不能启动OS X。（6代以前机器推荐使用）

### OswLowMemFix.efi

是OsxAptioFixDrv的简化版。两个不能同时使用。（一般不使用）

### Usb*.efi, UHCI.efi, EHCI.efi, XHCI.efi

用以解决依赖性关系不满足导致的内建驱动工作不正常的情况（选项B）的一组驱动。（一般不使用）

### PS2Mouse.efi, PS2MouseAbsolute.efi, UsbMouse*.efi

用来使鼠标，触摸板在CloverGUI界面工作的一组驱动，它们对操作系统没有影响。（一般不使用）

### DataHubDxe.efi

已经存在于在CloverEFI中，可能存在于UEFI中。建议还是使用它，不会产生冲突。

### CsmVideoDxe.efi

比UEFI里提供更多分辨率的显卡驱动（选项B）（6代以前核显开CSM使用）

### ApfsDriverLoader-64.efi

基于逆向工程Apple的ApfsJumpStart驱动程序的开源apfs.efi加载器。它链从此容器加载已嵌入APFS容器中的apfs.efi驱动程序。（推荐使用它，而不是apfs.efi）

- 从位于块设备上的APFS容器加载apfs.efi。
- Apfs驱动程序详细日志记录被抑制。
- 版本系统：将每个apfs.efi连接到检索它的设备。
- 链式加载apfs.efi驱动程序的嵌入式签名验证，可防止可能的植入注入。

### AppleImageLoader-64.efi

使用EfiBinary签名验证实现AppleLoadImage协议，保护AppleEfiFat二进制驱动程序。它通过预先验证其签名，将Apple EFI二进制文件安全加载到内存中。(一般不使用)（一般不使用）

### AppleUiSupport-64.efi

实现用于FileVault作为登录窗口的支持EfiLoginUi的协议集的驱动程序。简而言之，它实现了FileVault支持并替换了AppleKeyMapAggregator.efi，AppleEvent.efi，AppleUiTheme.efi，FirmwareVolume.efi，AppleImageCodec.efi。此外，它还包含一些主板的哈希服务修复和unicode排序规则。这些修复程序已从R23中的AptioMemoryFix中删除。（一般不使用）

### AppleEfiSignTool

用于验证Apple EFI二进制文件的开源工具。它支持ApplePE和AppleFat二进制文件。（一般不使用）

### AppleDxeImageVerificationLib

该库为EFI二进制文件提供Apple的加密签名算法。（一般不使用）

### AptioInputFix

参考驱动程序，用于为File Vault 2 GUI输入支持提供AMI APTIO专有鼠标和键盘协议。
通常修改的UsbKbDxe在APTIO V上工作得更好，但对于Z77，Z87和类似的AptioInputFix可能是唯一的解决方案。（一般不使用）

### AptioMemoryFix

原始OsxAptioFix2驱动程序的分支，具有更清晰（但仍然可怕）的

代码库，并提高了稳定性和功能。（6代及以上电脑推荐使用）

重要说明：要调试10.13上的boot.efi错误，除了通常的详细（-v）boot-arg之外，您可能需要boot.efi 补丁或nvram首选项。有必要获得合理的错误消息而不是’printf work ??’，但补丁可能有助于您在阅读nvram时遇到问题。

在使用AptioMemoryFix之前，请确保您拥有：

最新的UEFI BIOS固件（请查看您的主板供应商网站）。

如果存在，则在BIOS中禁用快速启动和硬件快速启动。

如果存在，则在BIOS中启用4G解码或类似功能。
警告：在某些主板（特别是华硕WS-X299-PRO）上，此选项会产生不利影响，必须禁用。虽然没有其他具有相同问题的主板，但请考虑此选项，首先检查您是否有不稳定的启动失败。

如果存在，则在BIOS中禁用VT-d（您也可以使用引导加载程序删除ACPI DMAR表）。

slideNVRAM或其他任何地方都没有引导参数（除非您根本无法启动或看到No slide values are usable! Use custom slide!来自AptioMemoryFix的消息，否则没有必要）。

如果存在，则在BIOS中禁用CFG锁定（MSR 0xE2写保护）（如果您有足够的技能，请考虑修补它）。有关详细信息，请参阅VerifyMsrE2部分。

如果存在CSM，则在BIOS中禁用CSM（您可能需要在NVIDIA 6xx / AMD 2xx或更旧版本上使用GOP ROM，使用GopUpdate或AMD

UEFI GOP MAKER可以简化操作）。

EHCI / XHCI 仅在启动停止时才在BIOS中启用切换功能，除非USB设备断开连接。

如果存在，则在BIOS中启用VT-x，超线程，执行禁用位。

虽然可能不需要，但有时您必须在BIOS中禁用Thunderbolt支持，Intel SGX和Intel Platform Trust（如果存在）。在调试睡眠问题时，您可能希望（暂时）禁用Power Nap和自动关闭电源，这有时会导致在旧平台上唤醒黑屏或引导循环问题。具体问题可能有所不同，但一般情况下，您应首先检查ACPI表。以下是一些Z68主板中发现的错误示例。要关闭Power Nap和其他命令，请在终端中运行以下命令：

```
sudo pmset autopoweroff 0
sudo pmset powernap 0
sudo pmset standby 0
```

请注意，这些设置可能会在硬件更改时以及在某些其他情况下重置。要查看其当前值运行pmset -g。

请注意，如果您在详细模式下看到许多开始/结束行，则必须确保没有bootercfg存储在NVRAM中的变量。有关详细信息，请参阅此帖如果它仍然没有帮助，您可以在终端中使用以下命令（禁用系统完整性保护或从Recovery HD）：

```
sudo nvram bootercfg="log=0 debug=0"
```

# （四）开发

## 许可证

由于许可问题，该项目没有任何商业价值。而且它太大了，不能由一个人开发。合乎逻辑的决定是用开源来开发它，这样任何人都可以为它做出贡献。目前它位于存储库 http://cloverefiboot.sourceforge.net中的 SourceForge上。

然而，开源项目具有一定的局限性，可以说是“许可”。特别是，该项目使用为Linux应用程序开发的GPL。然而，有一个缺点。英特尔不建议使用它。

## 过程

小题外话 – 在创建这样一个大项目时，必须遵循以下几点：

- 收集特定任务的文档，数据表，规范，应用程序示例。我有一个很好的起点 – 变色龙，其中“一切”都有效（强调引号）。然而，随着时间的推移，新的处理器，新的视频卡，新Mac

  OS X版本的新要求被创建，并且还需要进行更多的研究和测试。非常感谢Kabyl系统化这一起点。

- 当遇到给定输入和所需输出的问题时，有必要创建一个算法，该算法最小化，快速，无错误且稳定。这些任务深受程序员的喜爱，这是在这个项目中值得拥有大部分学分的助手。特别是我想提一下dmazar和гык-sse2，他们解决了一些非常棘手的问题。

- 现在是时候进行持续而无聊，而不是在编写千行代码的过程中，并不是很困难。只需复制和粘贴更正。这是部分，对不

起，几乎没有人帮助过我。你的项目 – 你做的工作。另外感谢apianti帮助我在这里。

- 测试和错误修复如下。在这里，我有很多帮助者，不可能记住它们 – 仅在applelife.ru上发布了20,000个帖子。测试从“无效”的简单抱怨到代码更改的详细说明不等。两种变体都很有用，甚至是抱怨，因为它鼓励人们想出一个停止的解决方案（;如果实际问题得到解决则无关紧要 – 抱怨需要停止！）。

- 最糟糕的错误是一个主要问题。半年来，我解决了三个初始问题。1.旧系统上的内核恐慌，2。睡眠不足，3。笔记本电脑启动失败。以下问题与dmazar和pene一起解决：4。iCloud，5.iMessage。还有两个问题还有待解决：在UEFI启动时睡眠时重新启动，同时在某些配置上使用传统视频BIOS和传统操作系统无法启动。我没有看到任何志愿者。

- 此外，值得一提的是：编译脚本，安装程序，系统脚本和与项目直接相关的附加组件，它们不是引导加载程序的一部分。所有这些巨大的工作都超出了我的能力，即使我开始了。大多数工作都是由jadran，crazybirdy，JrCs甚至是apianti完成的。

- 最后，必须记录软件。同样，这是一项乏味且不是特别有趣的工作。而且，似乎没有人关心。

其余的开发人员和帮助程序在源代码和安装程序中都有提及。

我写这一部分希望可能有其他志愿者愿意参与这个项目，为此你需要至少能够编译。

## 从源代码编译

要编译项目，您需要一个编译器和库 – 这是不言而喻的。在这种特

要编译项目，您需要一个编译器和库 – 这是不言而喻的。在这种特殊情况下需要做些什么？这些库由http://sourceforge.net/projects/edk2上的EDK2环境覆盖。由于该项目是为Hackintosh创建的，因此将在Mac OS X下的编译过程中设置焦点。但是，这不是唯一的选择。EDK2专为在Windows，Linux等不同系统上进行编译而设计。Windows需要Visual Studio 20xx，Mac OS X需要Xcode命令行工具，Linux需要gcc。预先捆绑的软件仍然不足以在Mac下成功编译。建议 – 与Linux相同 – 使用Clover目录中的脚本“buildgcc.sh”构建新版本的gcc。为什么不铿锵？ – 它仍然不会产生任何工作代码。但是，我们不要放弃希望。
现在到了真正的交易。

**下载源代码并准备环境**

```
cd ~
mkdir src
cd src
svn co  -r 18198
svn://svn.code.sf.net/p/edk2/code/trunk/edk2 edk2
cd edk2
make -C BaseTools/Source/C
svn co svn://svn.code.sf.net/p/cloverefiboot/code
Clover
cd Clover
```

**构建编译器。GCC-4.9。这是可以进行LTO优化的编译器。**

```
./buildgettext.sh
./buildgcc-4.9.sh
./buildnasm.sh
```

**使EDK2环境适应我们的需求**

./edksetup.sh

cp -R Clover/Patches_for_EDK2/* ./

1-3A。如果您已经执行了所有这些步骤并且只想更新Clover，则无需重复此步骤。做就是了：

```
cd Clover
svn up
```

1.现在可以构建CloverEFI。例如这样：

```
./ebuild.sh
./ebuild.sh -mc
./ebuild.sh --ia32
```

其他编译脚本可用，通常记录在案。看，选择并自己决定，你想要使用哪一个。

2.小注意：存储库中没有HFSPlus.efi。有两个选项：您可以在其他地方找到它或更改项目定义文件.fdf以使用开放驱动程序VboxHfs而不是私有驱动程序。它有点慢，有一些缺点，将来可能会纠正，但

不是私有驱动程序。它有点慢，有一些缺点，将来可能会纠正，但它是功能性的。更改：

```
# foreign file system support
#INF  Clover/VBoxFsDxe/VBoxHfs.inf
INF  RuleOverride=BINARY Clover/HFSPlus/HFSPlus.inf
```

至：

```
# foreign file system support
INF  Clover/VBoxFsDxe/VBoxHfs.inf
#INF  RuleOverride=BINARY Clover/HFSPlus/HFSPlus.inf
```

3.这个项目并没有停滞不前，这就是为什么这个指令可能会因为一些小的变化而变得过时。这个项目适合那些思考和能够找出错误和做什么的人。

4.构建安装程序包

```
cd ~/src/edk2/Clover/CloverPackage/
./makepkg
./makeiso
```

就这样！如果重复执行此过程，可能会遗漏一些步骤。例如，您可以发出“svn up”而不是再次重新下载整个项目，排除32位编译过程并跳过编译。新用户可以使用自动脚本CloverGrower及其Pro版本，但是他们应该首先考虑使用预制的安装程序包。
现在，随着一切准备就绪，您可以进行安装。

# （五）安装

## 使用安装程序

安装程序是为什么做的？安装程序！为什么安装人员可以更准确地完成所有工作。唯一的要求是安装Mac OS X. 一种变体是使用另一个引导加载程序引导安装程序DVD并启动安装程序。根据操作系统语言，安装程序将以您的首选语言显示，可能是俄语，英语或中文。目前，安装程序被翻译成17种语言，包括印尼语 – 谁知道，有人可能需要它。以下说明适用于英语。

clover_installer_1

按**继续**，阅读信息，再次**继续**。现在，您可以选择安装的内容，位置和原因。

clover_installer_2

**更改安装位置...** – 选择安装Clover的位置

clover_installer_3

在上一页上**自定义** – 引导加载程序选项

clover_installer_4

单击选项时，在下部文本字段中可以获得详细说明

单击选项时，在下部文本字段中可以获得详细说明。

**安装UEFI主板** – 此选项禁用安装启动文件。他们似乎非常困扰某人。

**在ESP中安装Clover** – 这种EFI分区可用时的最佳选择（GPT分区驱动器）。安装程序看不到此分区，您需要将其指向同一驱动器上的另一个分区，该分区包含ESP。假设Mac OS X在此分区上，我们将安装RC脚本和首选项面板。

**引导加载程序** – 当使用CloverEFI或UEFI引导时（选项B），这适用于BIOS引导（选项A）* **不更新MBR和PBR扇区** – 不要更新这些扇区，因为它们已经存在或者不需要选项B * **在MBR中安装boot0af** – 使用boot0af启动，即搜索活动分区。先前选择的分区将被标记为活动。* **在MBR中安装boot0ss** – 使用boot0ss启动，即搜索HFS +分区，即使它不活动。不会设置活动标志。适用于Windows的双启动配置，其中Windows需要位于活动分区上。**用户替代启动PBR-r**，可以将PBR扇区设置为暂停键盘输入1-9。此选项启用暂停。

**用户替代启动PBR** – 如在“（三）基本概念”，可以将PBR扇区设置为暂停键盘输入1-9。此选项启用暂停。

**CloverEFI** – 选择加载程序的位深度：32位或64位。此外还有一个**BiosBlockIO**选项。这是一个CloverEFI-64变体，带有一个特殊的boot7文件，适用于具有非标准SATA控制器的计算机。此驱动程序通过BIOS运行，通常可与任何控制器一起使用（因为BIOS需要支持它们！）。然而，它并不总是有效，例如Dell Inspiron 1525。

**Drivers64UEFI** – 具体的驱动程序在“（三）基本概念”部分中描述

clover_installer_5

**在目标卷上安装RC脚本** – 这些是在启动和退出OS X时执行的脚本rc.local和rc.shutdown.local。它们在Clover的概念中发挥了必要的作用。如果你没有计划使用Clover，你可能无法安装它们（那你在这里做什么呢？）。

**在所有其他可引导OSX卷上安装所有RC脚本** – 适用于安装多个Mac OS X系统的情况。安装程序将自动确定它们。例如，Windows和Linux系统将不受影响。

如果您知道自己在做什么，可以跳过脚本安装。

**可选的RC脚本** – 安装其他RC脚本* **禁用睡眠代理客户端** – ???

>> missing description <<

**安装Clover首选项窗格** – 这是一个首选项面板，可帮助更新Clover，选择主题并设置NVRAM变量。

>> missing pref panel screenshot <<

## 手动安装

它在两种情况下很有用：~~捕捉跳蚤或腹泻时~~首先你可能知道自己在做什么，想要控制每一步，不要相信安装人员（你应该！），其次是安装来自另一个无法运行安装程序的操作系统。

### OSX

如果您不熟悉终端，建议不要按照本节中的步骤操作。

#### 在HFS +分区上安装

安装将在位于MBR或混合GPT / MBR分区磁盘上的HFS +分区上进行。为何选择MBR？当磁盘已包含不得丢失的信息时，这是一种非常常见的情况。在这种情况下，只能设置新的引导加载程序。

#### MBR部门安装

```
cd BootSectors
sudo fdisk440 -f boot0 -u -y /dev/rdisk0
```

`fdisk440` – 特殊版本的fdisk实用程序，更正为仅使用引导扇区的440字节。对于与Apple版本未授予的Windows兼容性很有用

`boot0` – 文件，在“（三）基本概念”描述

`rdisk0` – 物理设备，用于安装。验证它确实编号为0

这些文件随Clover一起提供。

#### PBR部门安装

```
sudo dd if=boot1h2 of=/dev/rdisk0s9
```

`boot1h2` – HFS +文件系统的PBR扇区文件。支持大启动文件和使用数字键选择加载器。更多内容“（三）基本概念”

`rdisk0s9` – 所选设备上的第九个分区。为什么第九？避免因执

行编写命令的傻瓜造成的损坏。这样的分区可能不存在。需要调整到正确的数字，例如1以匹配第一个分区。

在所选设备和分区上写入MBR和PBR扇区后，您应该将分区标记为活动：

```
fdisk440 -e /dev/rdisk0
>f 9
>w
>q
```

第二行中的9 – 再次 – 表示分区号。做出结论。

现在，您可以将引导文件和EFI目录复制到此分区。

#### 在FAT32分区上安装

与之前的方法不同，有一个小问题。PBR扇区必须包含分区的几何。分区驱动器时会写入此数据，丢失它会产生后果。扇区安装方法变得更加复杂：

```
dd if=/dev/rdisk1s9 count=1 bs=512 of=origbs
cp boot1f32alt newbs
dd if=origbs of=newbs skip=3 seek=3 bs=1 count=87
conv=notrunc
dd if=newbs of=/dev/rdisk1s9 count=1 bs=512
```

`boot1f32alt` – 在“（三）基本概念”一节中提到。在FAT32分区上安装的扇区。不适用于FAT16，请记住这一点！

`rdisk1s9` – 再次是第一个设备上的第九个分区。替换为您自己

的数字。

*命令中的其余字母和数字已经过讨论，不作更改。其余命令类似于HFS +分区上的安装。*

### Linux

Linux也有一个终端，命令几乎相同。但是，安装可能只能在FAT32分区上进行。这些是差异：

- 而不是 `rdisk1` 使用 `sdb` Linux版本可能使用的其他定义
- 而不是 `fdisk440` 为MBR设置使用 `dd`

```
dd if=/dev/sdb count=1 bs=512 of=origMBR

cp origMBR newMBR
dd if=boot0 of=newMBR bs=1 count=440 conv=notrunc
dd if=newMBR of=/dev/sdb count=1 bs=512
```

### Windows

在Windows中，只有在FAT32分区的USB闪存驱动器上安装引导加载程序才有意义。启动脚本：

```
makeusb.bat E:
```

`E:` – USB闪存盘的字母

没有多个分区，毕竟它是Windows！

脚本所需的所有文件都与Clover捆绑在一起。但是，您需要先解压缩安装程序，或者从*edk2 / Clover / CloverPackage / CloverV2 / BootSectors*的存储库中获取文件。执行脚本后，您需要删除闪存驱动器并再次插入。然后复制*引导*文件和*EFI*目录。

更好的选择是使用cVaD的BootDiskUtility.exe，它将帮助您在Windows上格式化闪存驱动器。

# 设计

### 什么是主题

主题代表设计的元素：横幅，背景，图标，按钮，字体 – 所有这些都融为一体的意境。

### 选择一个主题

目前，某些主题与安装程序包捆绑在一起。您可以安装它们，然后设置您要使用的那个。

注意：在将来的某个时刻，主题将不再包含在安装程序中，而是Clover将附带一个嵌入式主题（它已经有）。

您现在可以通过使用终端手动下载它们或通过当前正在开发的OS X的新Clover主题管理器来从新的Clover主题存储库中获取主题。此Clover主题管理器可以直接从新的Clover主题存储库查看，安装/卸载和更新主题。

可以在此处找到Clover Theme Manager的源代码

可以在此处找到Clover Theme Manager的源代码。

您目前可以从以下论坛主题* insanelymac下载最新版本

您还可以在我们的主题数据库中快速浏览所有可用主题。然而，现在鼓励所有主题设计者使用Clover主题库来存在他们现有的和新的主题。

## 创建主题

### 文件类型

Clover的GUI可以加载和使用.png和.icns格式。

传统上，OS设备图标（例如，os_cougar.icns）使用Apple的ICNS格式，该格式将多个图像大小封装在一个文件中。使用带有Clover的.ICNS文件时，可以使用两种图像大小：128×128和32×32像素。单击以下链接到Metal主题的os_cougar.icns文件，例如：http：//bit.ly/1ipTskA

但是，Clover的.ICNS解码仅支持OS X 10.0中引入的图标类型。OS X 10.7引入了当前不支持的额外图标类型。例如，您可以在以下Wiki链接中看到从OS X 10.7及更新版本创建的128×128像素图像可以使用O07类型的ic07。这不受支持，因为即使Clover可以读取和解码嵌入的.PNG文件本身，它也不知道从ICNS文件中提取数据的新类型。http://en.wikipedia.org/wiki/Apple_Icon_Image_format

为了帮助解决这种情况，从Clover r2231开始，GUI现在也可以使用.PNG图像来表示OS设备图标。如有必要，Clover可以缩放.PNG图像，具体取决于用户/设计者对theme.plist中Badges-> Swap选项

的选择。

请注意：Clover仍然希望找到带有.icns文件扩展名的操作系统设备图标。因此即使图像可能是.png，它仍然需要具有.icns扩展名。

### 文件大小

完整的Clover主题可以包含许多图像，如果不加以控制，则可能导致主题目录的大小为兆字节。这对Clovers的打包安装程序产生了直接影响，该安装程序在去年大幅增长。因此，打包安装程序中的主题正在优化，因此文件可以尽可能小，同时保留原始设计。

设计师有责任保持图像的优化，并确保它们消除任何不必要的过量。为了帮助您实现这一目标，请阅读此insanleymac论坛主题，该主题是来自projectosx的原始帖子的基本概述，该帖子已不再可用。

### theme.plist

主题设置在theme.plist中定义，该主题设置位于主题目录中，并且对于每个主题都是唯一的。例如，*金属*主题的路径是*/EFI/CLOVER/themes/metal/theme.plist*。

在plist编辑器（推荐）或文本编辑器中打开现有的theme.plist将显示xml结构，如下所述。

### 描述

```
<key>Author</key>
<string>Slice</string>
```

```
<key>Year</key>
<string>2012</string>
<key>Description</key>
<string>Main metallic looking theme</string>
```

```
<key>Theme</key>
<dict>
```

**起源**

让Clover知道主题最初设计的屏幕分辨率。这样可以在以不同分辨率使用主题时重新计算动画位置。*注意，目前仅在Anime数组中使用较新的定位设置时使用（见下文）。

```
<key>Origination</key>
<dict>
    <key>DesignWidth</key>
    <integer>1920</integer>
    <key>DesignHeight</key>
    <integer>1080</integer>
</dict>
```

**背景**

```
<key>Background</key>
```

```
<dict>
    <key>Path</key>
    <string>MetalBack.png</string>
    <key>Type</key>
    <string>Crop</string>
    <key>Sharp</key>
    <string>0x80</string>
    <key>Dark</key>
    <true/>
</dict>
```

- `Path` – 背景图像的名称（或路径）
- `Type`
    - `Crop` – 剪切大图像以适合显示器尺寸或用纯色填充剩余空间而不重新缩放。
    - `Tile` – 平铺图像以填充区域。
    - `Scale` – 通过切割拉伸图像占据整个屏幕。

增加图像大小会产生方形像素，可以通过平滑将其最小化。但是，平滑会扭曲边缘。Clover将检测它们，以下参数允许微调：

- `Sharp`
    - 最小值 `0x0` – 无检测，边缘光滑。
    - 最大值 `0xFF` （255） – 完全检测，无平滑。
- `Dark`
    - `<true/>` – 明亮线条的暗图像。

- `<false/>` – 明亮的图像与暗线。

图像格式为PNG，最佳地具有正确的标题。例如，“预览”应用程序以正确的格式保存它们。

#### 徽章

徽章是加载程序条目右下角的小图标。最初的设计是在主图标（如Boot Camp）和徽章中的操作系统中显示磁盘。

```
<key>Badges</key>
<dict>
    <key>Show</key>
    <true/>
    <key>Inline</key>
    <true/>
    <key>Swap</key>
    <false/>
    <key>OffsetX</key>
    <integer>nn</integer>
    <key>OffsetY</key>
    <integer>nn</integer>
```

```
    <key>Scale</key>
    <integer>nn</integer>
</dict>
```

- `Show` – 显示徽章或不显示徽章
- `Inline` – 将徽章移动到所选加载器的信息行。`Swap` 在这种情况下被忽略。以*iClover*主题为参考。
- `Swap` – 交换图标和徽章的含义。图标将显示操作系统，徽章设备类型（在这种情况下显示它们并不感兴趣）。
- `Scale` – 将徽章的比例设置为原始大小的n / 16。示例（基于原始图像为128x128px）。值为5会导致缩放为40x40px，8缩放为64x64px，16不缩放，因此保留为128x128px。
- `OffsetX` – 在主菜单项内设置徽章的水平位置。
- `OffsetY` – 在主菜单项内设置徽章的垂直位置。

OffsetX和OffsetY设置OS设备选择区域左上角的上下两个像素数，并指示绘制徽章图标的位置。有效最低值为0，并且通过从OS设备选择区域（选择大）中减去徽章宽度（64px）来计算最大有效值。如果任一值大于结果，则自动设置结果的默认位置，并且引导日志将包含类似于“用户偏移X nn超出范围”的条目。

**旗帜**

横幅是一个居中的图像，具有有限的尺寸，具体取决于显示器尺

寸。例如，主题*黎明*有一个尺寸为672x190px的横幅，应该被视为限制。如果不使用背景图形，横幅应该是不透明的，其第一个像素将确定背景颜色。否则，您可以使用技巧并将第一个像素的透明度设置为1%。

```
<key>Banner</key>
<string>logo-trans.png</string>
```

更新：自rev2524（以及使用rev2534的更好位置）后，Banner现在可以选择使用用于动画的新位置设置。

```
<key>Banner</key>
<array>
    <dict>
        <key>Path</key>
        <string>logo_trans.png</string>
        <key>ScreenEdgeX</key>
        <string>left</string>
        <key>DistanceFromScreenEdgeX%</key>
        <integer>nn</integer>
        <key>ScreenEdgeY</key>
        <string>top</string>
        <key>DistanceFromScreenEdgeY%</key>
```

```
        <integer>nn</integer>
        <key>NudgeX</key>
        <integer>nn</integer>
        <key>NudgeY</key>
        <integer>nn</integer>
    </dict>
</array>
```

有关这些设置的说明，请参阅动画部分。

### 组件

组件是UI元素组，您可以指定主题使用的组。

```
<key>Components</key>
<dict>
  <key>Banner</key>
  <true/>
  <key>Functions</key>
  <true/>
  <key>Tools</key>
  <true/>
  <key>Label</key>
  <true/>
  <key>Revision</key>
  <true/>
```

```
  <key>MenuTitle</key>
  <true/>
  <key>MenuTitleImage</key>
  <true/>
</dict>
```

- `Banner` – 显示或隐藏横幅图像。
- `Functions` – 显示或隐藏系统功能，如“关于”，“重新启动”和“关机”。
- `Tools` – 显示或隐藏Shell等系统工具。
- `Label` – 显示或隐藏条目描述标签。
- `Revision` – 显示或隐藏Clover GUI修订版本。
- `MenuTitle` – 在帮助，关于和选项屏幕上显示或隐藏菜单标题。
- `MenuTitleImage` – 显示或隐藏帮助，关于和选项屏幕上菜单文本旁边绘制的菜单图标。

**选择**

```
<key>Selection</key>
<dict>
    <key>Color</key>
    <string>0xF3F3F380</string>
    <key>Small</key>
    <string>Select_trans_small.png</string>
    <key>Big</key>
    <string>Select_trans_big.png</string>
    <key>OnTop</key>
    <true/>
    <key>ChangeNonSelectedGrey</key>
```

```
    <true/>
</dict>
```

- `Color` – 菜单中的行选择颜色，最佳地匹配整体主题颜色。示例 `0x11223380`：red = 0x11，green = 0x22，blue = 0x33，alpha = 0x80。0x80的透明度（alpha）值为50%。0x00表示完全透明的选择，0xFF表示纯色。
- `Small` – 下GUI行中的小选项图标的选择图像。
- `Big` – 上部GUI行中的大OS图标的选择图像。
- `OnTop` – 如果为true，则将在主图像上绘制选择图像。
- `ChangeNonSelectedGrey` – 将所有未选定的设备图标及其相关徽章绘制为灰度。

注意：使用128×128像素设备图标时，选择大图像的大小为144×144像素，使用256×256像素设备图标时，大小为288×288像素。选择小图像的大小为64×64像素。

注2：完全支持尺寸为256×256像素的定位设备图标需要Clover r2707或更高版本。

**布局**

更改某些GUI元素的垂直位置。

```
<key>Layout</key>
<dict>
    <key>BannerOffset</key>
    <integer>nn</integer>
    <key>ButtonOffset</key>
```

```
    <integer>nn</integer>
    <key>MainEntriesSize</key>
    <integer>nn</integer>
    <key>TextOffset</key>
    <integer>nn</integer>
    <key>TileXSpace</key>
    <integer>nn</integer>
    <key>TileYSpace</key>
    <integer>nn</integer>
    <key>Vertical</key>
    <true/>
</dict>
```

- `BannerOffset` – 增加横幅下的空间（像素）。有效地在主屏幕上按下OS图标。
- `ButtonOffset` – 移动工具按钮的垂直位置（像素）。
- `MainEntriesSize` – 主设备图标的大小在代码内部设置为128.更改此值将指示Clover缩放设备图标。例如，在具有128px图标的主题上使用值256将显示缩放到256px的设备图标。注意：徽章不会受到影响，并且很可能需要重新定位。
- `TextOffset` – 移动主菜单文本的垂直位置（像素）；读取“从xxxx启动Mac OS X”的那个。
- `TileXSpace` – 每个主设备图标的Selection Big区域之间的间距。在内部，默认为8.因此，如果使用128×128像素图标并且144×144像素选择较大，则下一个设备图标的选择大区域将被绘制为前一个区域右侧144 + 8像素。
- `TileYSpace` – 主设备图标的“选择大”区域与较小工具按钮的“选择小区域”之间的间距。在内部，默认为24.注意，ButtonOffset设置也会影响此位置（需要进行验证）。

- `Vertical` – 垂直向下显示屏幕右边缘的操作系统图标。设置为true或false。（在rev2535中介绍）。

注意实验移动这些少量（即10个像素而不是100个像素，因为当以较小的分辨率观看时，这可以在屏幕上绘制）。

**字形**

```
<key>Font</key>
<dict>
    <key>Type</key>
    <string>Load</string>
    <key>Path</key>
    <string>BoG_LucidaConsole_10W_NA.png</string>
    <key>CharWidth</key>

    <integer>10</integer>
</dict>
```

- `Type` – 字体类型
    - `Alfa` – 内置字体
    - `Gray` – 内置字体
    - `Load` – 加载外部字体
- `Path` – 外部字体路径，例如 *BoG_LucidaConsole_10W_NA.png*
    - 以下约定存在文件名（blackosx）：
        - *BoG* – 黑色灰色
        - *LucidaConsole* – 原始字体名称
        - *10W* – 字体宽度

- *TOW* – 字体宽度
- *NA* – 无抗锯齿

- `CharWidth` – 字符宽度，默认为16

创建字体的一种方法是使用bash脚本和Imagemagick。在insanelymac上查看此主题

**滚动**

设置菜单可能大于显示器的实际垂直尺寸，在这种情况下会出现滚动条。

```
<key>Scroll</key>
<dict>
    <key>Width</key>
    <integer>N</integer>
    <key>Height</key>
    <integer>N</integer>
    <key>BarHeight</key>
    <integer>N</integer>
    <key>ScrollHeight</key>
    <integer>N</integer>
</dict>
```

- `Width` -
- `Height` -
- `BarHeight` -
- `ScrollHeight` -

**动画**

主题可能包含PNG格式的动画图像。

```
<key>Anime</key>
<array>
    <dict>
        <key>ID</key>
        <integer>1</integer>
        <key>Path</key>
        <string>logo_3D</string>
        <key>Frames</key>
        <integer>15</integer>
        <key>FrameTime</key>
        <integer>200</integer>
        <key>Once</key>
```

```
        <false/>
        <key>ScreenEdgeX</key>
        <string>left</string>
        <key>DistanceFromScreenEdgeX%</key>
        <integer>nn</integer>
        <key>ScreenEdgeY</key>
        <string>top</string>
        <key>DistanceFromScreenEdgeY%</key>
        <integer>nn</integer>
        <key>NudgeX</key>
        <integer>nn</integer>
        <key>NudgeY</key>
        <integer>nn</integer>
    </dict>
</array>
```

- `ID` – 确定动画类型和位置
    - `1` – 商标
    - `2` – 关于
    - `3` – 救命
    - `4` – 选项
    - `5` – 图形
    - `6` – 中央处理器
    - `7` – 二进制文件
    - `8` – DSDT
    - `9` – 启动顺序
    - `10` – SMBIOS

    - `11` – 桌子掉落
    - `12` – RC脚本变量
    - `13` – PCI设备
    - `14` – 主题
    - `21` – 苹果
    - `22` – WinXP
    - `23` – 幸运草
    - `24` – Linux
    - `25` – BootX64.efi
    - `26` – Vista
    - `30` – 恢复
    - `34` – 老虎
    - `35` – 豹

    - `36` – 雪豹
    - `37` – 狮子
    - `38` – 山狮
    - `39` – 天猫
- `Path` – 动画名称。指向包含动画序列的文件夹。在使用最后一个工作帧时，可以省略帧以创建暂停：
    - ML_Anim_000.png

    - ML_Anim_001.png
    - ML_Anim_008.png
    - ML_Anim_014.png
- `Frames` – 总帧数。将使用上述方法替换丢失的帧。
- `FrameTime` – 每个单帧之间的时间（以毫秒为单位）。
- `Once` – 循环设置
    - `<true/>` – 动画播放一次，直到使用*Esc*键退出菜单或按下鼠标右键。
    - `<false/>` – 动画无限循环，最后一帧和第一帧之间没有任何停顿。
- `ScreenEdgeX` – 用于计算的屏幕边缘。选项是左或右。
- `DistanceFromScreenEdgeX%` – 离屏幕边缘左侧或右侧的百分比以放置动画。值为整数。
- `ScreenEdgeY` – 用于计算的屏幕边缘。选项是顶部或底部。
- `DistanceFromScreenEdgeY%` – 远离屏幕边缘的顶部或底部放置动画的%。值为整数。
- `NudgeX` – 将水平位置微调+ -32像素范围。
- `NudgeY` – 将垂直位置微调+ -32像素范围。

# 配置

### 创建配置文件

Clover 可以根据硬件进行自动配置，但是自动配置组件并不总是完

美的。这也是保留用户可以自定义配置的原因。用户可以修改配置文件config.plist中的配置参数，或者基于GUI的配置界面进行修改配置。配置文件是基于XML的，可以以文本文件来处理。它可以用纯文本编辑器进行编辑，也可以用plist编辑器进行编辑，如PlistEdit。Clover 附带两种配置文件，一种包含所有的可能配置项，另一种包含所需的最少配置项。

配置文件 (config.plist) 必须放在EFI/CLOVER目录下。

**一些常用规则：如果你不知道参数的需求值是什么，请从配置文件中排除！不要有没有值的参数。 最后但并非不重要，如果你不明白值的参数，请不要设置一个值！**

可以通过下列方式来创建一个配置文件: * 安装最小版本, 仅有安全参数 * 加载Clover的界面，输入*Options*菜单（键盘按*O*） * 浏览所有选项，并了解设置，以及为什么设置 * 修复你所了解的，余下勿动 * 载入系统. 如果不能载入, 重复你最后的步骤只到启动

在系统中通过终端开启：

```
/usr/local/bin/clover-genconfig >config.plist
```

这种方法你会得到参数几乎完整并能成功加载操作系统的配置文件。一些调整工作而后完成。这里有不同的章节和参数说明:

### Config.plist结构

后面的内容请前往“认识功能强大的Clover引导（2）–《手把手教你玩黑苹果》”查看

← 认识黑苹果（Macintosh/OSx86）–《手把手教你玩黑苹果》

macOS Mojave 10.14.4【18E2034】原版带Clover+PE镜像 →

Written by

**Mison**

# One thought on “认识功能强大的Clover引导（1）–《手把手教你玩黑苹果》”

1.  **xhmooc**说道：

   2019年6月5日 下午7:43 编辑

   感谢，现在很少有人写这些，都在瞎摸，拼运气，这种文章值得一看

回复

## 发表评论

已登录为Mison。注销?

评论

发表评论

搜索…

## Audio Player

双击按钮播放=》

## 分类目录

选择分类目录

## 功能

- 管理站点
- 注销
- 文章RSS (Really Simple Syndication)
- 评论RSS (Really Simple Syndication)
- WordPress.org

## 友情链接

Dreamspark追风之人

## 标签

Mac系统镜像 (7) misonsky (1) 刷vBIOS (1) 变频 (1) 教程 (11) 编译驱动 (2) 脚本 (2) 驱动方法 (1)

粤ICP备19020771号-1 Copyright © 2019 云屋小站. 保留所有权利。