Upgrade/downgrade Yocto Kernel

In this example, I'll guide you through upgrading the Yocto kernel from **6.6 to 6.12**, including all necessary `.conf`, `.bb`, and `.bbappend` file changes, commands, and expected outputs.

Check the Current Kernel Version

Before making any changes, check the **current kernel version** being used:

```
bitbake -e virtual/kernel | grep ^PREFERRED_VERSION
```

Expected output:

```
PREFERRED_VERSION_linux-yocto="6.6"
```

Check Available Kernel Versions

List all available kernel versions in your **Yocto layers**:

```
ls meta*/recipes-kernel/linux/
```

Expected output:

```
linux-yocto_6.6.bb
linux-yocto_6.5.bb
linux-yocto_5.15.bb
```

If **6.12 is not available**, you may need to add a new `.bb` recipe manually.

Set the New Kernel Version in `local.conf` or Machine Config

We need to specify the new kernel version (`6.12`) in `local.conf` or the **machine configuration file**.

Option 1: Modify `local.conf` (Temporary for this Build)

```
vi build/conf/local.conf
```

Add or modify:

```
PREFERRED_VERSION_linux-yocto = "6.12"
```

Option 2: Modify Machine Configuration File (Permanent Change)

```
vi meta-yourlayer/conf/machine/yourmachine.conf
```

Add or modify:

```
PREFERRED_VERSION_linux-yocto = "6.12"
```

Modify the Kernel Recipe (`.bb` and `.bbappend`)

If the **6.12 kernel recipe is not present**, we must create it.

A. Create a New Kernel Recipe (`linux-yocto_6.12.bb`)

Navigate to the kernel recipe directory:

```
cd meta-intel/recipes-kernel/linux
```

Copy an existing kernel recipe and rename it:

```
cp linux-yocto_6.6.bb linux-yocto_6.12.bb
```

Edit the new `linux-yocto_6.12.bb`:

```
vi linux-yocto_6.12.bb
```

Modify:

```
require recipes-kernel/linux/linux-yocto.inc

DESCRIPTION = "Linux Yocto Kernel 6.12"
KERNEL_VERSION = "6.12"
SRCREV = "abcdef1234567890"  # Get the correct commit from upstream
SRC_URI = "git://git.yoctoproject.org/linux-yocto.git;branch=standard/base;name=machine"

# Update checksum
SRC_URI[sha256sum] = "new_checksum_value"

# Define the compatible machines
COMPATIBLE_MACHINE = "yourmachine"
```

Modify or Create a `.bbappend` File

Now, create a `.bbappend` file to apply **custom configurations** for `6.12`.

Navigate to the appropriate directory:

```
mkdir -p meta-yourlayer/recipes-kernel/linux
cd meta-yourlayer/recipes-kernel/linux
```

Create:

```
vi linux-yocto_6.12.bbappend
```

Add:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"

SRC_URI_append = " file://custom_kernel_patch.patch"

KERNEL_FEATURES_append = " features/custom_feature.scc"

COMPATIBLE_MACHINE = "yourmachine"
```

Clean and Update the Environment

Clean previous builds to avoid conflicts:

```
bitbake -c cleanall virtual/kernel
```

Verify the Kernel Selection

Run:

```
bitbake -e virtual/kernel | grep ^PREFERRED_VERSION
```

Expected output:

```
PREFERRED_VERSION_linux-yocto="6.12"
```

Fetch & Validate the Kernel Source

Ensure the kernel fetches correctly:

```
bitbake -c fetch virtual/kernel
```

Verify github commits or logs to get exact hash

Then, copy the new checksum into `SRC_URI[sha256sum]`.

Build the New Kernel

Now, build the kernel:

```
bitbake virtual/kernel
```

If successful, proceed to build the image:

```
bitbake core-image-minimal
```

Deploy & Test

After a successful build, flash the new kernel to the target device:

Copy the generated kernel image to your deployment directory:

```
cp tmp/deploy/images/yourmachine/bzImage /your/boot/partition/
```

1. Boot into the new kernel and check the version:

```
uname -r
```

2. Expected output:

```
6.12.0-yocto-standard
```

Summary of Changes

File	Changes Made
`local.conf`	Set `PREFERRED_VERSION_linux-yocto = "6.12"`
`yourmachine.conf`	Set `PREFERRED_VERSION_linux-yocto = "6.12"` (optional)
`linux-yocto_6.12.bb`	Created new kernel recipe
`linux-yocto_6.12.bbappend`	Added custom patches/features
`bitbake -c cleanall virtual/kernel`	Cleaned old builds
`bitbake virtual/kernel`	Built the new kernel
`bitbake core-image-minimal`	Built the new image

Conclusion

- **Downgrade or upgrade** the kernel version safely by modifying `PREFERRED_VERSION`.
- **Handle checksum issues** by fetching and updating `SRC_URI[sha256sum]`.
- **Rebuild the kernel & root filesystem** to reflect changes.
- **Test on hardware** and confirm with `uname -r`.

How to Add Patches to an Existing Kernel Version Using `.bbappend` in Yocto

If you need to apply custom patches to the kernel in Yocto, you should use a `.bbappend` file. This allows you to extend the existing kernel recipe (`linux-yocto_x.xx.bb`) without modifying the original `.bb` file.

Steps to Add Kernel Patches Using `.bbappend`

We will:

1. Create a `.bbappend` file for the kernel recipe.
2. Store patch files inside a `files/` directory.
3. Modify `SRC_URI` in `.bbappend` to apply patches.
4. Rebuild the kernel to test.

---

1. Locate the Kernel Recipe

First, find the kernel recipe version that is currently being used:

```
bitbake -e virtual/kernel | grep ^PREFERRED_VERSION
```

Example output:

```
PREFERRED_VERSION_linux-yocto="6.6"
```

This means we need to append to `linux-yocto_6.6.bb`.

2. Create a Custom Layer (If Not Already Created)

If you don’t already have a custom layer, create one:

```
cd <yocto-root>
mkdir -p meta-mycustomlayer
cd meta-mycustomlayer
mkdir -p recipes-kernel/linux/files
```

Add the layer to `bblayers.conf`:

```
vi build/conf/bblayers.conf
```

Add:

```
BBLAYERS += "${TOPDIR}/../meta-mycustomlayer"
```

Or

```
Bitbake-layers add-layer meta-mycustomlayer
```

---

3. Create a `.bbappend` File

Create a `.bbappend` file for the kernel:

```
mkdir -p meta-mycustomlayer/recipes-kernel/linux
cd meta-mycustomlayer/recipes-kernel/linux
nano linux-yocto_6.6.bbappend
```

4. Add Patch Files

1. Place your patch files inside `files/`:

```
mkdir -p meta-mycustomlayer/recipes-kernel/linux/files
```

2. Add a new patch file:

```
vi meta-mycustomlayer/recipes-kernel/linux/files/my_fix.patch
```

Example **`my_fix.patch`**:

```
--- a/drivers/net/ethernet/intel/e1000/e1000_main.c
+++ b/drivers/net/ethernet/intel/e1000/e1000_main.c
@@ -102,7 +102,7 @@
 static int e1000_probe(struct pci_dev *pdev, const struct pci_device_id
*ent)
 {
     struct net_device *netdev;
-    int err = -ENODEV;
+    int err = 0;  /* Fixed incorrect error code */
```

---

5. Modify `linux-yocto_6.6.bbappend`

Edit `linux-yocto_6.6.bbappend`:

```
vi meta-mycustomlayer/recipes-kernel/linux/linux-yocto_6.6.bbappend
```

Add:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
SRC_URI_append = " file://my_fix.patch"
do_configure[depends] += "virtual/kernel:do_patch"
```

- `FILESEXTRAPATHS_prepend` tells BitBake to look in `files/` for patches.
- `SRC_URI_append` adds the patch to the kernel source.
- `do_configure[depends]` ensures patches are applied before configuration.

---

6. Rebuild the Kernel

First, clean and rebuild the kernel:

```
bitbake -c clean virtual/kernel
bitbake virtual/kernel
```

7. Verify That the Patch Was Applied

After building, confirm that the patch is applied:

```
cd tmp/work/<machine>/linux-yocto/6.6.0+gitAUTOINC*/build
grep -r "Fixed incorrect error code"
drivers/net/ethernet/intel/e1000/e1000_main.c
```

If the patch was applied, the modified code should be present in the source.

---

8. Deploy & Test

Once built, deploy the kernel:

```
bitbake core-image-minimal
```

Then, copy the kernel to your target hardware and check:

```
uname -r
```

Summary of Changes

File	Description
`meta-mycustomlayer/recipes-kernel/linux/linux-yocto_6.6.bbappend`	Extends the kernel recipe to apply patches
`meta-mycustomlayer/recipes-kernel/linux/files/my_fix.patch`	Contains the kernel patch
`bblayers.conf`	Adds the new custom layer
`bitbake virtual/kernel`	Rebuilds the kernel with the patch

Conclusion

- Use `.bbappend` to modify the kernel without touching the original `.bb` file.
- Place patches in a `files/` directory and reference them in `SRC_URI_append`.
- Use `bitbake virtual/kernel` to rebuild and test the kernel.
- Verify patch application using `grep` inside the build directory.

Applying Patches and Defconfig from `git.yoctoproject.org/yocto-kernel-cache` in Yocto Kernel

Overview

This guide explains how to fetch and apply kernel patches and defconfig directly from **`git.yoctoproject.org/yocto-kernel-cache`** using Yocto's `.bbappend` mechanism.

Why Use `yocto-kernel-cache`?

- Official repository maintained by the Yocto Project.
- Contains optimized patches and defconfig files for various architectures.
- Keeps the kernel aligned with upstream Yocto changes.

---

1. Locate the Required Patches and Defconfig

Browse the **Yocto Kernel Cache** repository:

```
https://git.yoctoproject.org/yocto-kernel-cache/
```

Example **patch file**:

```
https://git.yoctoproject.org/yocto-kernel-cache/plain/bsp/intel/haswell/0001-intel-haswell-fix.patch
```

Example **defconfig file**:

```
https://git.yoctoproject.org/yocto-kernel-cache/plain/bsp/intel/haswell/defconfig
```

2. Create a Custom Layer (If Not Already Created)

If a custom Yocto layer is not available, create one:

```
cd <yocto-root>
mkdir -p meta-mycustomlayer/recipes-kernel/linux/files
```

Add the new layer to `bblayers.conf`:

```
vim build/conf/bblayers.conf
```

Add:

```
BBLAYERS += "${TOPDIR}/../meta-mycustomlayer"
```

---

3. Create a `.bbappend` File for Kernel

Find the kernel version in use:

```
bitbake -e virtual/kernel | grep ^PREFERRED_VERSION
```

Example output:

```
PREFERRED_VERSION_linux-yocto="6.6"
```

Now, create the `.bbappend` file:

```
mkdir -p meta-mycustomlayer/recipes-kernel/linux
vim meta-mycustomlayer/recipes-kernel/linux/linux-yocto_6.6.bbappend
```

Add the following:

```
FILESEXTRAPATHS_prepend := "${THISDIR}/files:"
SRC_URI_append = " \

git://git.yoctoproject.org/yocto-kernel-cache;branch=master;name=yocto-kern
```

```
el-cache \
    file://0001-intel-haswell-fix.patch \
    file://defconfig \
"
SRCREV_yocto-kernel-cache = "HEAD"
SRC_URI[sha256sum] =
"abcdef1234567890abcdef1234567890abcdef1234567890abcdef1234567890"
do_configure_prepend() {
    cp ${WORKDIR}/defconfig ${S}/arch/x86/configs/haswell_defconfig
}
```

Explanation

- Fetches patches and defconfig from `yocto-kernel-cache`.
- `SRC_URI_append` adds:
  - A patch file (`0001-intel-haswell-fix.patch`).
  - A kernel defconfig.
- `do_configure_prepend()` copies the defconfig to the correct location.

---

4. Rebuild the Kernel with Patches and Defconfig

1. Clean previous builds:

```
bitbake -c clean virtual/kernel
```

2. Fetch the new patches and defconfig:

```
bitbake -c fetch virtual/kernel
```

3. Verify that files were downloaded:

```
ls -lh tmp/work/<machine>/linux-yocto/6.6*/git/
```

4. Build the patched kernel:

```
bitbake virtual/kernel
```

5. Build the full Yocto image:

```
bitbake core-image-minimal
```

---

5. Deploy and Verify

Deploy the Kernel

Verify Patch Application

Check if the patch was applied:

```
grep -r "Fixed bug in Intel Haswell kernel" /usr/src/linux/drivers/
```

Verify Defconfig

Ensure the defconfig settings are active:

```
zcat /proc/config.gz | grep CONFIG_INTEL_HASWELL
```

Summary of Changes

File	Description
`meta-mycustomlayer/recipes-kernel/linux/linux-yocto_6.6.bbappend`	Adds patches and defconfig from Yocto Kernel Cache
`https://git.yoctoproject.org/yocto-kernel-cache/plain/bsp/intel/haswell/0001-intel-haswell-fix.patch`	Kernel patch
`https://git.yoctoproject.org/yocto-kernel-cache/plain/bsp/intel/haswell/defconfig`	Kernel configuration
`bitbake virtual/kernel`	Builds the new kernel
`bitbake core-image-minimal`	Builds the final Yocto image

Conclusion

- Fetch kernel patches and defconfig from `git.yoctoproject.org/yocto-kernel-cache`.
- Use `SRC_URI_append` in `.bbappend` to apply patches and configurations.
- Use `do_configure_prepend()` to copy defconfig.
- Verify that the patches and configurations are correctly applied.