Lab 2: Working with Editors and X Setup

Benedict Reuschling

February 19, 2019

After we installed and configured a basic Unix system, we will now make use of it. Knowing how to use a text editor is vital for success in this lab, as we are doing a lot of file editing.

1 Preparation

Familiarize yourself with the basics of the vi/vim text editor, based on the lecture slides. You should be able to perform the following editing functions with relative ease:

- Opening a file
- Changing between different modes (i.e. from command into insert mode)
- Doing basic text manipulation (inserting text, copying a line and putting it in a different location within the same file)
- Saving[1] and exiting the editor

If you struggle with using vi/vim, let me know. A good way to learn vim is by running `vimtutor`, which teaches you how to use vim. Make sure `vim` is installed, otherwise the command will not be available on your system. Another fun way of learning vim is by visiting `https://vim-adventures.com/`.

Also, make sure that you can copy and rename files, as well as creating directories and changing into or out of them. Review the lecture slides that give an overview of Unix to refresh your memory if needed.

2 Basic System Configuration

As we are going to start programming in this lab, we will prepare our working environment for more editing convenience. First, we install `vim` or `vim-lite` (a more lightweight version with less dependencies). Log in as the `root` user and install `vim` using the package system as we did in previous labs. Next, refer to slide 21/23 of the editors chapter of the lecture and set some of the editor options to get line numbers, syntax highlighting, etc. Make sure to set these for your non-root user account, too.

Next, we'll provide our non-root user with the ability to run certain commands with elevated privileges. That way, we don't have to switch back and forth between the normal user and `root` to perform certain operations that are normally reserved for `root`. Install

[1] make sure that you are allowed to write the file, some files can only be edited by `root`

the package `sudo`. While still logged into the root account, run the program `visudo`. This will start a vi editor session in the `sudoers` file. That file determines the permissions to be given to users and certain groups. Since we put our unprivileged user into the `wheel` group in lab 1, we can now give members of that group the permission to run commands using `sudo`. To do that, scroll down and find the line that looks like this:

```
# %wheel ALL=(ALL) ALL
```

Remove the comment character (`#`) in front of it, save, and close the editor. Log back into your regular user account. To see your sudo rights, use `sudo -l`, and after entering your regular user password, your permissions on this machine are listed (ALL). Next, run `sudo pkg update`. That operation normally requires system administration privileges. But since we added the `wheel` group to the allowed groups that may perform this operation, the update should run without any errors, just like if `root` would execute it.

You can now use `sudoedit` to make changes to configuration files that normally only `root` can edit. The environment variable `EDITOR` will determine which editor to run. To test it, add the following line to `/boot/loader.conf`:

```
autoboot_delay="2"
```

This will shorten the countdown time on the boot menu from 10 seconds to 2 seconds. Reboot your virtual machine to see that the changes took effect.

Note that when you activate and reboot into another boot environment (using `bectl`), this change will not be there as it happened in a different environment. That way, when you make errors in configuration files, you can still get your system back by booting into the other boot environment, correct the mistake, create and activate a new boot environment and boot into it. If you have a state that you are satisfied with, you can create another boot environment using `bectl create` together with `sudo`.

3 Setting up Xorg

in this section, we will set up the graphical Xorg environment to use tools that require it. Follow these steps in your VirtualBox VM:

1. First, we install the VirtualBox guest additions and drm-kmod (abbr. DRM kernel module), which will provide support for the VirtualBox graphics adapter[2] (among other things). Make sure that you are connected to the Internet to download the packages. Install the following package, either as the `root` user or with `sudo` as your normal user:

   ```
   pkg install virtualbox-ose-additions drm-kmod
   ```

   Once the installation has finished, it tells us to make a couple of additions to `/etc/rc.conf`. Two services need to be enabled: `vboxguest` and `vboxservice`. We can use `sysrc` to add these lines to `/etc/rc.conf` (check the sysrc man page for more information about this utility). Also, we need to list the modules to load for the graphics. In our case, it's `i915kms.ko` located in `/boot/modules`.

   ```
   sysrc vboxguest_enable=yes
   sysrc vboxservice_enable=yes
   sysrc kld_list=/boot/modules/i915kms.ko
   ```

   Check that each line was added to `/etc/rc.conf` using `cat` and when they are present, `reboot` the machine to make these changes take effect.

2. After the reboot, we install the xorg package. The following commands need to be run as the `root` user.

   ```
   # pkg install xorg
   ```

   When prompted, confirm with `yes` or `y`.

3. Add your user to the `video` group to be able to acess the video card using this command:

   ```
   # pw groupmod video -m <yournonrootuser>
   ```

4. Next, we start the X system to detect the new devices like screen, keyboard, and mouse that we now have available in the virtual machine. We will use Xorg's detection capabilities to create a config file that contains the important sections containing the detected devices.

   ```
   # startx
   ```

   This will start a very basic desktop environment (black screen). Hit Ctrl + Alt + F1 to get back to the shell and then Ctrl + C to stop the Xorg program.

5. A desktop environment provides window management functionality and many other features. We will install a lightweight desktop[3] for our virtual machine called fluxbox. We also install dbus as we need this later to run a browser:

[2] Remember to set the graphics controller setting `VMSVGA` from Lab 1

[3] Refer to `https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/x11-wm.html` to install heavy-weight and feature-rich desktops like KDE and Gnome. They take a lot of disk space and are probably too much for our purposes. Hence, we leave that exercise to you.

```
# pkg install fluxbox dbus
```

6. Switch back to your non-root user now. Create a new file called `.xinitrc` in your home directory (located at `/usr/home/<username>` or just `~`), which will be used to tell the X server which desktop to start and what other settings it should activate prior to loading the desktop. We add the following two lines (the first one is optional) to set the keyboard layout to german[4] and the window manager:

   ```
   setxkbmap -model pc105 -layout de
   /usr/local/bin/fluxbox
   ```

   We can use `startx` as our normal (non-root) user to start the fluxbox desktop now. Use the right mouse button to call up a little start-menu, hover over fluxbox menu, and then click on Exit.

7. We will finish our X configuration by adding the D message bus and hardware abstraction layer services that are required for browsers like Firefox or Chromium. To do so, we use sysrc again to safely add values to our configuration file `/etc/rc.conf`:

   ```
   # sysrc dbus_enable=yes
   # sysrc hal_enable=yes
   ```

   Check `rc.conf` to see how the change was applied. Next, we need to start the DBUS service for the current session (next time we reboot, the service will be started automatically).

   ```
   # service dbus start
   ```

   Now we only have to install `firefox` or `chromium` via `pkg install` to have either one of these browsers available. Once they are installed, regenerate the fluxbox menu by running:

   ```
   $ fluxbox-generate_menu
   ```

   Check the new entries in the fluxbox menu (right click on the desktop). At least firefox or chromium should be added to the menu, depending on which you've installed.

That's it, we now have a graphical desktop environment available which we can work in. Additional software can be installed to make the desktop more appealing, but for our purposes, what we have should be enough.

[4] Only if you have a QWERTZ keyboard, otherwise leave it alone