Konfiguration

Achtung!

Änderungen an den Konfigurationsdateien von GRUB 2 können dazu führen, dass das System bei unsachgemäßen Einstellungen nicht mehr gestartet wird. Man sollte daher vor Änderungen an der Konfiguration stets eine Sicherung der Dateien **/etc/default/grub** sowie der Verzeichnisse **/etc/grub.d** und **/etc/default/grub.d** anlegen. Außerdem benötigt man für den Notfall ein aktuelles **Rettungssystem** auf einem externen Medium, um GRUB 2 in einem solchen Fall wieder **reparieren** zu können.

Dieser Artikel wurde für die folgenden Ubuntu-Versionen getestet:

- **Ubuntu 22.04** Jammy Jellyfish
- **Ubuntu 20.04** Focal Fossa

Du möchtest den Artikel für eine weitere Ubuntu-Version testen? Mitarbeit im Wiki ist immer willkommen! Dazu sind die Hinweise **zum Testen von Artikeln** zu beachten.

Zum Verständnis dieses Artikels sind folgende Seiten hilfreich:

1. **Ein Terminal öffnen**
2. **Einen Editor öffnen**
3. **Mit Root-Rechten arbeiten**

Inhaltsverzeichnis

16. **GRUB_GFXPAYLOAD_LINUX**
17. **GRUB_BACKGROUND**
18. **GRUB_THEME**
19. **GRUB_DISABLE_OS_PROBER**
20. **GRUB_OS_PROBER_SKIP_LIST**
21. **GRUB_DISABLE_SUBMENU**
22. **GRUB_ENABLE_CRYPTODISK**
23. **GRUB_INIT_TUNE**
24. **GRUB_RECORDFAIL_TIMEOUT**
25. **GRUB_RECOVERY_TITLE**
26. **GRUB_FORCE_PARTUUID**

3. **Links**
    1. **Intern**
    2. **Extern**

Die Entwickler haben **GRUB 2** [https://wiki.ubuntuusers.de/GRUB_2/] so ausgelegt, dass er in Standard-Umgebungen – wie z.B. Rechnern mit genau einem Datenträger – automatisch zu einem stets startfähigem System führt und dabei weitestgehend unbemerkt vom Anwender im Hintergrund seine Arbeit verrichtet. So ist bei einem solchen System das GRUB-Menü standardmäßig versteckt, so dass es der Endanwender gar nicht zu Gesicht bekommt.

Die Erstellung der GRUB-2-Konfigurationsdatei, aus der GRUB 2 die Einstellungen für die Darstellung des GRUB-Auswahl-Menüs während des Systemstarts bezieht, ist standardmäßig vollständig automatisiert. Das ist für den Endanwender komfortabel, weil er sich mit der Konfiguration zunächst nicht auseinandersetzen muss. Auf der anderen Seite ist ein Eingriff in die Konfiguration durch die Automation unter Umständen komplexer, als man es sich zunächst vorstellen mag.

Dieser Artikel erklärt daher grundlegend, wie die Konfiguration bzw. die Erstellung der Konfiguration von GRUB 2 erfolgt und wie der Nutzer in einfacher Weise durch Setzen bzw. Ändern vordefinierter Parameter Einfluss auf die Konfiguration nehmen kann.

Hinweis:

Anwender die sich mit der Arbeit an Text-Dateien schwer tun, können Konfigurationsänderungen zunächst auch mit dem grafischen Konfigurationswerkzeug **GRUB Customizer** versuchen.

Konfigurationsübersicht

Die Konfiguration von GRUB 2 unterteilt sich zum einen in die Konfigurationsdatei aus der GRUB 2 bei dessen Ausführung während des Systemstarts die Parameter für die Gestaltung der **Laufzeitumgebung** [https://de.wikipedia.org/wiki/Laufzeit_(Informatik)] bezieht. In der Regel legt der Endanwender unter Ubuntu Parameter aber in Konfigurationsdateien fest aus denen dann das Programm `grub-mkconfig` die eigentliche Konfigurationsdatei für die Laufzeitumgebung automatisch generiert.

Von der Laufzeitumgebung verwendete Konfiguration - grub.cfg

GRUB 2 bezieht seine Konfiguration zur Laufzeit standardmäßig aus der Datei **/boot/grub/grub.cfg**. Diese Konfigurations-Datei ist in einer eigens dafür entwickelten Skriptsprache geschrieben, deren Syntax man im GRUB 2 Handbuch unter Ubuntu im Terminal[1] wie folgt findet:

```
info -f grub -n 'Shell-like scripting'
```

Alternativ kann man gleichen Inhalt auch online nachlesen: **https://www.gnu.org/software/grub/manual/grub/html_node/Shell_002dlike-scripting.html** [https://www.gnu.org/software/grub/manual/grub/html_node/Shell_002dlike-scripting.html]

Wer als Anwender ohne weitergehende Computer-Kenntnisse bei *"eigens dafür entwickelten Skriptsprache"* bereits Panik bekommt, kennt nun bereits einen Grund, warum die GRUB-Macher das Erstellen dieser Konfigurationsdatei standardmäßig automatisiert haben: Man erspart dem Benutzer damit, sich mit dieser Skriptsprache auskennen zu müssen.

Ein weiterer Grund ergibt sich aus dem Umstand, dass das GRUB-Menü hinsichtlich von Linux-Betriebssystemen Kernel-Einträge zur Auswahl anbietet. Die Kernel werden aber unter Linux regelmäßig aktualisiert, weshalb ein Kernel-Update regelmäßig ein Update der GRUB-Laufzeit-Konfiguration erfordert. Es wäre für den Endanwender lästig, dies stets manuell in der **/boot/grub/grub.cfg** anpassen zu müssen.

Außerdem könnte der Anwender das Aktualisieren der Datei vergessen und dann unter Umständen einen veralteten Kernel verwenden.

Automatische Generierung der Laufzeit-Konfiguration

Die Laufzeit-Konfiguration in **/boot/grub/grub.cfg** wird jedes mal erstellt bzw. aktualisiert, wenn durch den Anwender oder automatisch durch das System der Befehl `update-grub` mit **Rootrechten** [https://wiki.ubuntuusers.de/Rootrechte/] ausgeführt wird. `update-grub` ist dabei schlicht eine Kurzform des Befehls `grub-mkconfig -o /boot/grub.grub.cfg`. Dieser Befehl veranlasst GRUB, eine Konfiguration zu generieren und das Ergebnis mittels der Option `-o` in die Datei **/boot/grub/grub.cfg** auszugeben.

Ablauf des Generierungsprozesses

Sobald `grub-mkconfig` ausgeführt wird, liest es zunächst die Konfigurations-Parameter ein, die für den dann folgenden Generierungsprozesses gelten sollen. Diese Konfigurations-Parameter werden wie folgt bezogen:

1. Zum einen werden einige Parameter direkt innerhalb von `grub-mkconfig` - bei welchem es sich um ein einfaches Shell-Script handelt - gesetzt.
2. Daneben werden Standard-Parameter aus der Datei **/etc/default/grub** eingelesen.
3. Schließlich werden etwaig vorhandene Dateien aus **/etc/default/grub.d/*.cfg** in alphanumerischer Reihenfolge ausgewertet.

Das Einlesen erfolgt dabei in der angegebenen Reihenfolge. Sind die selben Konfigurations-Parameter dabei an verschiedenen Stellen gesetzt, so setzt sich der zuletzt eingelesene Parameter durch.

Nachdem alle Parameter eingelesen sind startet der Generierungsprozess unter deren Berücksichtigung. Dabei werden die Shell-Skripte aus dem Verzeichnis **/etc/grub.d** in alphanumerischer Reihenfolge abgearbeitet. Diese Skripte erzeugen dabei Schritt-für-Schritt den Inhalt der **/boot/grub/grub.cfg**.

Möglichkeiten der Einflussnahme auf den Generierungsprozess

Aus **obiger** Ablaufbeschreibung ergeben sich für den Anwender folgende Möglichkeiten der Einflussnahme auf den Generierungsprozess:

1. Änderung vordefinierter Konfigurations-Parameter (Schwierigkeitsgrad: einfach, nur Kenntnisse der bestehenden Konfigurationsparameter notwendig - Flexibilität: gering)
2. Ändern der Ablaufreihenfolge der Skripte innerhalb von **/etc/grub.d** (Schwierigkeitsgrad: einfach keine besonderen Kenntnisse notwendig - Flexibilität: gering, ermöglicht ausschließlich die Beeinflussung der Reihenfolge der Menü-Einträge)
3. Bestehende Skripte anpassen bzw. durch eigene ersetzen bzw. zu den bestehenden Skripten zusätzlich eigene hinzufügen (Schwierigkeitsgrad: Grundlegende **Shell-Programmier-Kenntnisse** [https://wiki.ubuntuusers.de/Shell/Bash-Skripting-Guide_f%C3%BCr_Anf%C3%A4nger/] notwendig - Flexibilität: hoch)

Dieser Artikel beschränkt sich im folgenden auf die Änderung **vordefinierter Konfigurations-Parameter**. Zur Änderung der Reihenfolge und des Inhaltes der bestehenden Skripte sowie zum Anlegen eigener Skripte siehe **GRUB 2/Skripte** [https://wiki.ubuntuusers.de/GRUB_2/Skripte/].

Manuelles Erstellen der grub.cfg

Neben der automatischen Generierung der Konfiguration durch die vorgegebenen GRUB-2-Skripte, spricht grundsätzlich überhaupt nichts dagegen, die Laufzeit-Konfigurationsdatei **grub.cfg** manuell zu erstellen. Schließlich bietet diese Möglichkeit die größtmögliche Flexibilität der Konfiguration. Man muss dabei neben den bereits [oben] erwähnten Nachteilen aber berücksichtigen, dass man sich dann auch um das Außerkraftsetzung der von den GRUB-Entwicklern und Ubuntu-Paketbetreuern eingerichteten Automatismen kümmern muss, damit die manuell erstellte Konfiguration nicht überschrieben wird.

Da man dabei die Vorgaben der Ubuntu-Paketbetreuer verlässt, wird diese Vorgehen nur erfahrenen Anwendern empfohlen und ist auch nicht Gegenstand dieses Artikels!

Es sei diesbezüglich nochmals auf das **GRUB-Handbuch** [https://www.gnu.org/software/grub/manual/grub/html_node/Shell_002dlike-scripting.html] 🇬🇧 verwiesen.

Vordefinierte GRUB-2-Konfigurationsparameter

Die grundlegenden, allgemeinen Einstellungen von GRUB 2 befinden sich standardmäßig in der Datei **/etc/default/grub**, bei der es sich um eine einfache Textdatei handelt, die dem POSIX-Standard für die Eingabe folgt.

Die Konfigurationsparameter (Variablen) werden dort nach folgendem, einfachen Schema definiert:

```
Variable=Wert
```

- `Wert` muss in Anführungszeichen eingefasst werden, sobald er Leerzeichen enthält.
- Eine `Variable` kann deaktiviert und trotzdem in der Datei belassen werden indem man der Zeile mit dem betreffenden Parameter eine `#` voranstellt. So lassen sich auch Kommentare in die Datei integrieren.
- Weiterhin ist zu beachten, dass manche für das Durchreichen durch GRUB bestimmte Zeichen wie z.B. `$` gleich doppelt durch Backslash (`\\`) maskiert werden müssen.

Standardvariablen

Die Textdatei **/etc/default/grub** sieht direkt auf neueren Installationen in aller Regel wie folgt aus:

```
1  # If you change this file, run 'update-grub' afterwards to update
2  # /boot/grub/grub.cfg.
3  # For full documentation of the options in this file, see:
4  #   info -f grub -n 'Simple configuration'
5
6  GRUB_DEFAULT=0
7  GRUB_TIMEOUT_STYLE=hidden
8  GRUB_TIMEOUT=0
9  GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
10 GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
11 GRUB_CMDLINE_LINUX=""
12
```

Bearbeitung der Variablen

Grundsätzlich kann die Datei **/etc/default/grub** mit einem **Editor** [https://wiki.ubuntuusers.de/Editor/] mit **Rootrechten bearbeitet** [https://wiki.ubuntuusers.de/mit_Root-Rechten_arbeiten/#sudoedit] werden. Unter Ubuntu-basierten Distributionen liest `grub-mkconfig` aber auch Parameter-Konfigurationen aus dem Verzeichnis **/etc/default/grub.d** ein.

Deshalb sollte man Änderungen an bereits definierten oder auch noch nicht gesetzten Variablen in einer eigenen Datei im Verzeichnis **/etc/default/grub.d** ablegen.

Dies hat folgende Vorteile:

- Man sieht mit einem Blick in die entsprechende Datei sofort die eigenen Anpassungen und muss diese nicht extra dokumentieren.

- Eine eigene Datei im Verzeichnis **/etc/default/grub.d** wird bei Neuinstallation von GRUB 2 oder bei einem Upgrade seiner Pakete nicht überschrieben.
- Man kann Konfigurations-Parameter in jeweils einer eigenen Datei gruppieren und so dann leicht aktivieren oder deaktivieren.

Für die Erstellung eigener Konfigurations-Dateien im Verzeichnis **/etc/default/grub.d** gelten die folgenden Regeln:

- Damit die Datei von `grub-mkconfig` eingelesen wird, muss der Dateiname auf **.cfg** enden und die Datei muss ausführbar sein. Folglich lässt sich eine Konfiguration durch Setzen bzw. Entfernen der Erweiterung aktivieren bzw. deaktivieren oder auch durch Entfernen/Setzen der Ausführungsrechte.
- Die Dateien im Verzeichnis werden in alphanumerischer Reihenfolge eingelesen. Parameter-Dateien die mit **11** beginnen werden vor Dateien eingelesen die mit **zz** beginnen. D.h. bei gleichen Parametern in beiden Dateien überschreiben die aus der Datei **zz** die aus der Datei **11**.

Beispiel: Möchte man den Wert von **GRUB_TIMEOUT_STYLE** auf den Wert `menu` ändern, um das GRUB-Menü bei jedem Start anzuzeigen, so geht dies wie folgt:

- Anlegen einer Datei **/etc/default/grub.d/zz_myconf.cfg** im Terminal:
  ```
  sudoedit /etc/default/grub.d/zz_myconf.cfg
  ```
- Setzen ausschließlich der Variablen, die man bearbeiten möchte:
  ```
  GRUB_TIMEOUT_STYLE=hidden
  ```
- Speichern der Datei **/etc/default/grub.d/zz_myconf.cfg**, beim Editor **nano** [https://wiki.ubuntuusers.de/nano/] geschieht dies über die Tasten Strg + O , bestätigen mit ⏎ und verlassen mittels Strg + X .

Bei der nächsten **Aktualisierung** von GRUB 2 wird die geänderte Einstellung dann übernommen.

Kontrolle der Bearbeitung

Bevor man die Änderungen abschließend in die Datei **/boot/grub/grub.cfg** überträgt, kann und sollte man sich von Auswirkung der Änderungen ein Bild machen. Dazu führt man einfach `grub-mkconfig` ohne Dateiausgabe-Option aus:

```
sudo grub-mkconfig
```

Da die Ausgabe recht lang und damit unübersichtlich ist und für jedes Skript das abgearbeitet wird eine eigene Sektion in der Ausgabe vorhanden ist, kann man sich auch nur Teilausschnitte anzeigen lassen. Z.B. nur den Kopf der GRUB-Konfiguration (00_header):

```
sudo grub-mkconfig | grep -Poz "(?<=### BEGIN /etc/grub.d/00_header ###)(?s).*(?=### END
/etc/grub.d/00_header ###)"
```

GRUB-Konfiguration aktualisieren

Nach jeder Änderung an den Konfigurations-Dateien **/etc/default/grub.d/*** bzw. oder an Skripten im Verzeichnis **/etc/grub.d**, muss die GRUB-Konfiguration - die in der Datei **/boot/grub/grub.cfg** gespeichert ist - aktualisiert werden. Andernfalls werden Änderungen beim nächsten Systemstart nicht sichtbar. Dies erledigt man mittels:

```
sudo update-grub
```

Bedeutung der Variablen

Mit der Eingabe in ein Terminal von

```
info -f grub -n 'Simple configuration'
```

erhält man eine Übersicht 🇬🇧 aller Variablen, die nachfolgend in Teilen und frei übersetzt wiedergegeben wird. Teilweise wurden die folgenden Informationen um Erfahrungen aus der Praxis ergänzt und gehen damit über die Darstellung im obigen Handbuch hinaus. Einige Variablen, die im Handbuch erklärt werden, können im folgenden fehlen:

GRUB_DEFAULT

Gibt an, welcher Eintrag im Menü standardmäßig hervorgehoben wird. Dieser Eintrag wird geladen, falls keine andere Auswahl getroffen wird.

- **Mögliche Werte**: Zahl, Menü-Eintrag-ID, Menü-Eintragbeschreibung oder der Spezialwert `saved`
  - `Zahl`: Die Zählung beginnt mit 0. Der dritte Eintrag im Menü würde also durch eine **2** hervorgehoben. Diese Vorgabe ist statisch.
  - `Menü-Eintrag-ID`: Wählt den Eintrag mit der exakten Eintrags-ID.
    - Beispiel: Lautet der betreffenden Menü-Eintrag in der **grub.cfg**
      ```
      menuentry 'Ubuntu' --class ubuntu --class gnu-linux --class gnu --class os
      $menuentry_id_option 'gnulinux-simple-2881e6a7-4b74-4dc0-98e3-02ef130a5b5c'
      ```
      , dann definiert man

      `GRUB_DEFAULT="gnulinux-simple-2881e6a7-4b74-4dc0-98e3-02ef130a5b5c"`
  - `Menü-Eintragbeschreibung`: Legt den Standardeintrag anhand der Menü-Beschreibung aus. Diese Option sollte man nach Möglichkeit nicht mehr verwenden, da sich die Beschreibungen unter Umständen ändern können.
    - Beispiel: Lautet der betreffenden Menü-Eintrag in der **grub.cfg**
      ```
      menuentry 'Ubuntu, mit Linux 5.15.0-58-generic' --class ubuntu --class gnu-linux --class
      gnu --class os $menuentry_id_option 'gnulinux-5.15.0-58-generic-advanced-2881e6a7-4b74-
      4dc0-98e3-02ef130a5b5c'
      ```
      , dann definiert man

      `GRUB_DEFAULT="Ubuntu, mit Linux 5.15.0-58-generic"`
  - `saved`: Trägt man den Wert `saved` ein, so wird GRUB veranlasst, den in **/boot/grub/grubenv** gespeicherten Menüeintrag auszulesen und als Vorgabe zu verwenden, unabhängig davon, ob die Reihenfolge inzwischen durch ein Kernel-Update oder durch eigene Skripte verändert wurde. Siehe auch **GRUB_SAVEDEFAULT**.
- **Standard**: `GRUB_DEFAULT=0`

GRUB_SAVEDEFAULT

Ermöglicht dem Anwender zur Laufzeit von GRUB durch Auswahl eines Eintrages im GRUB-Menü mittels ↑ / ↓ und ⏎ den betreffenden Eintrag für künftige GRUB-Läufe als Standardeintrag festzulegen. Dies erfordert jedoch, dass zugleich die Variable **GRUB_DEFAULT** auf den Wert `saved` gesetzt ist.

- **Mögliche Werte**: boolescher Ausdruck `true`, `false`
  - `true`: Das Auswählen eines Menüeintrages im GRUB-Menü durch den Benutzer legt diesen für künftige GRUB-Starts als Standardeintrag fest.
  - `false` oder nicht gesetzt: Das Auswählen eines Menü-Eintrages durch den Anwender verändert den von GRUB verwendeten Standard nicht.
- **Standard**: `GRUB_SAVEDEAULT=`

Hinweis:

Diese Funktion setzt auf das Vorhandensein des GRUB-Environment-Blockes, der jedoch nicht in jeder Systemkonfiguration zur Verfügung steht.

---

grubenv anpassen

Hatte man Probleme beim Starten mit einem Menüeintrag oder kommt man wegen *falscher* Einstellungen in der Datei **/etc/default/grub** bei den Werten `GRUB_DEFAULT` oder anderer Fehler nicht ins GRUB-Menü, so kann man die Datei **/boot/grub/grubenv** auch vom Desktop her in einem Terminal mit `grub-set-default` neu konfigurieren.

Will man dagegen einem bestimmten Eintrag beim nächsten Start des Rechners verwenden, so kann man die Datei **/boot/grub/grubenv** vom Desktop her in einem Terminal [1] entsprechend mit `grub-reboot` einrichten, so dass dieser Eintrag einmalig aufgerufen wird. Der nächste Start erfolgt dann wieder mit dem vorher gespeicherten Eintrag.

> **Achtung!**
>
> Das Abspeichern funktioniert aus Sicherheitsgründen nicht auf einem RAID-Verbund, einem **LVM**-Dateisystem, einem **Btrfs**-Dateisystem oder einem **ZFS**-Dateisystem (siehe **GNU GRUB MANUAL**).

GRUB_TIMEOUT

Mit dieser Variablen lässt sich eine Zeit in Sekunden festlegen, die verstreichen soll bevor GRUB 2 ohne Benutzerinteraktion mit dem Laden des Standard-Eintrages (siehe **GRUB_DEFAULT** fortfährt. Dem Benutzer bleibt damit die angegebene Zeit, um auf GRUB 2 Einfluss nehmen zu können.

- **Mögliche Werte**: Zahl
  - `-1`: Kein Timeout, d.h. gebootet wird erst nach Eingabe durch den Benutzer.
  - `0`: Es wird ohne Verzögerung direkt der unter **GRUB_DEFAULT** festgelegte Eintrag gestartet, ohne dass das Auswahlmenü angezeigt wird.
  - `postive Zahl X`: Wartet ohne Tastendruck durch den Anwender `X` Sekunden bevor der unter **GRUB_DEFAULT** gesetzte Eintrag gestartet wird.
    - Beispiel: `GRUB_TIMEOUT=15` (wartet für 15 Sekunden auf eine Aktion des Benutzers, startet andernfalls nach Ablauf der 15 Sekunden den Standardeintrag.)
- **Standard**: `GRUB_TIMEOUT=5`

> **Hinweis:**
>
> Greift der Anwender durch Druck einer Taste ein, so wird der Timeout unterbrochen. GRUB 2 fährt dann mit dem Systemstart erst nach entsprechender Aufforderung durch den Benutzer mittels entsprechender Taste fort.

GRUB_TIMEOUT_STYLE

Legt das Erscheinungsbild des Timeouts fest.

- **Mögliche Werte**: `countdown`, `hidden`, `menu`
  - `countdown`: Es wird kein GRUB-Menü angezeigt und die mit `GRUB_TIMEOUT` eingestellte Zeit sichtbar heruntergezählt. Danach startet das System mit dem mit `GRUB_DEFAULT` voreingestellten Eintrag. Kurzzeitiges Drücken von Esc innerhalb dieses Zeitfensters bricht das herunter Zählen ab und es kann eine Auswahl vorgenommen werden.
  - `hidden`: Es wird kein GRUB-Menü angezeigt und die mit `GRUB_TIMEOUT` eingestellte Zeit gewartet, bis das System nach der voreingestellten Zeit startet. Durch kurzzeitiges Drücken von Esc innerhalb dieses Zeitfensters kann das GRUB-Menü angezeigt und eine Auswahl vorgenommen werden.
  - `menu` oder nicht gesetzt: Das GRUB-Menü wird immer angezeigt, auch wenn kein weiteres Betriebssystem (Windows / Linux) über GRUB verwaltet wird (z.B. wegen `GRUB_DISABLE_OS_PROBER=true`). Das System wird mit der Auswahl (↓ / ↑ + ⏎ oder nach Ablauf der mit `GRUB_TIMEOUT` eingestelltem Zeit gestartet.
- **Standard**: `GRUB_TIMEOUT_STYLE=hidden` (beachte aber den nachfolgenden Hinweis!)

> **Hinweis:**
>
> Die Einstellung `hidden` bzw. `countdown` wird ignoriert und das GRUB-Menü immer angezeigt, wenn mehr als ein Betriebssystem von GRUB verwaltet wird. Bei `GRUB_TIMEOUT=0` (Vorgabe Null) wird das GRUB-Menü dann 10 Sek. lang angezeigt. Alle anderen Werte (also `-1` oder >= `1`) werden der Vorgabe entsprechend verarbeitet.

GRUB_DISTRIBUTOR

Ermöglicht es, die Einträge der Distribution individuell bzw. aussagekräftiger zu gestalten. Bei einer Installation im UEFI-Modus bestimmt diese Variable grundsätzlich außerdem die Benennung des Eintrages im NVRAM sowie die Benennung des Ordners auf der EFI-Systempartition.

- **Mögliche Werte**: Beliebige Zeichenkette, die jedoch keine Umlaute, Leer- und Sonderzeichen enthalten darf.
  - Beispiele:
    - `GRUB_DISTRIBUTOR=LubuntuJammy`
    - `GRUB_DISTRIBUTOR=Ubuntu_Jammy_22-04`
- **Standard**: GRUB_DISTRIBUTOR=`` `lsb_release -i -s 2> /dev/null || echo Debian` `` (dieser Befehl führt standardmäßig bei allen Ubuntu-Derivaten zu dem Wert `Ubuntu`)

Sobald man für `GRUB_DISTRIBUTOR` einen eigenen Wert definiert, sind bei Verwendung eines Systems im UEFI-Modus zur Erzielung des gewünschten Ergebnisses weitere Maßnahmen notwendig. Dabei ist noch mal zu unterscheiden, ob im UEFI-Modus Secure-Boot aktiv ist oder nicht:

Besonderheiten bei UEFI-Boot-Modus

Bei einem System, dass im UEFI-Boot-Modus ohne Secure-Boot betrieben wird, müssen zur ordnungsgemäßen Funktion einer eigenen Einstellung folgende zusätzlichen Maßnahmen ergriffen werden:

- Das Paket **grub-efi-amd64-signed** und falls vorhanden **shim-signed** müssen deinstalliert werden:

  > **Achtung!**
  >
  > Die hier verwendete **apt**-Option `--allow-remove-essential` erlaubt für das System als essentiell wichtig gekennzeichnete Pakete zu entfernen und sollte daher nur mit Bedacht verwendet werden! Das Paket **shim-signed** ist auf einem System mit aktiviertem Secure-Boot tatsächlich essentiell und sollte dort auf gar keinen Fall entfernt werden! Das Entfernen sollte daher nur auf Systemen erfolgen, die die UEFI-Boot-Methode ohne Secure-Boot dauerhaft verwenden.

  ```
  sudo apt purge grub-efi-amd64-signed shim-signed --allow-remove-essential
  ```

- Die etwaige Neuinstallation der Pakete **grub-efi-amd64-signed** und **shim-signed** muss unterbunden werden:

  ```
  sudo apt-mark hold grub-efi-amd64-signed shim-signed
  ```

- GRUB 2 neu auf die EFI-Systempartition installieren:

  ```
  sudo grub-install
  ```

- Abschließend sollte man noch kontrollieren, dass der Ubuntu Start-Eintrag im NVRAM auf die Datei **grubx64.efi** anstatt **shimx64.efi** endet:

  ```
  efibootmgr -v
  ```

Einschränkungen bei Verwendung von Secure-Boot

Bei der Verwendung von Secure-Boot unter Ubuntu verwendet GRUB 2 standardmäßig konzeptbedingt stets die von Canonical signierte und über das Paket **grub-efi-amd64-signed** bereitgestellte EFI-App **grubx64.efi**. Zur ordnungsgemäßen Umsetzung der Einstellung `GRUB_DISTRIBUTOR` ist aber die Verwendung einer angepassten **grubx64.efi** notwendig, weil der in der EFI-App enthaltene Pfad zum Verzeichnis auf der EFI-Systempartition angepasst werden muss.

Eine solche angepasste **grubx64.efi** wird zwar im Rahmen von `grub-install` auch erstellt, allerdings kann sie konzeptbedingt nicht von Canonical signiert werden, was auf einem Standard-Ubuntu-Secure-Boot-System notwendig wäre.

Daher verwendet `grub-install` am Ende der Prozedur schlussendlich auf einem Standard-Secure-Boot-System stets die von Canonical signierte **grubx64.efi** die stets fest und unveränderlich auf das Verzeichnis **/EFI/ubuntu** auf der EFI-Systempartition verweist. Gleichzeitig wird aber auch der vom Endanwender durch `GRUB_DISTRIBUTOR` vorgegebene Ordner auf der EFI-Systempartition eingerichtet, was ziemlich verwirrend sein kann.

Möchte man auf einem Ubuntu-Secure-Boot-System trotzdem vollumfänglich von den Funktionen von `GRUB_DISTRIBUTOR` Gebrauch machen, so bleibt einem nur, die angepasste aber nicht verwendete EFI-App selbst zu signieren und auch selbst auf die EFI-Systempartition zu kopieren. Dazu sind verschiedene Schritte abzuarbeiten, die leider nicht ganz trivial sind und daher nur von erfahrenen Anwendern umgesetzt werden sollten und im folgenden auch nur konzeptionell aufgelistet werden:

- Eine eigene Secure-Boot Zertifikats-Infrastruktur aufsetzen und die EFI-App selbst signieren. Siehe dazu grundsätzlich: **https://ubuntu.com/blog/how-to-sign-things-for-secure-boot** [https://ubuntu.com/blog/how-to-sign-things-for-secure-boot] 🇬🇧
- Hinzufügen des eigenen Zertifikats zum Shim-Zertifikatsspeicher mittels `mokmanager`
- Setzen eines Triggers, der dazu führt, dass nach jedem Ausführen von `grub-install` ein eigenes Script gestartet wird, dass die Standard-Ubuntu-EFI-App durch die angepasste ersetzt und die angepasste EFI-App mit dem eigenen Zertifikat unterzeichnet.

GRUB_CMDLINE_LINUX

Über diese Variable lassen sich **Bootoptionen** [https://wiki.ubuntuusers.de/Bootoptionen/] für die Kernel-Kommandozeile der jeweiligen Menüeinträge für Linux-Systeme übergeben. Die hier festgelegten Optionen werden sowohl an die Kernel-Kommandozeile der Standard- als auch der Recovery-Einträge angehängt.

- **Mögliche Werte**: **Bootoptionen** [https://wiki.ubuntuusers.de/Bootoptionen/] durch Leerzeichen voneinander getrennt. Wenn die gesamte Zeichenkette Leerzeichen enthält, muss sie in in Anführungszeichen eingefasst werden.
  - Beispiel: `GRUB_CMDLINE_LINUX="nomodeset"`
- **Standard**: `GRUB_CMDLINE_LINUX=""`

GRUB_CMDLINE_LINUX_DEFAULT

Diese Variable erlaubt es, **Bootoptionen** [https://wiki.ubuntuusers.de/Bootoptionen/] ausschließlich den Kernel-Kommandozeile der Standard-Menüeinträge anzuhängen.

- **Mögliche Werte**: **Bootoptionen** [https://wiki.ubuntuusers.de/Bootoptionen/] durch Leerzeichen voneinander getrennt. Wenn die gesamte Zeichenkette Leerzeichen enthält, muss sie in in Anführungszeichen eingefasst werden.
- **Standard**: `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"`

Hinweis:

Die Nutzung dieser Variable macht nur Sinn, solange der Recovery-Modus aktiv ist – was bei Ubuntu standardmäßig der Fall ist. Siehe auch **GRUB_DISABLE_RECOVERY**.

GRUB_TERMINAL_INPUT

Diese Variable erlaubt das Festlegen des Eingabegerätes für das GRUB 2 Terminal.

- **Mögliche Werte**: Gültiges `Eingabegerät`. Es können mehrere Geräte durch Leerzeichen getrennt eingegeben werden.
  - `Eingabegerät`: Die folgenden Eingabegeräte können angegeben werden:
    - `console`: Das an die nativ auf der Rechnerplattform zur Verfügung stehende Konsole angeschlossene Eingabegerät.
    - `serial`: Die Eingabe soll über ein an einer seriellen Schnittstelle angeschlossenes Gerät erfolgen.
    - `serial_[Port]`: Wie `serial`, nur das die serielle Schnittstelle `[Port]` explizit mit angegeben wird.
    - `at_keyboard`: Ein angeschlossenes AT-Keyboard wird ausdrücklich verwendet.
    - `usb_keyboard`: Ein angeschlossenes USB-Keyboard wird ausdrücklich verwendet.
- **Standard**: Das auf der Rechnerplattform nativ angeschlossene Eingabegerät.

Hinweis:

Die auswählbaren möglichen Werte hängen von der jeweiligen Plattform ab.

Diese Einstellung ist nur in wenigen Ausnahmesituationen zu ändern und sollte daher in aller Regel unberührt bleiben!

GRUB_TERMINAL_OUTPUT

Diese Variable erlaubt das Festlegen des Ausgabegerätes für das GRUB 2 Terminal.

- **Mögliche Werte**: Gültiges `Ausgabegerät`. Es können mehrere Geräte durch Leerzeichen getrennt eingegeben werden.
  - `Ausgabegerät`: Die folgenden Ausgabegeräte können angegeben werden:
    - `console`: Das an die nativ auf der Rechnerplattform zur Verfügung stehende Konsole angeschlossene Ausgabegerät.
    - `serial`: Die Ausgabe erfolgt über ein an einer seriellen Schnittstelle angeschossenen Gerät.
    - `serial_[port]`: Die Ausgabe erfolgt über ein an der explizit für `[Port]` angegebenen seriellen Schnittstelle angeschlossenen Gerät.
    - `gfxterm`: Grafischer Ausgabemodus
    - `vga_text`: VGA-Textausgabe
    - `mda_text`: MDA-Textausgabe
    - `morse`: Morse-Code-Ausgabe über Systemlautsprecher
    - `spkmodem`: Ausgabe mit einem einfachen Daten-Protokoll auf den Systemlautsprecher
- **Standard**: Das auf der Rechnerplattform nativ zur Verfügung stehende Ausgabegerät. In der Regel bei Ubuntu `gfxterm`.

Hinweis:

Diese Einstellung ist nur in wenigen Ausnahmesituationen zu ändern und sollte daher in aller Regel unberührt bleiben!

GRUB_TERMINAL

Wenn diese Variable gesetzt wir, überschreibt das angegebene Gerät etwaige Angaben innerhalb von **GRUB_TERMINAL_INPUT**, **GRUB_TERMINAL_OUTPUT**. Es wird dann also das angegebene Gerät sowohl für die Ein- als auch die Ausgabe verwendet.

- **Mögliche Werte**: `console` und `serial`
  - `console`: Die nativ auf der Rechnerplattform zur Verfügung stehende Konsole wird als Ein- und Ausgabegerät verwendet.
  - `serial`: Ein an eine serielle Schnittstelle angeschlossenes Terminal wird als Ein- und Ausgabegerät verwendet.
- **Standard**: nicht gesetzt

Achtung!

Wurde ein RAID1-Verbund als Boot-Gerät konfiguriert, so sollte `GRUB_TERMINAL=console` eingestellt werden. Andernfalls kann nach einem Ausfall der ersten Festplatte nicht mehr gebootet werden. (siehe **594202**)

GRUB_SERIAL_COMMAND

Über diese Variable lässt sich die serielle Schnittstelle konfigurieren, sofern diese über **GRUB_TERMINAL_INPUT**, **GRUB_TERMINAL_OUTPUT** bzw. **GRUB_TERMINAL** als Ein-, Ausgabegerät oder beides definiert wurde.

- **Mögliche Werte**: `serial` optional in Verbindung mit weiteren `Optionen`
  - `Optionen`: Eine oder mehrere der folgenden Optionen zur Konfiguration der seriellen Schnittstelle:
    - `--unit=`: Ganze Zahl im Bereich `0-3`, welche die serielle Schnittstelle – häufig mit der Bezeichnung `COM` – repräsentiert. `0` bezeichnet dabei `COM1`, `1` bezeichnet `COM2` usw.. Standard ist `--unit=0`.
    - `--port=`: Port ist der UART-I/O-Port. Falls angegeben geht dieser Wert dem `--unit`-Wert vor.
    - `--speed=`: Gibt die Geschwindigkeit in kbps an. Standard ist `--speed=9600`.
    - `--word=`: Gibt die Zahl der Datenbits aus dem Bereich `5-8` an. Standard ist `--word=8`.
    - `--parity=`: Gibt die Parität mit den folgenden Werten an: `no`, `odd`, `even`. Standard ist `--parity=no`

        - `--stop=`: Gibt die Zahl der Stopbits aus dem Bereich `1-2` an. Standard ist `--stop=1`.
    - Beispiel: `GRUB_SERIAL_COMMAND="serial --unit=0 --speed=115200"`
- **Standard**: grundsätzlich nicht gesetzt, sobald aber durch den Anwender mittels **GRUB_TERMINAL_INPUT**, **GRUB_TERMINAL_OUTPUT** bzw. **GRUB_TERMINAL** eine serielle Schnittstelle angegeben wurde, wird als Standard `GRUB_SERIAL_COMMAND="serial"` gesetzt, was heißt, dass dann die Standardwerte für die Optionen genutzt werden.

Hinweis:

Setzt man die Variable `GRUB_SERIAL_COMMAND`, dann wird es regelmäßig erwünscht sein, auch die Kerneloptionen entsprechend anzupassen. Dazu setzt man dann entweder **GRUB_CMDLINE_LINUX** oder **GRUB_CMDLINE_LINUX_DEFAULT** entsprechend. Also z.B.:

```
GRUB_CMDLINE_LINUX="serial=ttyS0 console=ttyS0,115200n8"
```

GRUB_DISABLE_LINUX_UUID

Mittels dieser Variablen lässt sich festlegen, ob GRUB das Root-Dateisystem dem startenden Kernel über deren **UUID** [https://wiki.ubuntuusers.de/UUID/] benennen soll oder nicht.

- **Mögliche Werte**: boolescher Ausdruck `true`, `false`
    - `true`: GRUB verwendet nicht die **UUID** [https://wiki.ubuntuusers.de/UUID/] zur Benennung des Root-Dateisystems.
    - `false` oder nicht gesetzt: GRUB verwendet die **UUID** [https://wiki.ubuntuusers.de/UUID/] zur Benennung des Root-Dateisystems.
- **Standard**: nicht gesetzt

Hinweis:

Die Übergabe mittels **UUID** ist die zuverlässigste Art und Weise, das Root-Dateisystem an den Linux-Kernel zu übergeben. Diese Variable sollte daher nur in Ausnahmefällen auf `true` gesetzt werden. Bevor man durch Aktivieren dieser Variable das Root-Dateisystem mittels Device-Nummerierung (z.B. **/dev/sdXX**) an den Linux-Kernel übergibt, sollte man prüfen, ob man zur Übergabe nicht auch die UUID der Partitions nutzen kann (siehe **GRUB_DISABLE_LINUX_PARTUUID**).

GRUB_DISABLE_LINUX_PARTUUID

Mit dieser Variable kann man festlegen, dass GRUB 2 das Root-Dateisystem über die **Partitions-UUID** [https://wiki.ubuntuusers.de/blkid/] an den Kernel übergeben soll.

- **Mögliche Werte**: boolescher Ausdruck `true`, `false`
    - `true`: Die Partitions-UUID wird zur Identifizierung des Root-Dateisystems nicht verwendet.
    - `false`: Die Partitions-UUID wird zur Identifizierung des Root-Dateisystems verwendet.
- **Standard**: `GRUB_DISABLE_LINUX_PARTUUID=true`

Hinweis:

Die Verwendung der Partitions-UUID ist weniger zuverlässig als die Verwendung der **Dateisystem-UUID** (siehe zu letzterer auch **GRUB_DISABLE_LINUX_UUID**). Die Partitions-UUID ist aber wiederum zuverlässiger, als die Verwendung der Linux-Device-Nummerierung (**/dev/sdXX**).

GRUB_DISABLE_RECOVERY

Mit dieser Variablen wird festgelegt, ob bei der Generierung der **grub.cfg** auch Einträge im GRUB-Menü für Recovery-Kernel erzeugt werden.

- **Mögliche Werte**: boolesche Ausdrücke `true`, `false`
  - `true`: Es werden keine Recovery-Kernel-Einträge für das GRUB-Menü erzeugt.
  - `false` oder nicht gesetzt: Es werden Recovery-Kernel-Einträge für das GRUB-Menü erzeugt.
- **Standard**: Die Variable ist nicht gesetzt, d.h. die Recovery-Kernel-Einträge werden dem GRUB-Menü hinzugefügt.

GRUB_GFXMODE

Diese Variable legt die Auflösung des grafischen Terminals für GRUB `gfxterm` und damit die Auflösung des Bildschirms, der das GRUB-Menü zur Laufzeit darstellt, fest. Bei der Standard-Darstellung des GRUB-Menüs kann man mit dieser Einstellung in erster Linie die Schriftgröße beeinflussen.

- **Mögliche Werte**: `BREITExHÖHE[xFARBTIEFE]` oder `auto`
  - `BREITExHÖHE[xFARBTIEFE]`: Es wird die angegebene Auflösung für die Darstellung des GRUB-Menü verwendet. Die Auflösung kann nicht beliebig gewählt werden. Es sind nur solche Auflösungen möglich, die von der Grafikkarte mittels **VESA BIOS Extensions (VBE)** [https://de.wikipedia.org/wiki/VESA_BIOS_Extension] unterstützt werden. `[xFARBTIEFE]` ist optional und muss nicht angegeben werden.
  - `auto`: GRUB versucht die für das System optimale Auflösung automatisch zu ermitteln.
  - Beispiele:
    - `GRUB_GFXMODE=800x600`
    - `GRUB_GFXMODE=640x400x24`
- **Standard**: `GRUB_GFXMODE=auto`

Hinweis:

Wie man die zur GRUB-Laufzeit unterstützten Auflösung ermittelt und weitere Details beschreibt **Auflösung ermitteln**. Außerdem muss darauf geachtet werden, dass die Auflösung die maximal mögliche Auflösung des angeschlossenen Monitors nicht überschreitet.

GRUB_GFXPAYLOAD_LINUX

Mit dieser Variabel lässt sich bestimmen, mit welcher Auflösung das Laden des Linux-Kernels durchgeführt wird.

- **Mögliche Werte**: `BREITExHÖHE[xFARBTIEFE]`, `keep` oder `text`
  - `BREITExHÖHE[xFARBTIEFE]`: Es wird/werden die angegebene(n) Auflösung(en) für die Darstellung während des Kernel-Starts verwendet. Die Auflösung kann nicht beliebig gewählt werden. Es sind nur solche Auflösungen möglich, die von der Grafikkarte mittels **VESA BIOS Extensions (VBE)** [https://de.wikipedia.org/wiki/VESA_BIOS_Extension] unterstützt werden. `[xFARBTIEFE]` ist optional und muss nicht angegeben werden. Es können mehrere Auflösungen durch Komma oder Semikolon getrennt aufgelistet werden. Diese werden dann der Reihe nach bis zu einer funktionierenden Auflösung durchprobiert.
    - Beispiele:
      - `GRUB_GFXMODE=800x600`
      - `GRUB_GFXMODE="1024x768x32;800x600x24;640x400"`
  - `keep`: Diese Einstellung bewirkt, dass für das Laden der Kernel die mit **GRUB_GFXMODE** eingestellte Auflösung weiterverwendet wird.
  - `text`: Es wird die Verwendung des normalen Textmodus für das Laden des Kernels erzwungen. Diese Einstellung sollte man wählen, falls es mit den beiden obigen Vorgaben Probleme beim Start gibt (schwarzer bzw. verzerrter Bildschirm).
- **Standard**: nicht gesetzt

Hinweis:

Wie man die zur GRUB-Laufzeit unterstützten Auflösung ermittelt und weitere Details beschreibt **Auflösung ermitteln**.

Die mit dieser Variablen getroffenen Einstellungen werden bei Ubuntu in der Regel von **Plymouth** überschrieben! Entsprechende Einstellungen werden dort erläutert.

GRUB_BACKGROUND

Mit dieser Variablen kann man ein Hintergrundbild für das GRUB-Menü im grafischen Terminal `gfxterm` festlegen.

- **Mögliche Werte**: Gültiger Dateipfad
  - `Dateipfad`: Pfad zu der Bilddatei. Der Dateipfad muss für GRUB zur Laufzeit erreichbar sein. Zulässige Dateitypen sind **.png**, **.tga**, **.jpg** oder **.jpeg**.
    - Beispiel: `GRUB_BACKGROUND="/boot/grub/images/hintergrund.jpg"`
- **Standard**: nicht gesetzt, es wird also kein Hintergrundbild verwendet.

GRUB_THEME

Mit dieser Variablen lässt sich auf Wunsch ein Design-Thema für das grafische Terminal `gfxterm` von GRUB aktivieren.

- **Mögliche Werte**: Gültiger Dateipfad
  - `Dateipfad`: Pfad zu dem Ordner, der die entsprechende Konfiguration des Design-Themas enthält. Dieser Ordner muss zur Laufzeit von GRUB erreichbar sein.
    - Beispiel: `GRUB_THEME=/boot/grub/themes/mein_grub_theme`
- **Standard**: nicht gesetzt, es wird also kein grafisches Design-Thema verwendet.

Hinweis:

GRUB bringt standardmäßig unter Ubuntu kein Design-Thema mit. Über die **Paketverwaltung** lassen sich die beiden vorgefertigten Themen **breeze** und **starfield** installieren. Weitere vorgefertigte GRUB-Themen findet man z.B. auf **gnome-look.org**.

Natürlich kann man auch ein eigenes Thema erstellen. Alle Details dazu sowie zur Einrichtung eines Themas beschreibt der Artikel **GRUB 2/Aussehen - erweiterte Konfiguration**.

GRUB_DISABLE_OS_PROBER

Diese Variable legt fest, ob GRUB bei der Generierung der **grub.cfg** das Programm `os-prober` ausführt und damit nach vorhandenen Betriebssystemen auf anderen Partitionen sucht oder nicht.

- **Mögliche Werte**: boolescher Ausdruck `true`, `false` oder ab GRUB 2.06 und damit ab Ubuntu 21.10 zusätzlich `auto`
  - `true`: Das Programm `os-prober` wird nicht ausgeführt. Damit werden keine weiteren Betriebssysteme in das GRUB-Menü aufgenommen.
  - `false`: Das Programm `os-prober` wird immer ausgeführt, wenn mittels `grub-mkconfig` bzw. `update-grub` eine neue **grub.cfg** generiert bzw. eine bestehende **grub.cfg** aktualisiert wird. Diese Option erfordert das Paket **os-Prober**.
  - `auto` (erst ab Ubuntu 21.10 bzw. Grub 2.06 verfügbar): Bei dieser Einstellung wird zunächst überprüft, ob in einer bereits vorhanden Datei **/boot/grub/grub.cfg** Menü-Einträge anderer Betriebssysteme vorhanden sind. Nur wenn dies zutrifft, wird bei der Generierung auch das Programm `os-prober` ausgeführt. Erfordert das Paket **os-prober**.
- **Standard**:
  - **< = Ubuntu 21.04**: `GRUB_DISABLE_OS_PROBER=false`, es wird also immer bei jedem Durchlauf von `update-grub` bzw. `grub-mkconfig` nach weiteren Betriebssystemen gesucht.
  - **> = Ubuntu 21.10**:
    - `GRUB_DISABLE_OS_PROBER=auto`

Hinweis:

Die seit Ubuntu 21.10 vorhandene Option `auto` wurde von den Ubuntu-Entwicklern in erster Linie für Upgrade-Szenarien eingeführt und soll verhindern, dass auf schon vorhandenen Systemen gewohntes Verhalten - also dass andere Betriebssysteme auf anderen Partitionen in das GRUB-Menü übernommen werden - durch ein Upgrade auf eine neuere Ubuntu-Version gebrochen wird. Sie greift allerdings im Moment nur, wenn man auf seinem System mit dem Standardpfad **/boot/grub/grub.cfg** arbeitet. Dies sollte zwar für die allermeisten Systeme zutreffen, falls aber nicht, so werden hier bei einem System-Upgrade von z.B. Ubuntu 20.04 auf Ubuntu 22.04 etwaig vorhandene GRUB-Menü-Einträge für andere Betriebssysteme auf dem System trotz `auto`-Option nicht übernommen.

In einem solchen Fall muss man dann ausdrücklich `GRUB_DISABLE_OS_PROBER=false` setzen und gegebenenfalls das Paket **os-prober** nachinstallieren.

Gleiches gilt, wenn man andere Betriebssysteme nachträglich auf dem System installiert. Auch in einem solchen Fall werden in dem schon vorhandenen Ubuntu die nachträglich installierten Betriebssysteme nur in das GRUB-Menü aufgenommen, wenn wie im vorhergehenden Absatz dargelegt nachgearbeitet wird.

GRUB_OS_PROBER_SKIP_LIST

Wenn `os-prober` im Rahmen von `grub-mkconfig` bzw. `update-grub` ausgeführt wird (siehe **GRUB_DISABLE_OS_PROBER**), kann man mit Hilfe dieser Variablen Partitionen und bei Verwendung des UEFI-Modus .efi-Dateien von der Durchsuchung durch `os-prober` explizit ausschließen.

- **Mögliche Werte**: `"UUID@Partition"`, `"UUID@.eif-Datei"`
  - `"UUID@Partition"`: Die auszuschließende Partition wird mit Ihrer Dateisystem-**UUID** [https://wiki.ubuntuusers.de/UUID/] gefolgt von einem `@` gefolgt von der Linux-Gerätenummer der Partition angegeben. Mehrere Partitionen sind durch Leerzeichen voneinander zu trennen und insgesamt in Anführungszeichen einzufassen.
    - Beispiele:
      - `GRUB_OS_PROBER_SKIP_LIST="BAEE3575EE352B51@/dev/sda2"`
      - `GRUB_OS_PROBER_SKIP_LIST="3575E5B51@/dev/sdb3 24b43e007@/dev/sdb5"`
  - `"UUID@.efi-Datei"`: Die auszuschließende .efi-Datei wird mit der Dateisystem-**UUID** [https://wiki.ubuntuusers.de/UUID/] des Dateisystems angegeben auf dem sie sich befindet, gefolgt von einem `@`, gefolgt vom Pfad ausgehend vom Wurzelverzeichnis der betreffenden Partition. Funktioniert nur im UEFI-Modus.
    - Beispiele:
      - `GRUB_OS_PROBER_SKIP_LIST="58EE-F18B@/efi/Microsoft/Boot/bootmgfw.efi"`
      - `GRUB_OS_PROBER_SKIP_LIST="58EE-F18B@/efi/Microsoft/Boot/bootmgfw.efi 529C90047C7FE0B9@/dev/sda3 10B6739EF6837339@/dev/sda4"`
- **Standard**: Die Variable ist nicht gesetzt, d.h. bei jedem Durchlauf von `os-prober` werden stets alle Partitionen gemountet und nach Bootmanagern von Betriebssystem durchsucht.

Hinweis:

Es ist sinnvoll das Durchsuchen von Partitionen nach Betriebssystemen durch `os-prober` mit dieser Option auf die Partitionen und UEFI-Apps zu beschränken, die auch wirklich für den Systemstart von Bedeutung sind.

Weiterhin ist die Nutzung der Option geboten, wenn z.B. neben dem regulären Windows ein `Recovery` vorhanden ist, was zu Missverständnissen oder die Auswahl sogar zu Schäden führen kann.

GRUB_DISABLE_SUBMENU

Mit dieser Variablen kann man festlegen, ob die gefundenen Kernel-Einträge im GRUB-Menü alle auf einer Ebene dargestellt werden sollen oder ob es eine Unterteilung der Darstellung in einen Haupteintrag für den jeweils aktuellen Kernel und einen Untereintrag für die restlichen Kernel geben soll.

- **Mögliche Werte**: boolescher Ausdruck `true`, `false`
  - `true`: Es werden alle gefunden Kernel direkt im GRUB-Menü angezeigt. Auf eine Unterteilung der Darstellung in Haupteintrag und Unter-Eintrag wird verzichtet.
  - `false` oder nicht gesetzt: Im Grub-Menü wird der aktuelle Kernel als Haupteintrag angezeigt. Die restlichen zur Auswahl stehenden Kernel werden in einem eigenen Menü-Punkt `Advanced Options for...` nochmals zusammen mit dem Standardkernel zusammengefasst und sind erst nach Aktivierung dieses Eintrages sichtbar.
- **Standard**: nicht gesetzt, das GRUB-Menü wird also strukturiert in einen Haupteintrag und einen Menü-Punkt `Advanced Options for...` unterteilt dargestellt.

GRUB_ENABLE_CRYPTODISK

Mit Hilfe dieser Variablen kann man `grub-install` veranlassen, nach verschlüsselten Laufwerken zu suchen und die für den Zugriff durch GRUB während des Systemstarts notwendigen Module in das **core.img** zu integrieren.

- **Mögliche Werte**: `y`
  - `y`: `grub-install` wird nach verschlüsselten Laufwerken suchen und die notwendigen Module zur Entschlüsselung in das **core.img** integrieren.
- **Standard**: nicht gesetzt

Hinweis:

Wird dies Variable auf `y` gesetzt, ist kein unbeaufsichtigter Start mehr möglich, weil GRUB auf die Eingabe der Passphrase zum Öffnen des verschlüsselten Laufwerks wartet, bevor das Menü angezeigt wird. Beim Start des verschlüsselten Systems wird man ein weiteres Mal zum Eingeben der Passphrase aufgefordert.

Außerdem muss beachtet werden, dass bei Änderungen an dieser Variablen im Anschluss stets `grub-install` ausgeführt werden muss.

GRUB_INIT_TUNE

Mit Hilfe dieser Variablen kann man einen Ton oder eine Folge von Tönen über den Lautsprecher des Rechners ausgeben lassen, sobald das GRUB-Menü zur Eingabe bereitsteht.

- **Mögliche Werte**: Folge von ganzen Zahlen, eingeschlossen in doppelte Anführungszeichen nach dem Schema: `"Tempo [Ton1-Höhe Ton1-Dauer] [Ton2-Höhe Ton2-Dauer] [Ton3-Höhe Ton3-Dauer] ..."`
  - `Tempo`: Ganzzahliger Wert, der als Basiswert für die Tondauer-Angaben dient. `60` steht für eine Sekunde, `120` steht für eine halbe Sekunde usw.
  - `Tonhöhe`: Ganzzahlige Angabe in Hertz. Eine Hertz-Angabe von `0` erzeugt eine Pause.
  - `Tondauer`: Ganzzahlige Anzahl von Tempo-Einheiten
- Beispiele:
  - Einzelner Ton (Kammerton A): `GRUB_INIT_TUNE="480 440 1"`
- **Standard**: nicht gesetzt, es wird also kein Ton ausgegeben.

Hinweis:

Das eingestellte **Timeout** für das GRUB-Menü beginnt erst zu laufen nachdem der Ton oder die Tonfolge zu Ende gespielt wurde. Eine Auswahl durch den Benutzer ist also auch erst im Anschluss an Ton oder Tonfolge möglich.

GRUB_RECORDFAIL_TIMEOUT

Mit dieser Variablen lässt sich die Zeit einstellen, die das Auswahlmenü angezeigt wird, falls es beim normalen Start zu Problemen kommt.

- **Mögliche Werte**: Zahl
  - `-1`: Schaltet die Timeout-Funktion ab, d.h. gebootet wird nur nach Eingabe durch den Benutzer.

    - `postive Zahl X`: Wartet ohne Anwender-Insteratkion `X` Sekunden bevor der unter **GRUB_DEFAULT** gesetzte Eintrag gestartet wird.
        - Beispiel: `GRUB_RECORDFAIL_TIMEOUT=15`
- **Standard**: `GRUB_RECORDFAIL_TIMEOUT=30`

GRUB_RECOVERY_TITLE

Mit dieser Variablen lässt sich der Text einstellen der hinter einem Kernel-Eintrag in Klammern steht und dem Anwender anzeigt, dass es sich bei dem Kernel-Eintrag um einen zur Wiederherstellung des Systems handelt.

- **Mögliche Werte**: Beliebige Zeichenkette, die in Anführungszeichen eingefasst werden muss, sofern sie Leer- oder Sonderzeichen enthält.
- **Standard**: `GRUB_RECOVERY_TITLE="recovery mode"`

GRUB_FORCE_PARTUUID

Mit dieser Variable lässt sich eine bestimmte Partitions-UUID angeben, die dem Kernel das Root-Dateisystem mitteilen soll. Ist diese Variable gesetzt überschreibt sie alle anderen etwaig festgelegten Arten, die sonst das Root-Dateisystem an den Kernel übergeben.

- **Mögliche Werte**: `Partitions-UUID`
    - Beispiel: `GRUB_FORCE_PARTUUID="dfbc3bd0-2b21-4e5a-8b16-d0036327d8f9"`
- **Standard**: nicht gesetzt

Links

Intern

- **GRUB 2 Übersicht** [https://wiki.ubuntuusers.de/GRUB_2/]
    - **GRUB 2 Probleme und Lösungen** [https://wiki.ubuntuusers.de/GRUB_2/Problembehebung/]
    - **GRUB 2 Installation** [https://wiki.ubuntuusers.de/GRUB_2/Installation/]
    - **GRUB 2 Skripte** [https://wiki.ubuntuusers.de/GRUB_2/Skripte/]
    - **GRUB 2 Reparatur** [https://wiki.ubuntuusers.de/GRUB_2/Reparatur/]
    - **GRUB 2 Umgebung analysieren** [https://wiki.ubuntuusers.de/GRUB_Umgebung_analysieren/]
    - **GRUB 2 Shell** [https://wiki.ubuntuusers.de/GRUB_2/Shell/]
    - **GRUB 2 optisch anpassen** [https://wiki.ubuntuusers.de/GRUB_2/Aussehen/]
    - **GRUB 2 Grundlagen** [https://wiki.ubuntuusers.de/GRUB_2/Grundlagen/]
- **GRUB Customizer** [https://wiki.ubuntuusers.de/GRUB_Customizer/] - grafisches Werkzeug zum Bearbeiten der Bootmanager-Konfiguration
- **efibootmgr** [https://wiki.ubuntuusers.de/efibootmgr/] - Konfiguration des (U)EFI Bootmenüs

Extern

- **Grub 2 Handbuch** [https://www.gnu.org/software/grub/manual/] □
- **community/Grub2** [https://help.ubuntu.com/community/Grub2]
    - **Password Protection** [https://help.ubuntu.com/community/Grub2#Password_Protection] Passwortschutz für GRUB 2
- **Grub 2 (Title) Tweaks** [https://ubuntuforums.org/showthread.php?t=1287602] - diverse manuelle GRUB-Anpassungen

---

Diese Revision [https://wiki.ubuntuusers.de/GRUB_2/Konfiguration/a/revision/1014229/] wurde am 7. April 2024 16:25 von **Berlin_1946** erstellt.
Die folgenden Schlagworte wurden dem Artikel zugewiesen: **Bootloader** [https://wiki.ubuntuusers.de/wiki/tags/Bootloader/], **Installation** [https://wiki.ubuntuusers.de/wiki/tags/Installation/], **System** [https://wiki.ubuntuusers.de/wiki/tags/System/]