Hackintosh Hilfe in Berlin

Beitrag von „griven“ vom 27. März 2021, 21:08

Sehr gerne 

Ich mache mich später ran spätestens aber morgen denn aktuell sind wir im Teammeeting und da kann ich parallel nicht sinnvoll basteln aber wir bekommen das hin 

So, Meeting ist durch und eine mehr oder weniger erholsame Nacht später kann es nun also losgehen und wir können das Projekt "wie komme ich von einem alten Clover zu einem aktuellen OpenCore" angehen. Bevor wir loslegen können müssen wir natürlich erstmal alle Zutaten besorgen hier gehe ich persönlich immer so vor das ich zunächst eine Bestandsaufnahme mache und davon ausgehend entscheide was benötigt wird, was aktualisiert werden muss und was 1:1 übernommen werden kann. Fangen wir also mit dem Bestand an hier wäre also mal der IST Zustand aus den vorliegenden Informationen:

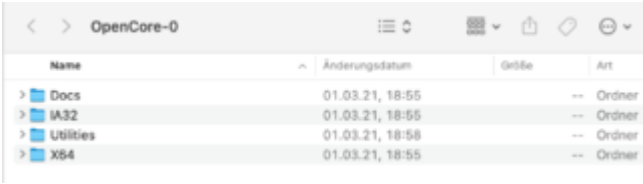
Bootloader: irgendeine ältere Clover Version

Extensions:

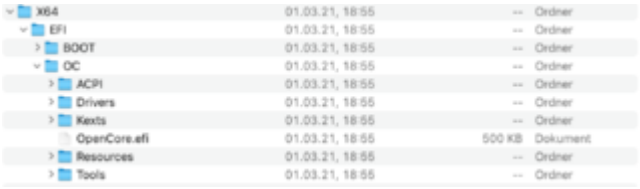
- FakeSMC nebst Plugins
- Lilu
- AppleALC
- IntelMausiEthernet
- UsbInjectAll
- WhatEverGreen

Ausgehend vom Ist Zustand entscheide ich nun was ich hiervon unverändert übernehmen kann was weg kann und was ggf. aktualisiert werden sollte und komme zu dem Ergebnis das es in diesem fall nicht schadet alles in der jeweils aktuellen Version neu zu besorgen bzw. eben einige Dinge auch zu ersetzen da es hier inzwischen bessere Alternativen gibt. Zum herunterladen der notwendigen Extensions bediene ich mich des KextUpdaters und wähle hier wie folgt aus:

Das Ergebnis der Aktion ist ein Ordner namens Kext-Updater der nun auf dem Desktop zum liegen gekommen ist und der alle Downloads aus dem Tool beinhaltet. Wie vielleicht auffällt habe ich die bisher verwendete FakeSMC aussortiert und an ihrer statt VirtualSMC ausgewählt der Rest ist 1:1 geblieben und liegt nun in der jeweils aktuell Form vor. Der Grund dafür das FakeSMC aussortiert wurde liegt darin das dieser Kext schon seit geraumer Zeit nicht mehr gepflegt wird und in virtualSMC eine aktuellere und weiterhin aktiv entwickelte Alternative gefunden hat. Bezogen auf die Extensions haben wir hier also schon mal alles zusammen was wir brauchen und somit geht es im nächsten Schritt an den Bootloader selbst. Ich habe mir hierzu angewöhnt die jeweils aktuell Release Variante von der Acidanthera Github Seite zu verwenden (https://github.com/acidanthera/OpenCorePkg/releases) zumindest dann wenn ich nicht an meiner eigenen EFI schraube alternativ kann man aber auch hierzu den KextUpdater bemühen der letztlich auch nichts anderes macht als die Release Version von Github zu laden. Egal woher man den Loader nun nimmt am Ende des Downloads steht eine Verzeichnisstruktur die wie folgt aussieht:



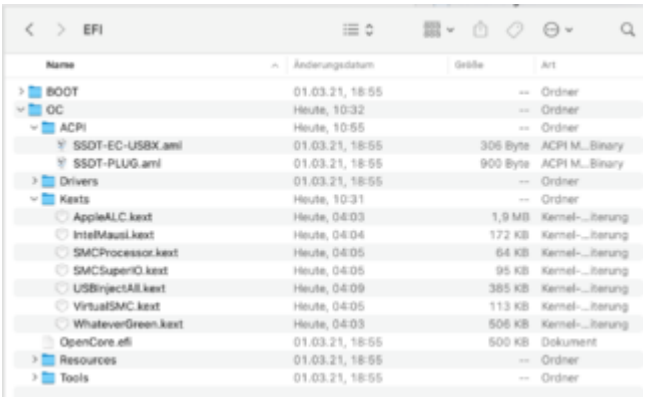
Die für uns nun relevanten Ordner sind die Ordner Docs und X64. Der Docs Ordner enthält neben der gesamten Dokumentation zu OpenCore auch alle nötigen Vorlagen im X64 Ordner finden wir eine Vorlage für unser künftiges EFI Verzeichnis sprich hier haben wir dann auch direkt die Struktur des Verzeichnisses vorgegeben und wir können schon mal damit anfangen die Struktur zu füllen.



Alle Extensions die wir heruntergeladen haben bzw. die wir direkt aus dem alten EFI Ordner übernehmen können wir also schon mal in den Ordner **Kexts** ziehen und haben damit schon einen guten Teil auf dem Weg hin zu OpenCore erledigt. Im nächsten Schritt geht es nun darum den Ordner ACPI mit Inhalt zu füllen hierzu muss man wissen das OpenCore anders als Clover von sich aus nichts am ACPI des Rechners verändert hier sind wir sind also selbst dafür verantwortlich alle notwendigen Fixes selbst einzubringen aber keine Sorge bei Desktop Systemen hält sich der notwendige Aufwand dabei in sehr engen Grenzen und darüber hinaus liefern die AcidAnthera Jungs auch alles Nötige direkt mit. Um zu ermitteln welche ACPI Fixes wir wirklich brauchen empfiehlt sich an der Stelle ein Blick in den schon mehrfach erwähnten Dortania Guide denn hier wird in Abhängigkeit zur eingesetzten Systemgeneration sehr übersichtlich aufgezeigt was nötig ist. In Deinem Fall brauchen wir für ein Z390 System folgende "Zutaten":

- SSDT-Plug.aml
- SSDT-EC-USBX.aml

die wir im Ordner Docs -> AcpiSamples -> Binaries finden und von dort in den Ordner X64/EFi/OC/ACPI kopieren (für den Moment soll uns das reichen und wir akzeptieren einfach das wir die Dateien benötigen ohne zu hinterfragen wozu sie notwendig sind). An der Verzeichnisstruktur haben wir nun fast alles Notwendige getan fehlt eigentlich nur noch ein Schritt nämlich das hinzufügen der Ressourcen sofern wir zum Beispiel die grafische Bootauswahl verwenden möchten. Die Ressourcen finden wir unter https://github.com/acidanthera/OcBinaryData wobei wir uns das Github Repo der Einfachheit halber einfach als ZIP herunterladen. Aus dem heruntergeladenen Ordner kopieren wir aus dem Ordner Recources die Ordner "Font", "Image" und "Label" und fügen sie in unsere EFI Ordnerstruktur in den Ordner Recources ein die Frage ob wir die bestehenden Ordner ersetzen möchten beantworten wir mit "ja für alle". Im Endergebnis sieht der EFI Ordner also bisher so aus:



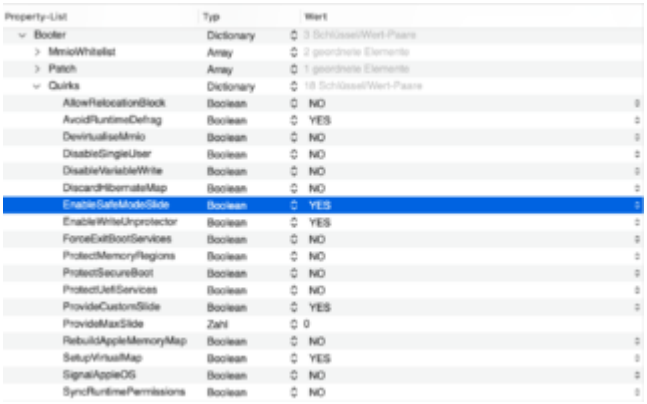
Damit sind die Vorbereitungen auch schon abgeschlossen und wir kommen zum spannenden Teil nämlich dem anpassen der config wobei uns auch hier ein Vorlage praktischerweise direkt mitgeliefert wird. Zum bearbeiten der config benötigen wir einen .plist Editor (auch wenn es grafische Editoren gibt empfiehlt es sich für OpenCore eher einen .plist Editor zu verwenden) hier gibt es unterschiedliche, sowohl kostenfreie als auch kostenpflichtige, zur Auswahl. Ich habe mich über die Jahre an PlistEditPro gewöhnt (kostenpflichtig) am langen Ende ist es aber reine Geschmacksache. Öffnet man die beigelegte sample.plist erkennt man folgende Struktur die nun von oben nach unten an unsere Gegebenheiten angepasst wird:

Ich werde versuchen zu jedem Punkt den ich in der Config anfasse eine kleine Erklärung zu liefern beschränke mich hierbei aber wirklich auf die nötigsten Punkte denn andernfalls könnte ich auch gleich die Dokumentation von OpenCore posten und das will ja niemand 😉

Vom Grundsatz her finden wir unter vielen Punkten in der Config eine ähnliche Struktur. immer wieder begegnen uns "Add", "Delete", "Patch" und Quirks wobei die Logik immer ist "füge hinzu was noch nicht vorhanden ist" (Add), lösche was definiert wurde (Delete), Verändere was definiert wurde (Patch) und bügle aus was kaputt ist (Quirks). Von der Reihenfolge wird immer

zuerst Delete dann Add dann Patch und dann Quirks ausgeführt (sollte man für alles was man tut im Hinterkopf haben). Eine Eigenheit von OpenCore ist es das anders als bei Clover nichts vom Dateisystem verwendet wird solange es nicht explizit in die config eingetragen wurde was für uns bedeutet das wir im nächsten Schritt also die Dateien die wir zuvor im Dateisystem abgelegt haben quasi in der Config einchecken. Fangen wir oben mit dem Punkt ACPI an wenn wir hier den Knoten Add aufklappen gibt es im Sample schon eine ganze Litanei an eingetragenen Objekten und im Grunde müssen wir uns hier nur die beiden Einträge aussuchen die mit den Dateien korrespondieren die wir in den Ordner /EFI/OC/ACPI gelegt haben und diese aktivieren.

die restlichen Einträge benötigen wir nicht diese können wir entweder löschen oder einfach so lassen wie sie sind (ich lösche sie der Übersicht halber immer) und damit sind wir mit dem Knoten ACPI schon fertig und können weiter zu Booter wandern. Unter "Booter" ist der Punkt Quirks der spannende Punkt denn alles was wir hier einstellen können/müssen beeinflusst direkt wie sich der Booter von MacOS (boot.efi) auf unserem System verhält. Ich habe hier für Dein Setup folgende Einstellungen gewählt:



Property-List	Typ	Wert
Booter	Dictionary	
MmioWhitelist	Array	
Patch	Array	
Quirks	Dictionary	
AllowRelocationBlock	Boolean	NO
AvoidRuntimeDefrag	Boolean	YES
DevirtualiseMmio	Boolean	NO
DisableSingleUser	Boolean	NO
DisableVariableWrite	Boolean	NO
DiscardHibernateMap	Boolean	NO
EnableSafeModeSlide	Boolean	YES
EnableWriteUnprotector	Boolean	YES
ForceExitBootServices	Boolean	NO
ProtectMemoryRegions	Boolean	NO
ProtectSecureBoot	Boolean	NO
ProtectUefiServices	Boolean	NO
ProvideCustomSlide	Boolean	YES
ProvideMaxSlide	Zahl	0
RebuildAppleMemoryMap	Boolean	NO
SetupVirtualMap	Boolean	YES
SignalAppleOS	Boolean	NO
SyncRuntimePermissions	Boolean	NO

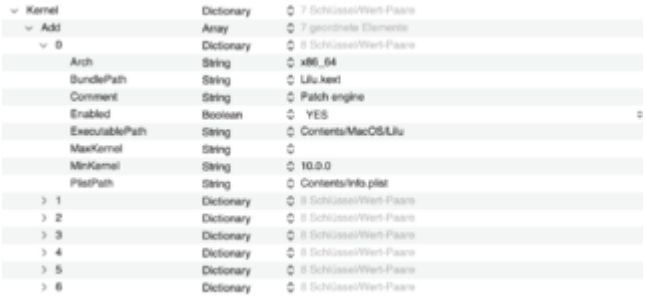
und habe mich dabei mehr oder weniger an die Voreinstellung gehalten was man im übrigen

bei aktueller Hardware in den meisten Fällen auch getrost machen kann (bei älterer oder exotischere Hardware wie zum Beispiel bei X-Serie Boards ist hier ggf. mehr Arbeit nötig). Weiter im Text geht es mit dem Knoten Kernel der Knoten hat in der config folgende Struktur:

Übersetzt auf Clover entspricht:

- Add -> Den Kext Verzeichnissen unter /Clover/Kexts/ wobei unter Clover die Extensions nicht explizit in der Config eingetragen werden
- Block -> Hier hat Clover keine Entsprechung
- Emulate -> Entspricht dem Bereich CPU in Clover (CloverConfigurator)
- Force -> Entspricht "Force Kext2Load"
- Patch -> Entspricht "Kext2Patch" und "Kernel2Patch"
- Quirks -> Entspricht Quirks in neueren Clover Versionen und ist bei alten Versionen nicht vorhanden
- Scheme -> ist in Clover nicht vorhanden

In OpenCore müssen wir also unsere Extensions in die config eintragen damit OpenCore diese verwendet hierbei gilt es zu beachten das die richtige Reihenfolge eine Rolle spielt denn alle Extensions werden in exakt der Reihenfolge geladen in der sie in die config eingetragen wurden wenn also Extensions Abhängigkeiten zueinander haben müssen wir das an der Stelle berücksichtigen. Ausgehend von dem was wir zuvor in das Verzeichnis Kexts kopiert haben tragen wir also ein:



An der Stelle ein paar Worte zu den einzelnen Punkten da ich denke das es dem Verständnis zuträglich ist...

- Arch -> Entspricht der zu verwendenden Architektur mögliche Werte sind hier X86_64 (64Bit), I386 (32Bit) oder Any
- BundlePath -> Der volle Dateiname der einzutragenden Extension
- Comment -> Ein frei wählbarer Kommentar der Beschreibt um was es sich handelt
- ExecutablePath -> Der Pfad zum ausführbaren Teil der Extension (eigentlich immer /Contents/MacOS/name des Kext ohne die Endung .kext
- MaxKernel und MinKernel -> Die MacOS Version bis zu der die Extension maximal verwendet werden soll oder die mindestens vorliegen muss damit sie verwendet wird (entspricht den Versionsordnern in Clovers Kexts Verzeichnis)
- PlistPath -> Pfad zur Info.plist eigentlich immer /Contents/Info.plit

In Deinem Fall ist die Liste der Extensions glücklicherweise übersichtlich und wir daher schnell mit dem eintragen fertig Wichtig ist das Lilu.kext an erster Stelle in der Liste steht weil alles was danach kommt von diesem Kext abhängig ist (WhateverGreen, AppleALC, VirtualSMC...) haben wir hier alles eingetragen geht es weiter mit den Punkten MISC, NVRAM und PlattformInfo. Ich spare jetzt die Punkte MISC und NVRAM für den Moment aus und reiche sie in einem weiteren Beitrag nach und gehe nur noch schnell auf den Punkt PlattformInfo ein welcher dem Punkt SMBIOS in Clover entspricht.

Unter OpenCore ist es relativ einfach die Systemdefinition einzutragen und zu erzeugen denn alles was wir dazu brauchen sind im Kern 4 Werte aus unserer alten Clover Config den Rest können wir getrost OpenCore überlassen denn es wird uns ein schöner Automatismus mitgeliefert der aus unseren eingegeben vier Werten ein komplettes und valides SMBOIS erzeugt. In Deinem Fall ist der Rechner aktuell als iMac18,3 deklariert und das übernehmen wir auch erstmal so (später wäre es sinnvoll auf iMacPro 1,1 zu gehen bei Deinem Setup) alles was wir brauchen ist ausser dem Modell noch die Seriennummer, die SystemSerialNumber (MLB) die SystemUUID (SMUUID) und den ROM Wert um den Rest kümmert sich OpenCore. Also fix eingetragen und Haken an den Punkt PlattformInfo. Soweit erstmal für den Moment die resultierende EFI findest Du hier: EFI.zip die kannst Du Dir herunterladen und auf einen wie folgt vorbereiteten USB Stick packen:

Diesen Stick steckst Du dann an den Rechner und wählst ihn im Bios Bootmenu aus. Wenn alles geklappt hat sollte Dich der Bootpicker von OpenCore begrüßen.

EDIT: Ich habe die EFI nochmal ausgetauscht weil mir noch zwei kleine Fehler aufgefallen sind sorry dafür ☹