Wunderplatte

Viele Distributionen per USB booten

Onkobu Tanaake

2018-09-20T10:00:00

Was im Folgenden gebraucht wird:

- UAS-fähiges Fesplattengehäuse mit Festplatte
- primärer Linux-Host, um Partitionieren und die Installation betreiben zu können
- auf dem Host: ein Partitionierungsprogramm, fdisk, gdisk, parted, …
- auf demselben Host: Werkzeuge zur Informationsbeschaffung über die Platte, dmesg, smartctl, lsblk, blkid
- ebenso das Kommandozeilenprogramm `dd`, um die ISOs auf den USB-Stick schreiben zu können
- Kenntnisse im Umgang mit wenigstens einem Texteditor für die Kommandozeile, vim, nano oder emacs, eine Grub-Konfiguration muss bearbeitet werden
- jede Menge ISOs der zu installierenden Linux-Distributionen und ein weiterer USB-Stick, das ISO der Installation bereitzustellen
- Etwa einen Tag Zeit

Ich verbrachte ein paar Stunden damit, das richtige Gehäuse zu finden. Geschwindigkeit ist kein wirkliches Kriterium, denn per USB 3.0 zu booten ist nicht möglich. Allerdings wollte ich auch nicht Stunden mit der Installation zubringen. Und vielleicht möchte ich die Platte auch in den Schnellwechselrahmen meines Desktops stecken. So entschied ich mich für ein durchschnittliches Icy Box Gehäuse und eine durchschnittliche 1TByte Western Digital 2,5" Platte. Das Gehäuse kann USB attached SCIS oder Kurz UAS (VL711 Chipsatz). Die Festplatte war zur Hand und leidet nicht an Stromspar-Eskapaden wie Seagate-Platten (in Intenso-Gehäuse verbaut).

Die Reihenfolge ist wichtig. Linux-Installer versuchen clever zu sein und benutzen Varianten des Programmes os-prober. Dieser kleine Sonderling sucht auf den erreichbaren Festplatten nach anderen Betriebssystemen. Zusammen mit Grub – dem Bootloader der Wahl – erzeugt das eine nur halb boot-bare Installation. Deswegen die nachstehende Reihenfolge:

1. Partitionieren der Festplatten, USB boot-bar

2. Installation komischer Linux-Varianten wie Ubuntu or Mint

3. Installation ernster Varianten wie Debian

4. Installation des verlässlichen unverfälschten Arch Linux

5. Reparatur von /boot/grub/grub.cfg per Hand

Die Platten zu partitionieren ist noch immer eine schräge Sache und es ist viel zu leicht es falsch zu machen. Es gibt wenig Hilfe da draußen bis auf die technischen Einschränkungen. Durch Grub braucht es eine winzige Partition ganz am Anfang. Dort legt Grub seinen eigenen Startcode ab. Der ist nicht mit dem Mount-Punkt `/boot` zu verwechseln. Dieser wird die zweite Partition und als VFAT (FAT32). Es wäre nicht wirklich als VFAT notwendig, wenn man nicht Windows vom selben Laufwerk nutzen will. Ich tue es einfach, weil ich es kann. Da Grub das Booten übernimmt, könnte es auch ein ext2 sein (oder ext4, was aber bei 512MByte zuviel Verwaltung bedeutet). Schließlich kommen die übrigen vielen Partitionen um je eine Distribution komplett aufzunehmen. Ich lasse auch den Auslagerungsspeicher (Swap) weg. Linux läuft sehr gut ohne. Das USB-bootbare Spielzeug braucht keine solche Speicherverlängerung und es soll auch kein Hibernate-Modus zum Einsatz kommen. Jeder kann sein Lieblingswerkzeug zur Partitionierung einsetzen. Ich bevorzuge fdisk aber auch gdisk, gparted, parted oder jedes andere sind absolut in Ordnung.

Warnung

Prüfe die Zielfestplatte für die Installation mehrmals. Der Gerätename und die Modellbezeichnung sind sehr wichtig. Nach dem Einstecken der Festplatte muss mit `lsblk` und `dmesg` der Gerätename festgestellt werden. Im Folgenden verwende ich den kaputten Platzhalter /dev/sdX. Für dich kann es /dev/sdc o.ä. sein. Die Modellbezeichnung ist ebenso sehr wichtig und sollte aufgeschrieben werden, bspw. ST500LM000-1EJ162 (`smartctl -a /dev/sdX | grep 'Device Model'`). Die verschiedenen Installationsprogramme den Gerätenamen ändern und keinesfalls darf eine bestehende Linux-Installation des primären Hosts verändert werden.

- Eine neue GPT Partitionstabelle für /dev/sdX anlegen, es werden viele Partitionen gebraucht
- Eine erste Partition von 1MByte Größe ganz an den Anfang legen, Grub braucht nur 30kByte, genügt also vollauf
- Diese Partition muss aus Bios Boot (fdisk) oder 0xEF02 (gdisk) markiert werden, bzw. ist das Flag bios_grub (parted) zu setzen
- Eine zweite Partition von 512MByte anlegen, Windows File System/ FAT
- Weitere Partitionen von mindestens 10GByte Größe, eine je gewünschter Distribution, in diesem Beispiel Nummern 3-6
- Partitionstabelle schreiben und das Programm zum Partitionieren beenden
- Das FAT32-Dateisystem für die Partition #2 anlegen, `mkfs.vfat /dev/sdX2`
- Die ext4 Dateisysteme für die Partitionen 3-6, `for i in 3 4 5 6; do mkfs.ext4 /dev/sdX$i; done`

Ubuntu wird als erstes installiert. Das braucht etwa 2h. Zuerst ist das ISO zu beschaffen und per `dd` auf den USB-Stick zu schreiben. Das kann man nun booten – die Zielfestplatte ist an einen USB-3-Anschluss gesteckt. Ich benutzte 20.04-irgendwas und bekam einen schönen Absturz am Ende der Installation. Ich hatte die Installation von Drittanbieter-Paketen/ Media-Codecs angehakt. Die Installation selbst ist aber einwandfrei nutzbar. Ubuntu lässt auch eine kaputte Grub-Installation zurück und schließt auch os-prober-Aktionen ein. Es ist auch unfähig, die Partition

#2 als Mountpunkt `/boot` zu verwenden. Partition #3 ist die einzige, die bei der Installation verwendet wird. Wenn man eine Warnungsmeldung erhält, keine Änderungen an Partitionen oder Dateisystemen seien vorgenommen worden, dann ist alles richtig gelaufen.

Nach den Herunterfahren ist Linux Mint dran. Es ist die selbständigste Installation. Man muss nichts tun, bis auf die Wahl der Zielpartition läuft es von allein. Wie bei Ubuntu ist die Grub-Konfiguration am Ende wieder/ immer noch kaputt. Ich verwende grundsätzlich nur die Xfce-Variante. Partition #2 kann nicht genutzt werden. Hier wird auch nur Partition #4 verwendet. Wie bei Ubuntu weist der Installer darauf hin, dass keine Änderungen an der Partitionstabelle oder den Dateisystemen gemacht würden.

```
archroot> mount /dev/sdX3 /mnt
archroot> mv /boot/*-5.4.* /boot
archroot> umount /mnt
```

Danach startete ich `grub-mkconfig` wie o.a. und korrigierte alle Menüeinträge. Dafür sind alle UUIDs der Partitionen 3-6 notwendig, was von `blkid` angezeigt wird. Per grep oder anders gefiltert kann das direkt an grub.cfg angehängt werden: `blkid >> /boot/grub/grub.cfg`. Mit dem Lieblingseditor muss diese Datei nun geöffnet und bearbeitet werden. Für jede Distribution gibt es einen `menuentry` der einen Block mit geschweiften Klammern einschließt. Darin auch der `linux`-Aufruf mit *root=UUID=*. Grub nahm an, alle liegen in der Arch-Partition #6. Stattdessen sind für Ubuntu, Mint und Debian deren UUIDs der Partition einzutragen. Einfach am Ende der Datei ausschneiden und innerhalb des menuentry ersetzen.

```
menuentry 'Gentoo GNU/Linux, with Linux 5.4.48-gentoo-sec' --class
  load_video
  insmod gzio
  insmod part_gpt
  insmod fat
  set root='hd0,gpt1'
  if [ x$feature_platform_search_hint = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt1 -
  else
    search --no-floppy --fs-uuid --set=root 4E3D-1ADC
  fi
  echo 'Loading Linux 5.4.48-gentoo-sec ...'
  #                                              vvvvvvvvvvvvvvvvvvv
  linux /vmlinuz-5.4.48-gentoo-sec root=UUID=b00e271d-8c78-4175-a3
  #                                              ^^^^^^^^^^^^^^^^^^^
  echo 'Loading initial ramdisk ...'
  initrd /amd-uc.img /initramfs-5.4.48-gentoo-sec.img
```

```
}
```

Warum war eine virtuelle oder containerisierte Installation unzureichend? Ich brauche Hardware-Zugriff und ein vollständiges Linux. Das schließt Desktops mit ein, den Touch-Monitor, den I²C-Bus oder Hardwarezugriff per `smartctl`. Der Netzwerkstapel muss auch voll zugänglich sein. Das bringt auch Einschränkungen mit sich: Es ist sehr schwer, die Distributionen aktuell zu halten. Die meisten starten nach der Aktualisierung des Kernels oder der *linux-firmware* noch per Trigger eine Grub-Konfiguration per `grub-mkconfig`.

Als Schlussbemerkung verweise ich auf die Sicherheitsaspekte dieser unverschlüsselten ko-existierenden Betriebssysteme und der Nutzerinhalte. Nur eine der Distributionen kann gestartet werden. Nichtsdestotrotz hat man damit root-Zugriff auf alle übrigen. Einfach die fremden Partitionen per `sudo` mounten und Dateien können kopiert oder Konfigurationen geändert werden. Bis auf Arch legen alle anderen Distributionen einen normalen Nutzer mit solchen sudo-Rechten an. Ubuntu macht es zwar etwas schwerer `ls` auf nur-root-Verzeichnisse anzuwenden. Aber zumindest der Nutzerinhalt ist an die numerische ID (UID) gebunden. Alle beginnen die Zählung mit 1000 und selbe UID bedeutet selber Nutzer. Ein home-Verzeichnis für UID 1000 ist von allen anderen Nutzern mit derselben UID direkt schreib- und lesbar. Ein kleines Arch-Linux-Stäbchen mit ein paar Werkzeugen, NTFS-Schreibzugriff und Netzwerküberwachung wirkt Wunder. Kein Passwortschutz in der Boot-Konfiguration: einstecken und laden lassen. Keine verschlüsselten Festplatten: eine Einladung zu ändern.