Der Kernel:

Der Kernel ist die Schnittstelle zwischen Hard und Software, der Kernel kümmert sich also darum unserem System Zugriff auf vorhandene Hartware zu gewähren.

Modular oder Monolithisch:

Ein Monolithischer Kernel, beinhaltet alle Treiber und Funktionen die eventuell mal benötigt werden könnten und wird dadurch sehr groß, ein eigener Monolithischer Kernel beinhaltet meist alles was auf dem eigenen System benötigt wird. Monolithische Kernel können keine Module und Funktionen nachladen, daher sind sie nicht gerade Flexibel.

Modulare Kernel werden heute der Normalfall sein, hier sind nur grundlegende Funktionen und Treiber im Kernel vorhanden, alles andere kann Später je nach bedarf nachgeladen werden, so ist es möglich einen Kernel zu bauen der auf einer Vielzahl von Rechnern Laufen kann ohne Anpassungen vornehmen zu müssen. Auch gibt es viele Treiber von Drittanbietern, (nvidia, ati ...) die man gar nicht fest in den Kernel einbauen kann.

Voraussetzungen:

Informationen über die Hardware
Kernel Source (http://www.kernel.org)
gcc
make
build-essential
(lib)ncurses5-dev (für make menuconfig)

Kernel Source nach /usr/src kopieren

```
cp /server/kernel/linux-2.6.27.6.tar.gz /usr/src
```

Kernel Source auspacken

```
cd /usr/src
tar -xzf linux-2.6.27.6.tar.gz
```

Den Kernel Source Linken, da alle weiteren Schritte davon ausgehen, das wir im Verzeichnis /usr/src/linux arbeiten:

```
ln -s linux-2.6.27.6 linux
cd linux
```

Kernel Konfigurieren:

1. Alte Konfiguration übernehmen und Anpassen

```
cp /boot/config-2.6.18-6-486 /usr/src/linux/.config
zcat /proc/config.gz > /usr/src/linux/.config
```

Alte Konfigurationsdatei an den neuen Kernel anpassen

```
make oldconfig
```

2. Kernel Konfigurieren:

(EXTRAVERSION anpassen (***vi Makefile***)

```
                make config
.config  ----> make menuconfig  -->  .config
                make xconfig
```

Kernel Kompilieren

```
make clean          → Entfernt alle schon Kompilierten teile
make dep            → Überprüft die Abhängigkeiten
make bzImage        → Erstellt das bzImage
make modules        → Erstellt die Module
make modules_install → Installiert die Module
```

```
make && make modules_install
```

Den Kernel ins System installieren:

```
x86:      cp arch/i386/boot/bzImage /boot/kernel-UEBUNG
          cp System.map /boot
x86_64:   cp arch/x86_64/boot/bzImage /boot/kernel-UEBUNG
          cp System.map /boot
```

Den neuen Kernel im Bootloader eintragen: /boot/grub/menu.lst

```
title           Debian GNU/Linux, kernel 2.6.27-6 UEBUNG
root            (hd0,0)
kernel          /boot/kernel-UEBUNG vga=0x317 root=/dev/sda5
```

6. Neustarten

Kernelmodule:

Kernelmodule sind Treiber und Funktionen, die bei laufendem System nachgeladen oder entladen werden können.

Anzeigen gerade geladener Module:

Mit dem Befehl "***lsmod***" werden alle gerade geladenen Module angezeigt. die Ausgabe könnte folgendermaßen aussehen:

```
Module                  Size  Used by
rfcomm                 27604  0
l2cap                  16772  5 rfcomm
bluetooth              43876  4 rfcomm,l2cap
```

Sollte bei "Used" mal die "-1" auftauchen, heißt das, dass das Modul selber entscheidet ob es entladen werden kann.

Informationen über Module:

Mit dem Befehl "***modinfo <modulname>***" werden Informationen über ein Modul angezeigt. "modinfo" hat auch ein paar Optionen:

```
-d          -> Zeigt die Modulbeschreibung
-a          -> Zeigt den Author
-l          -> Zeigt die Lizenz
-p          -> Zeigt mögliche Modulparameter
-n          -> Dateinamen des Moduls
-F depends -> Zeigt die Modulabhängigkeiten
```

Entladen von Modulen (alte Variante) :

Mit dem Befehl "***rmmod <Modulname>***" kann ein Modul entladen werden, jedoch nur dann wenn dieses Modul von keinem anderem Modul benötigt wird.

* → "***rmmod -a***" 2 mal ausgeführt, sollte alle nicht benutzten Module entladen

Laden von Modulen (alte Variante) :

Mit dem Befehl "***insmod /path/zum/modul.ko***" kann ein modul geladen werden, insmod nimmt keinerlei Rücksicht auf Modulabhängigkeiten.

Laden und entladen von Modulen (neu Variante) :

Der Befehl "***modprobe***" wird zum Laden und Entladen von Modulen verwendet, ***modprobe*** berücksichtigt dabei auch die Modulabhängigkeiten. Die Modulabhängigkeiten stehen in der Datei "***/lib/modules/<kernelversion>/modules.dep***" und diese Datei wird mit dem Befehl "***depmod***" erstellt.

modprobe <Modulname>	→ Lädt das angegebene Modul incl. aller Module von denen es Abhängig ist.
modprobe -r <Modulname>	→ entlädt das angegebene Modul incl. aller Module von denen es Abhängig ist.
modprobe -l	→ Zeigt alle Module an die es für den Aktuellen Kernel gibt
modprobe -c	→ Zeigt die Aktuelle Modul-Konfiguration.

Modulkonfiguration:

Bei den meisten Modulen können Parameter an die Module übergeben. Dies geschieht beim Laden des Modules, im z.B.: der Netzwerkarte den IRQ 7 zuzuweisen:

insmod /lib/modules/2.6.23-7/kernel/net/forcedeth.ko irq=7
modprobe forcedeth irq=7

Um nun nicht alle benötigten Optionen u.ä. jedesmal beim laden wieder angeben zu müssen, gibt es für "modprobe" Konfigurationsdateien

Kernel 2.2.x	***/etc/conf.modules***
Kernel 2.4.x	***/etc/modules.conf***
Kernel 2.6.x	***/etc/modprobe.conf*** ***/etc/modprobe.d/****

vi /etc/modprobe.d/eigene

In diesen Dateien können folgende Dinge festgelegt werden:

- Modulparameter
- Aliasnamen für Module
- Module die vor oder nach einem Modul geladen werden sollen
- Programme die ausgeführt werden sollen anstelle das modul zu laden

1. ***alias bluetooth2 rfcomm***
2. ***install rfcomm <Befehl>***
3. ***remove rfcomm <Befehl>***
4. ***pre-install rfcomm <Befehl>***
5. ***post-install rfcomm <Befehl>***
6. ***pre-remove rfcomm <Befehl>***
7. ***post-remove rfcomm <Befehl>***
8. ***option forcedeth irq=7***

zu 1.) Hier wird dem Modul "rfcomm" der Aliasname "bluetooth2" gegeben, wird nun "modprobe bluetooth2" eingegeben, wird das Modul "rfcomm" geladen.
zu 2.) Wird "modprobe rfcomm" eingegeben, wird nicht das Modul geladen sondern ein Befehl ausgeführt.
zu 3.) Wird "modprobe -r rfcomm" eingegeben, wird nicht das Modul entladen sondern ein Befehl ausgeführt.
zu 4.) Vor dem Laden des Modules "rfcomm" wird ein Befehl ausgeführt.
zu 5.) Nach dem Laden des Modules "rfcomm" wird ein Befehl ausgeführt.
zu 6.) Vor dem Entladen des Modules "rfcomm" wird ein Befehl ausgeführt.
zu 7.) Nach dem Entladen des Modules "rfcomm" wird ein Befehl ausgeführt.
zu 8.) Das Modul "forcedeth" wird immer mit dem Parameter "irq=7" geladen.

Parameter an Treiber die in den Kernel eingebaut sind übergeben:

Dies wird durch Kernelparameter erledigt, dieser werden im BootLoader eingetragen bei GRUB in der "/boot/grub/menu.lst" z.B.:

```
title           Debian GNU/Linux, kernel 2.6.27-6 UEBUNG
root            (hd0,0)
kernel          /boot/kernel-UEBUNG vga=0x317 root=/dev/sda5
forcedeth:irq=7
```