Ex. No: 13
Date: 11.5.24

DEVELOP A SIMPLE KLM

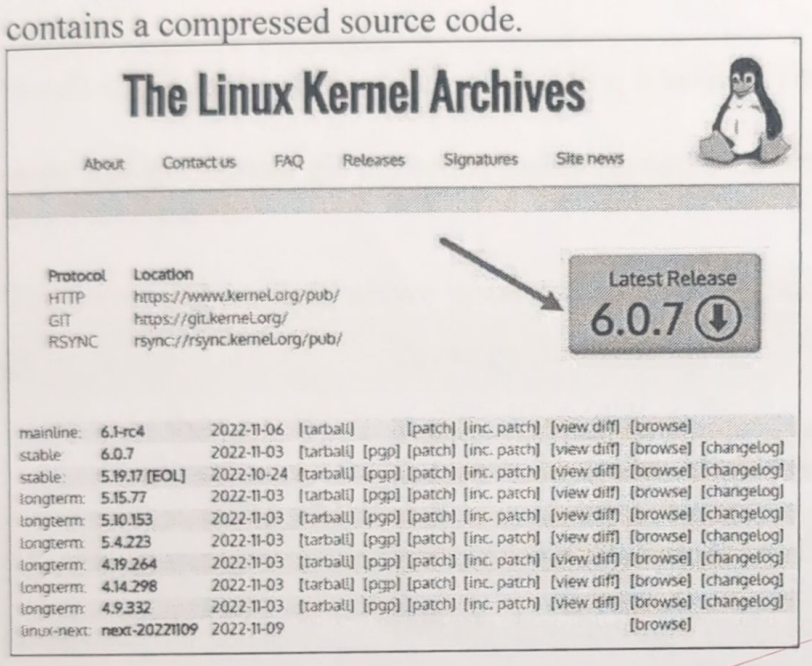
Aim:

To build a Linux Kernel from Scratch

Steps:

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

   The output shows the "saved" message when the download completes.

```
marko@pnap: $ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
--2022-11-09 17:04:51--  https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151.101.129.
176, ...
Connecting to cdn.kernel.org (cdn.kernel.org)|151.101.1.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 133884956 (128M) [application/x-xz]
Saving to: 'linux-6.0.7.tar.xz'

linux-6.0.7.tar.xz   100%[====================>] 127.68M  11.0MB/s    in 15s

2022-11-09 17:05:06 (8.35 MB/s) - 'linux-6.0.7.tar.xz' saved [133884956/133884956]

marko@pnap: $
```

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

The output displays the extracted kernel source code:

```
marko@pnap: $ tar xvf linux-6.0.7.tar.xz
linux-6.0.7/virt/kvm/irqchip.c
linux-6.0.7/virt/kvm/kvm_main.c
linux-6.0.7/virt/kvm/kvm_mm.h
linux-6.0.7/virt/kvm/pfncache.c
linux-6.0.7/virt/kvm/vfio.c
linux-6.0.7/virt/kvm/vfio.h
linux-6.0.7/virt/lib/
linux-6.0.7/virt/lib/Kconfig
linux-6.0.7/virt/lib/Makefile
linux-6.0.7/virt/lib/irqbypass.c
marko@pnap: $
```

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.

```
marko@pnap: $ sudo apt install git fakeroot build-essential ncurses-dev xz-utils libssl
-dev bc flex libelf-dev bison
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfl-dev libfl2 libsigsegv2 m4
Suggested packages:
  bison-doc flex-doc ncurses-doc libssl-doc m4-doc
The following NEW packages will be installed:
  bison flex libelf-dev libfl-dev libfl2 libncurses-dev libsigsegv2 libssl-dev m4
0 upgraded, 9 newly installed, 0 to remove and 1 not upgraded.
Need to get 4,102 kB of archives.
After this operation, 19.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] 
Setting up flex (2.6.4-8build2) ...
Setting up libfl-dev:amd64 (2.6.4-8build2) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
marko@pnap: $
```

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

```
marko@pnap:~$ cd linux-6.0.7/
marko@pnap:~/linux-6.0.7$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-52-generic' -> '.config'
marko@pnap:~/linux-6.0.7$
```

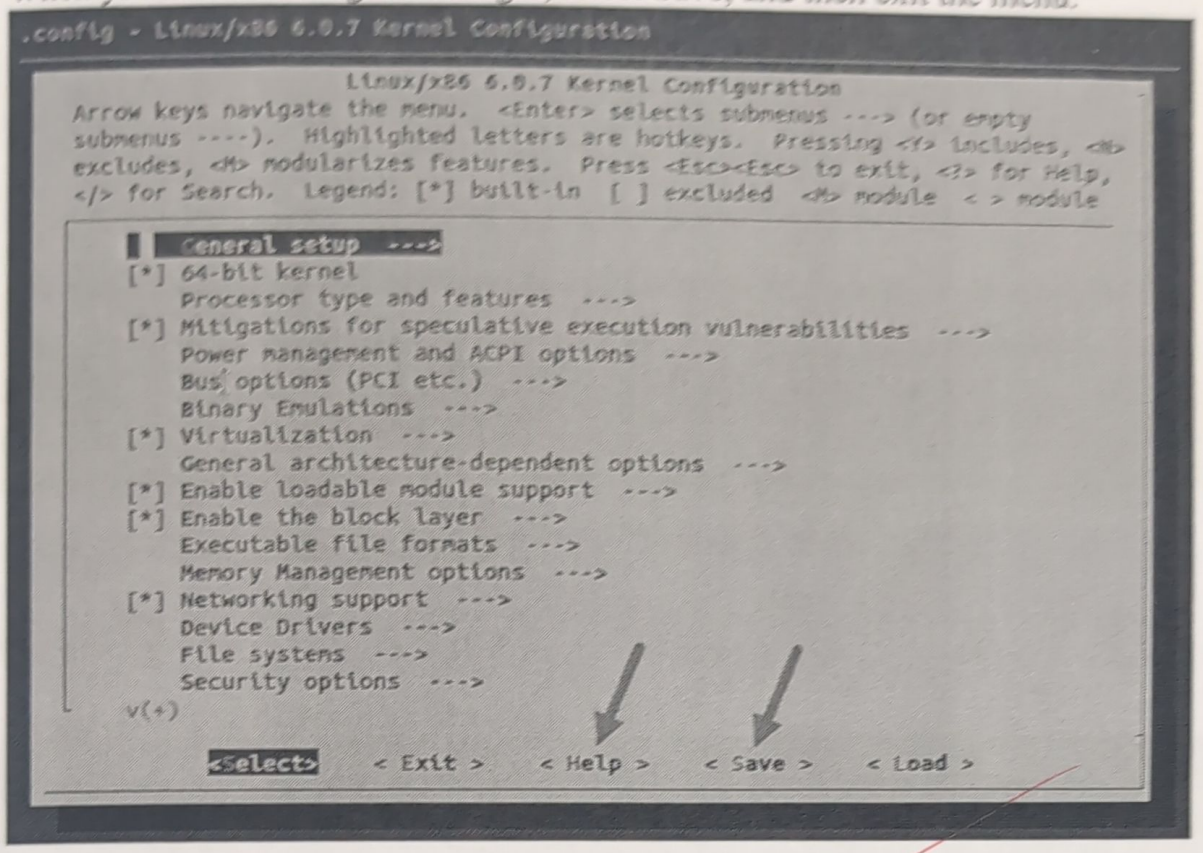
3. To make changes to the configuration file, run the make command:

make menuconfig

The command launches several scripts that open the configuration menu:

```
marko@pnap:~/linux-6.0.7$ make menuconfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
```

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.

```
marko@pnap:~/linux-6.0.7$ make
  SYNC    include/config/auto.conf.cmd
  HOSTCC  scripts/kconfig/conf.o
  HOSTLD  scripts/kconfig/conf
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_32_ia32.h
  SYSHDR  arch/x86/include/generated/asm/unistd_64_x32.h
  SYSTBL  arch/x86/include/generated/asm/syscalls_64.h
  HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
  HOSTCC  arch/x86/tools/relocs_32.o
  HOSTCC  arch/x86/tools/relocs_64.o
  HOSTCC  arch/x86/tools/relocs_common.o
  HOSTLD  arch/x86/tools/relocs
  HOSTCC  scripts/genksyms/genksyms.o
  YACC    scripts/genksyms/parse.tab.[ch]
```

2. Install the required modules with this command:

sudo make modules_install

```
marko@pnap:              $ sudo make modules_install
  INSTALL sound/usb/line6/snd-usb-line6.ko
  INSTALL sound/usb/line6/snd-usb-pod.ko
  INSTALL sound/usb/line6/snd-usb-podhd.ko
  INSTALL sound/usb/line6/snd-usb-toneport.ko
  INSTALL sound/usb/line6/snd-usb-variax.ko
  INSTALL sound/usb/misc/snd-ua101.ko
  INSTALL sound/usb/snd-usb-audio.ko
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  INSTALL sound/x86/snd-hdmi-lpe-audio.ko
  INSTALL sound/xen/snd_xen_front.ko
  DEPMOD  6.0.7
marko@pnap:              $
```

3. Finally, install the kernel by typing:

sudo make install

The output shows done when finished:

```
marko@pnap:              $ sudo make install
sh ./arch/x86/boot/install.sh 6.0.7 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/dkms 6.0.7 /boot/vmlinuz-6.0.7
 * dkms: running auto installation service for kernel 6.0.7                  [ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.0.7 /boot/vmlinuz-6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.0.7 /boot/vmlinuz-6.0.7
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
done
marko@pnap:              $
```

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

The terminal prints out the process and confirmation message:

```
marko@pnap:              $ sudo update-initramfs -c -k 6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
marko@pnap:              $ sudo update-grub
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-6.0.7
Found initrd image: /boot/initrd.img-6.0.7
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
done
```

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

The terminal prints out the current Linux kernel version.

```
marko@pnap:~$ uname -mrs
Linux 6.0.7 x86_64
marko@pnap:~$
```

RESULT:

The linux kernel has been successfully built from scratch.

11/9/24 10/10