# Compiling the linux kernel in archlinux running on a VirtualBox VM

Adam Hoover

The following instructions allow you to complete this lab on your own computer. Additional information is given in class and on the course website.

---
**Download and install VirtualBox**
---
https://www.virtualbox.org/wiki/Downloads

---
**Download archlinux iso for attaching to VM**
---
https://archlinux.org/download/

Download from any mirror, e.g. kernel.org. Download any version, e.g. 3.01_x86_64.
You should create a folder where you plan to work on VMs and put this file in that folder (archlinux-2025.03.01-x86_64.iso).

---
**Create new VirtualBox VM**
---
start VirtualBox
Machine->New
 name: arch1 (or whatever name you want)
 folder: D:\ece468\ (or wherever you want; needs 10 GB space)
 ISO image: D:\ece468\archlinux-2025.03.01-x86_64.iso
 type: Linux
 subtype: ArchLinux
 version: Arch Linux (64-bit)
[Finish]
verify settings->Storage->arch1.vdi
 hard disk file type: VDI
 file virtual size: 16 GB (twice the size of the default 8 GB)
 storage on physical hard disk: dynamically allocated

---
**Troubleshooting starting VM**
---
If you get the error "NtCreateFile(\Device\VBoxDrvStub) failed: 0xc000000034 STATUS_OBJECT_NAME_NOT_FOUND (0 retries)"
then see instructions on following link to install VirtualBox driver and start its service.
https://forums.virtualbox.org/viewtopic.php?t=66442

---

**How to install archlinux in VirtualBox VM**

---

A more detailed explanation of the following instructions is available here:
https://freedium.cfd/https://medium.com/code-art/virtualbox-complete-guide-to-install-archlinux-on-virtual-machine-338aca8a5000

Start the VM. Select this option from the boot loader:

| boot "Arch Linux install mediaum (x86_64, BIOS)" |
|---|

Once the VM is booted and you see a shell, type all the following commands:

| Command | Explanation |
|---|---|
| fdisk -l | list partition tables |
| fdisk /dev/sda | opens HD to work on partition |
| m | shows fdisk commands |
| l | shows fdisk partition types |
| g | create new empty GPT partition table |
| n | add a new partition |
| [ENTER] | accept default value for partition number (1) |
| [ENTER] | accept default value for first sector |
| +1M | set size of partition to 1 MB |
| t | change partition type (should show partition (1)) |
| 4 | change partition type to "BIOS boot" |
| n | add a new partition |
| [ENTER] | accept default value for partition number (2) |
| [ENTER] | accept default value for first sector |
| +12G | set size of partition to 12 GB |
| t | change partition type |
| [ENTER] | select last partition (2) |
| 23 | change partition type to "Linux root (x86-64)" |
| n | add a new partition |
| [ENTER] | accept default value for partition number (3) |
| [ENTER] | accept default value for first sector |
| +1G | set size of partition to 1 GB |
| t | change partition type |
| [ENTER] | select last partition (3) |
| 19 | change partition type to "Linux swap" |
| w | write partition table |

| Command | Explanation |
|---|---|
| mkfs.ext4 /dev/sda2 | format "Linux root" partition to ext4 file system |
| mkswap /dev/sda3 | initialize "Linux swap" partition |
| mount /dev/sda2 /mnt | mount "Linux root" to /mnt |
| swapon /dev/sda3 | enable "Linux swap" |
| fdisk –l | print out partition table to see it |

Set up pacman for the latest package distribution links.

| Command | Explanation |
|---|---|
| pacman -Sy | sync master package database |
| pacman -S archlinux-keyring | install signing keys for packages |

Next, install a bunch of packages. You can customize this list if you want to include more tools.

| |
|---|
| pacstrap /mnt base linux linux-firmware grub dhcpcd openssh vim nano which net-tools |

The following commands set up the filesystem, timezone, and boot loader. For the commands that edit files, I suggest using nano instead of vim if you are unfamiliar with vim. There are many tutorials and quick references on the internet for both.

| Command | Explanation |
|---|---|
| genfstab -U /mnt >> /mnt/etc/fstab | |
| arch-chroot /mnt | Enter chroot envir. |
| ln -sf /usr/share/zoneinfo/America/New_York /etc/localtime | |
| hwclock --systohc | |
| vim /etc/locale.gen | Edit using vim or nano. |
| (uncomment by deleting #) en_US.UTF-8 UTF-8 | This is the only change made to file; save after editing. |
| locale-gen | |
| vim /etc/locale.conf | |
| (add) LANG=en_US.UTF-8 | This is the only content of this file; save after editing. |
| vim /etc/hostname | |

| | |
|---|---|
| (add) varch | This will be the name of your machine; use whatever name you prefer. Save after editing. |
| passwd | |
| (change to) archlinux (or desired password) | This will be your root password; use whatever you prefer. |
| grub-install --target=i386-pc /dev/sda | |
| grub-mkconfig -o /boot/grub/grub.cfg | |
| exit | Exits arch-chroot |
| umount -R /mnt | Unmounts all partitions |
| reboot | |

During this reboot, select the following options:

| |
|---|
| select "Boot existing OS" |
| select "Arch Linux" (after adding another kernel, "Arch Linux, with Linux linux") |
| login as "root", password "archlinux" or whatever you picked |

The following commands are optional. They customize the way your shell operates. If you want to know more, search the internet for customizing bash.

| | |
|---|---|
| vim ~/.bash_profile | Optional: edit bash profile to customize. Add these lines or your preferences. |
| alias ls='ls -F' | Makes it easier to see folders. |
| alias rm='rm -i' | Prompts to confirm file removal. |
| alias mv='mv -i' | Prompts to confirm file moving. |
| alias cp='cp -i' | Prompts to confirm copy overwriting. |
| alias vi='vim -u NONE' | Runs vim as more traditional vi. |
| PS1='\w> ' | Customized prompt. |

If you edited your .bash_profile file, save it, and then run the following shell command.

| | |
|---|---|
| source ~/.bash_profile | Command updates shell preferences |

Initialize networking, and configure the machine to it always starts networking on reboot.

| Command | Explanation |
|---|---|
| ip l | list network interfaces; look for enp0s3 or similar for next command |
| dhcpcd enp0s3 | start network connection |
| systemctl enable dhcpcd | start network automatically on reboot |

Optionally, install any other packages/commands you like to use. For example.

```
pacman -S which tcsh
```

---
**kernel compilation in archlinux**
---
The following link provides a deeper explanation of the list of commands below.
https://wiki.archlinux.org/title/Kernel/Traditional_compilation

| Command | Explanation |
|---|---|
| pacman -S base-devel | installs gcc and other base tools |
| pacman -S xmlto kmod inetutils bc | install more tools |
| pacman -S libelf git cpio perl tar xz | install more tools |
| pacman -S wget | install wget |

Check https://www.kernel.org/ for the latest kernel. At the time of writing it is 6.14. You can use a different version if you prefer, but will need to update all the below commands as needed, starting with downloading the source code:

| wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.14.tar.xz |
|---|

Next, unpack the source code and compile it. There are many ways to set up the .config file; below I demonstrate using menuconfig. When the menuconfig screen comes up, make the one change indicated below, then save and exit to create to the .config file.

| Command | Explanation |
|---|---|
| tar -xvf linux-6.14.tar.gz | Unpack source code |
| cd linux-6.14 | This is the source code folder |
| make mrproper | Clean up any previous builds |
| make menuconfig | Create the .config file |
| ->general setup, turn off "Compile the kernel with warnings as errors" | Warnings will halt the compile otherwise |
| time make | Build the kernel (and time it). This takes 20-60 minutes. |

---
**set up kernel, module and ramdisk files**
---

The following commands create and/or position the newly compiled kernel and support files into places needed for booting it.

| Command | Explanation |
|---|---|
| cd ~/linux-6.14 | directory holding kernel source |

| | |
|---|---|
| make modules | creates drivers for this kernel |
| make modules_install | copies them to /lib/modules/… |
| cp arch/x86/boot/bzImage /boot/vmlinuz-linux6.14 | copy the kernel file to /boot and rename it to unique filename |
| cd /etc/mkinitcpio.d/ | directory holding ramdisk info |
| cp linux.preset linux6.14.preset | copy config file for new kernel |
| vim linux6.14.preset | edit the following fields |
| ALL_kver="/boot/vmlinuz-linux6.14" | |
| default_image="/boot/initramfs-linux6.14.img" | |
| fallback_image="/boot/initramfs-linux6.14-fallback.img" | |
| mkinitcpio -p linux6.14 | create the new ramdisk file |

---

**make new grub bootloader**

---

The following commands edit the boot loader and configure it to add the new kernel as a boot option.

| Command | Explanation |
|---|---|
| cd /etc/default | directory holding grub settings |
| vim grub | edit "grub" file and make these changes |
| uncomment "GRUB_TERMINAL_OUTPUT" | |
| uncomment "GRUB_DISABLE_SUBMENU" | |
| cd /boot/grub | directory holding bootloader files |
| cp grub.cfg grub.cfg.orig | make backup - always a good idea |
| grub-mkconfig -o grub.cfg | make new bootloader |
| reboot | |

During this reboot, select the following options:

| |
|---|
| select "Boot existing OS" |
| select "Arch Linux, with Linux linux6.14” |
| login as "root", password "archlinux" or whatever you picked |

Type the following commands to verify everything:

| | |
|---|---|
| uname -a | verify it shows 6.14 kernel |
| ls -al /boot | verify can see two different sized vmlinuz files |

---
**add a function to the kernel**
---
Add an entry to the system call table. The following instructions assume your current directory is wherever you placed your kernel source (e.g. ~/linux-6.14).

| | |
|---|---|
| cd ~/linux-6.14 | directory holding kernel source |
| vim arch/x86/entry/syscalls/syscall_64.tbl | edit the syscalls table file |

Find an empty slot and add a new entry as shown below. For linux-6.14, the value 467 is unused, but you may need to change it for a different kernel.

```
467     common      clemson                 sys_clemson
```

Add the following custom function call to the fs/sync.c kernel source file. We could create a new source code file to hold it, but then we would have to edit the Makefile, so this is simplest.

| | |
|---|---|
| vim fs/sync.c | edit a kernel source code file |

Add this code at the bottom of the file.

```
SYSCALL_DEFINE1(clemson, int, number)
{
printk(KERN_INFO "Hello %d Tigers!", number);
return(1);
}
```

Now compile and install the kernel using the same instructions as above, but with the new kernel filename linux6.14.b. Step-by-step instructions are below; see above for more complete descriptions.

| | |
|---|---|
| cd ~/linux-6.14 | directory holding kernel source |
| time make | Build the kernel (and time it). This takes 20-60 minutes. |
| make modules | creates drivers for this kernel |
| make modules_install | copies them to /lib/modules/… |
| cp arch/x86/boot/bzImage /boot/vmlinuz-linux6.14b | copy the kernel file to /boot and rename it to unique filename |
| cd /etc/mkinitcpio.d/ | directory holding ramdisk info |
| cp linux.preset linux6.14b.preset | copy config file for new kernel |

| | |
|---|---|
| vim linux6.14b.preset | edit the following fields |
| ALL_kver="/boot/vmlinuz-linux6.14b" | |
| default_image="/boot/initramfs-linux6.14b.img" | |
| fallback_image="/boot/initramfs-linux6.14b-fallback.img" | |
| mkinitcpio -p linux6.14b | create the new ramdisk file |

---
**make new grub bootloader**
---

| Command | Explanation |
|---|---|
| cd /boot/grub | directory holding bootloader files |
| cp grub.cfg grub.cfg.orig | make backup - always a good idea |
| grub-mkconfig -o grub.cfg | make new bootloader |
| reboot | |

During this reboot, select the following options:

| |
|---|
| select "Boot existing OS" |
| select "Arch Linux, with Linux linux6.14b" |
| login as "root", password "archlinux" or whatever you picked |

Type the following commands to verify everything:

| | |
|---|---|
| uname -a | verify it shows 6.14b kernel |
| ls -al /boot | verify can see three different sized vmlinuz files |

---
**test your system call**
---

Write a C program using the following code.

```
#include <stdio.h>
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
long int ret;

ret=syscall(467,3);          //syscall 467 with argument 3
printf("System call sys_clemson returned %ld\n",ret);
return(1);
}
```

Compile it, and run it.  You should see the following output.

```
> gcc -o testclemson testclemson.c
> ./testclemson
System call sys_clemson returned 1
> dmesg | tail -1
Hello 3 Tigers!
>
```