make config
make menuconfig
make xconfig
make defconfig

make oldconfig' is used to update your old .config file to the newer kernel. For exapmle, you have .config file of your current kernel and you downloaded new kernel and want to build your new kernel.
“`make oldconfig` takes the current kernel configuration in the `.config` file, and updates it based on the new kernel release.”

Compile the Linux Kernel

Compile the main kernel:

```
# make
```

Compile the kernel modules:

```
# make modules
```

Install the kernel modules:

```
# make modules_install
```

Module dependencies

Some kernel modules can depend on other modules, which need to be loaded first

Example: the `ubifs` module depends on the `ubi` and `mtd` modules.

Dependencies are described both in
`/lib/modules/<kernel-version>/modules.dep`
and in
`/lib/modules/<kernel-version>/modules.dep.bin`
These files are generated when you run
`make modules_install`

Kernel log

When a new module is loaded, related information is available in the kernel log.

The kernel keeps its messages in a circular buffer (so that it doesn't consume more memory with many messages)

Kernel log messages are available through the `dmesg` command (diagnostic message)

Note that you can write to the kernel log from user space too:
`echo "<n>Debug info" > /dev/kmsg`

`modinfo <module_name>` (for modules in `/lib/modules`)
`modinfo <module_path>.ko`

Gets information about a module without loading it: parameters, license, description and dependencies.

Sudo insmod <module_path>.ko
Tries to load the given module. The full path to the module object file must be given.

Understanding module loading issues

->when loading fails, insmod often does not give you enough details.
-> details are often available in the kernel log.

Example:
```
$ sudo insmod ./intr_monitor.ko
insmod: error inserting './intr_monitor.ko': -1 Device or resource busy
$ dmesg
[17549774.552000] Failed to register handler for irq channel 2
```

`sudo modprobe <module_name>`
Most common usage of `modprobe` : tries to load all the modules the given module depends on, and then this module. Lots of other options are available.

`Modprobe` automatically looks in `/lib/modules/<version>/` for the object file corresponding to the given module name.
`lsmod`
Displays the list of loaded modules Compare its output with the contents of `/proc/modules` !

`sudo rmmod <module_name>`
Tries to remove the given module.
Will only be allowed if the module is no longer in use (for example, no more processes opening a device file)
`sudo modprobe -r <module_name>`
Tries to remove the given module and all dependent modules (which are no longer needed after removing the module)

Passing parameters to modules
Find available parameters:

```
modinfo usb-storage
```

Through
`insmod`:
```
sudo insmod ./usb-storage.ko delay_use=0
```

Through `modprobe`:
Set parameters in `/etc/modprobe.conf` or in any file in`/etc/modprobe.d/`:
```
options usb-storage delay_use=0
```

Through the kernel command line, when the driver is built statically into the kernel:
```
usb-storage.delay_use=0
```

`->usb-storage` is the driver name

`->delay_use` is the driver parameter name. It specifies a delay before accessing a USB storage device (useful for rotating devices).

`-> 0` is the driver parameter value

Q.How to find the current values for the parameters of a loaded module?
Check `/sys/module/<name>/parameters`.
There is one file per parameter, containing the parameter value.