

```
/=\/=\  
|====|  
\\==/  
\\/  

```

Kernel anpassen

Die Umgebung in welcher ein Kernel gebaut wird braucht ca:

- 1GB RAM
- 8GB Disk Space
- Arbeitsverzeichnis erstellen und reinwechseln

```
mkdir uebung && cd uebung
```

- auf kernel.org gehen und URL des aktuellsten Kernels notieren
- Kernel runterladen. Z.B.

```
apt-get install wget vim  
wget \  
https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.8.2.tar.xz
```

- Kernel auspacken

```
tar xzvf linux-6.8.2.tar.xz
```

- ins Kernel Quellcode-Verzeichnis wechseln

```
cd linux-6.8.2
```

- Abhängigkeiten des Kernel Builds installieren:

- unter Debian/Ubuntu/etc.

```
apt-get install ncurses-dev pkg-config bc gcc  
libc6-dev make bzip2 binutils \  
dpkg-dev flex bison libssl-dev \  
libelf-dev rsync debhelper-compat
```

- unter Fedora/RedHat/CentOS/etc,

```
yum install bison flex bc rpm-build bc gcc make \  
bzip2 ncurses-devel  
# nicht getestet!
```

- Kernel konfigurieren

- wer sich Mal anschauen möchte, was man so alles im Kernel konfigurieren kann:

```
make menuconfig
```

- per default ist sehr viel drin und einen entsprechenden Kernel kompilieren kann sehr lange dauern. Deshalb empfiehlt es sich, so viel Unnötiges wie möglich wegzukonfigurieren, damit die Bauzeit kürzer wird.

- im Netz findet man Konfigurationen für Kernel, darunter diese hier für Virtual-Box: <https://raw.githubusercontent.com/EvilOlaf/vbox-guest-config/master/linux/linux-4.15.x-server.config> Diese kann man nach `linux-6.8.2/.config` kopieren (und wenn der Kernel gebaut wird bei Fragen alles mit Return bestätigen).

- `vim +421 arch/x86/boot/compressed/misc.c`

- folgende Zeile finden:

```
debug_putstr("done.\nBooting the kernel.\n");
```

- In nächste Zeile eine freie Meldung reinschreiben:

```
debug_putstr("Hossa, mein eigenes, verbessertes OS!\n");
```

- Paket des Kernels bauen

- Debian

```
sudo make bindeb-pkg
```

- dauert lange (28min in einer VirtualBox mit 384M RAM und einer CPU)

- rpm

```
sudo make binrpm-pkg
```

- dauert lange
 - Paket ist unter `/root/rpmbuild/RPMS/$ARCH/kernel-6.8.2-1.i386.rpm`

- im Falle, dass man den neuen Kernel ausserhalb der VM gebaut hatte, diesen nun hineinkopieren:

- Parameter des folgenden Kommandos müssen an lokale Gegebenheiten angepasst werden:

```
scp -P 1234567 \  
  ../linux-image-6.8.2_6.8.2-1_amd64.deb \  
  localhost:/tmp
```

- neuen Kernel in VM installieren

```
cd /dorthin_wo_linux-image-6.8.2_6.8.2-1_amd64.deb_ist  
# (entweder unter /tmp oder ../)
```

- Debian

```
dpkg -i linux-image-6.8.2_6.8.2-1_amd64.deb
```

- rpm

```
rpm -i kernel-6.8.2-1.i386.rpm  
vim /etc/grub.d/40_custom
```

- menu entry hinzufügen, analog zu /boot/grub2/grub.conf
grub2-mkconfig -o /boot/grub2/grub.cfg

- VM neustarten

Post scriptum:

In der Theorie sollte es möglich sein, sich einen Kernel zu konfigurieren, welcher der aktuell laufenden Konfiguration entspricht, das scheint aber leider nicht korrekt zu funktionieren:

```
make localmodconfig
```

Quellen:

- <https://help.ubuntu.com/community/Kernel/Compile>
- <https://kernel-team.pages.debian.net/kernel-handbook/>