Title : Compilation of Linux kernel
Aim : Study of compilation of Linux kernel.
Objective: To study **Building Linux Kernel**
Theory:

**Introduction**

The Linux Kernel is the foundation of all the Linux distributions. The kernel is responsible for communication between hardware and software and the allocation of available resources.

All Linux distributions are based on a predefined kernel. But, if you want to disable certain options and drivers or try experimental patches, you need to compile your own Linux kernel.

**Building Linux Kernel**

The process of building a Linux kernel can be performed in seven easy steps. However, the procedure may require a significant amount of time to complete, depending on the system speed.

Following are the steps to build the latest Linux kernel-

Step 1: Download the Source Code

1. Visit the official kernel website and download the latest kernel version. The downloaded file contains a compressed source code.

2. Open the terminal and use the wget command to download the Linux kernel source code:

***wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz***

Step 2: Extract the Source Code

When the file is ready, run the tar command to extract the source code:

***tar xvf linux-6.0.7.tar.xz***

Step 3: Install Required Packages

Install additional packages before building a kernel. To do so, run this command:

***sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison***

The command we used above installs the following packages:

| Package | Package description |
|---|---|
| **git** | Tracks and makes a record of all changes during development in the source code. It also allows reverting the changes. |
| **fakeroot** | Creates the fake root environment. |
| **build-essential** | Installs development tools such as C, C++, gcc, and g++. |
| **ncurses-dev** | Provides API for the text-based terminals. |

| | |
|---|---|
| **xz-utils** | Provides fast file compression and file decompression. |
| **libssl-dev** | Supports SSL and TSL that encrypt data and make the internet connection secure. |
| **bc** (Basic Calculator) | Supports the interactive execution of statements. |
| **flex** (Fast Lexical Analyzer Generator) | Generates lexical analyzers that convert characters into tokens. |
| **libelf-dev** | Issues a shared library for managing ELF files (executable files, core dumps and object code) |
| **bison** | Converts grammar description to a C program. |

**Step 4: Configure Kernel**

The Linux kernel source code comes with the default configuration. However, you can adjust it to your needs. To do so, follow the steps below:

1. Navigate to the **linux-6.0.7** directory using the cd command:

***cd linux-6.0.7***

2. Copy the existing Linux config file using the **cp** command:

***cp -v /boot/config-$(uname -r) .config***

3. To make changes to the configuration file, run the make command:

***make menuconfig***

4. The configuration menu includes options such as firmware, file system, network, and memory settings. Use the arrows to make a selection or choose **Help** to learn more about the options. When you finish making the changes, select **Save,** and then exit the menu.

**Step 5: Build the Kernel**

1. Start building the kernel by running the following command:

***make***

The process of building and compiling the Linux kernel takes some time to complete.

The terminal lists all Linux kernel components: memory management, hardware device drivers, filesystem drivers, network drivers, and process management.

If you are compiling the kernel on Ubuntu, you may receive the following error that interrupts the building process:

***No rule to make target 'debian/canonical-certs.pem***

Disable the conflicting security certificates by executing the two commands below:

***scripts/config --disable SYSTEM_TRUSTED_KEYS***

***scripts/config --disable SYSTEM_REVOCATION_KEYS***

The commands return no output. Start the building process again with **make**, and press **Enter** repeatedly to confirm the default options for the generation of new certificates.

2. Install the required modules with this command:

***sudo make modules_install***

3. Finally, install the kernel by typing:

***sudo make install***

**Step 6: Update the Bootloader (Optional)**

The GRUB bootloader is the first program that runs when the system powers on.

The **make install** command performs this process automatically, but you can also do it manually.

1. Update the **initramfs** to the installed kernel version:

***sudo update-initramfs -c -k 6.0.7***

2. Update the **GRUB bootloader** with this command:

***sudo update-grub***

**Step 7: Reboot and Verify Kernel Version**

When you complete the steps above, reboot the machine.

When the system boots up, verify the kernel version using the uname command

***uname -mrs***