„Introduction guide”

Arkadiusz Chrobot, PhD
Karol Tomaszewski, PhD

February 24, 2024

Contents

Introduction

These introductory guide contains brief descriptions of the tools whose knowledge will be necessary for the efficient implementation of tasks within the laboratories of Operating Systems 2. Each team will be assigned a virtual machine that will need to be operated from the console, therefore it is necessary to remember the basic commands of the system shell. These commands will not be described in these guides. Section 1 describes how to use the `ssh` (*Secure Shell*) command, Section 2 presents the `scp` (*Secure Copy*) command, and Section 3 refers to the use of the Linux Cross Reference service.

1. ssh

The `ssh` command is used for working with the remote computer. This work is done using a system shell (console), and the network traffic is encrypted. In case of laboratory classes, the remote computer is a virtual machine. To connect to it, issue the following command in a local computer terminal or console:

```
ssh so2@81.26.20.32 -p X
```

The `so2` in the command is a username, `81.26.20.32` is an IP address of virtual machine server, and `X` is a port number. Every team will have a separate virtual machine assigned. The port numbers used by the machines will be given by the person conducting the laboratory classes. When the command is issued, the remote system will ask for the password. Enter the password provided by the person conducting the laboratory. While typing the password, no characters will appear on the screen until the `Enter` button is pressed. Please note, that You can connect to the virtual machine only from within the Department of Information Systems internal network, provided the machine is on.

The command allows You to log as the `so2` user. This account has limited privileges. To load or unload a kernel module, what is essential to accomplish tasks from laboratory guides, You need to log to the `root` account, using the following command:

```
su -
```

The system will ask for the `root` user password, which also will be given by the person conducting the laboratory. If the password is valid, then You will be granted access to an account that allows You to administrative operations, including those mentioned earlier.

It is also possible to log in as the `root` using another command:

```
sudo su -
```

This time, the system will ask for the `so2` user password, not the `root` one. If it is valid, You will also be granted access to the `root` account.

The `sudo` command has more applications. Generally, it is uses to perform privileged commands by an unprivileged user. However, these cannot be any commands. They should have been previously specified for that user by the `root`.

2. scp

The `scp` command belongs to the same package as `ssh`. It allows the user to copy files between a local and a remote computer. Copying of the file named `source.c` located in the current directory on the local machine, to the home directory of the `so2` user on the remote virtual machine on the X port, can be done using the following command:

```
scp -P X source.c so2@81.26.20.32:~
```

Please note the „ : ” character after the IP address followed by the directory name. In this case, it is the user's home directory, which in Linux is denoted with a tilde character (~). Should the destination directory be `lab1`, located inside the home directory, then the command would have to be:

```
scp -P X source.c so2@81.26.20.32:~/lab1
```

The remote system will ask for the `so2` user password, just like in the case of `ssh`.

Similarly, files can be copied in the opposite direction. For example, to copy a file named `Makefile` from the home directory of the `so2` user at the virtual machine, to the local computer current directory, issue the following command:

```
scp -P X so2@81.26.20.32:~/Makefile .
```

The dot at the end of the command denotes the current directory. Please note, that the order of `scp` arguments is the same, as in the case of the `cp` command, used for copying files localy.

There are two ways to copy the entire directory to a remote system. Suppose this directory is called `sources` and is a subdirectory of the current directory. The first way is to use the `-r` option of the `scp` command, which recursively copies the directory:

```
scp -P X -r sources so2@81.26.20.32:~
```

In the second method, the directory should be first compressed, e.g. with the `tar` command. It can be done as follows:

```
tar jcvf sources.tar.bz2 sources,
```

where `sources.tar.bz2` is the name of the output file being a compressed archive of the `sources` directory. After copying it to a remote computer, its contents can be extracted with the command:

```
tar xvf sources.tar.bz2
```

3. Linux Cross Reference

There is documentation for the Linux kernel source code in the form of `man` pages, but unfortunately it is very limited. The best way to investigate what a piece of code does and how it is built is to browse kernel sources, which are very extensive. Fortunately, there is a tool that can generate HTML pages that allow to easily navigate this code. The pages created this way are available, among others, at `https://elixir.bootlin.com/linux/latest/source`. The following example describes how to use this site, searching for information about the `schedule()` function[1]. After visiting the given URL address, the page should appear like in Figure 1.

On this page, the version `4.15` of the kernel sources should be selected, as shown in Figures 2 and 3.

After selecting the kernel source code version, enter the name of function: *schedule* in the *Search Identifier* field, then click the search icon or press `Enter` (Figure 4).

The results page is divided into two parts. The first part, marked in yellow in Figure 5, contains references to the lines in the files in which the prototype or definition of the function is located. The second part, whose fragment is marked in green, is a list of links to the file lines in which this function is used.

After clicking the first link in the *Defined in 3 files:* section, the user will be taken to the page with the function declaration. Its prototype (header) is marked in yellow in Figure 6.

After clicking the second link in the aforementioned section, the user will be taken to the page with the definition of this function, which is marked in yellow in Figure 7.

[1] This function is an implementation of the processor scheduler.

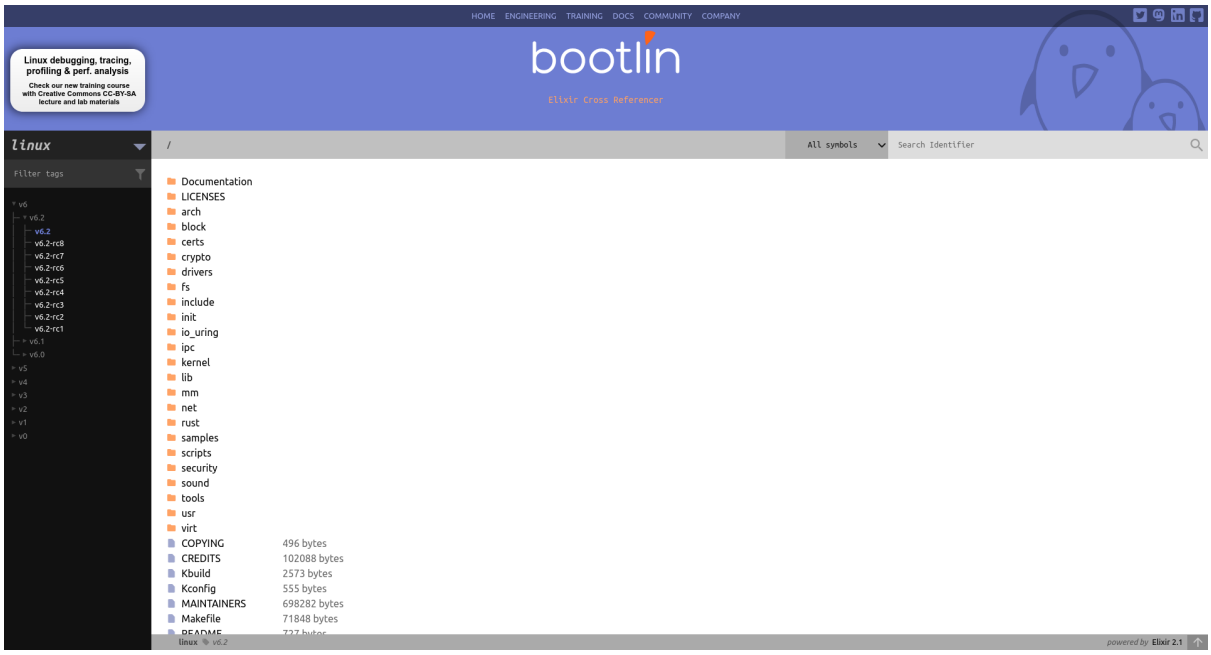

Figure 1: Main page of the Linux Cross Reference service

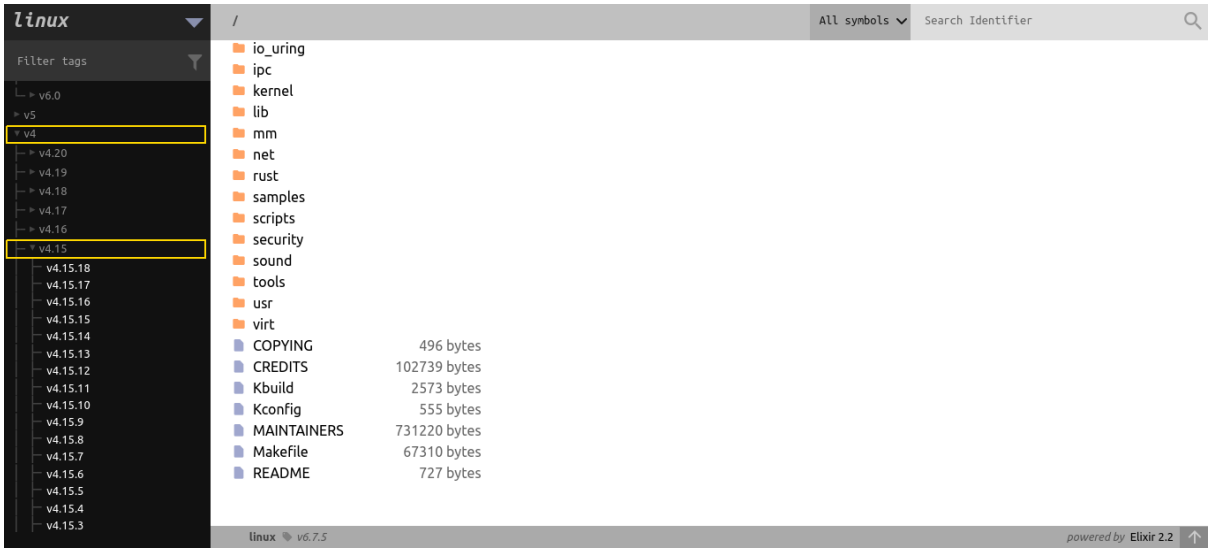

Figure 2: Kernel source version selection menu

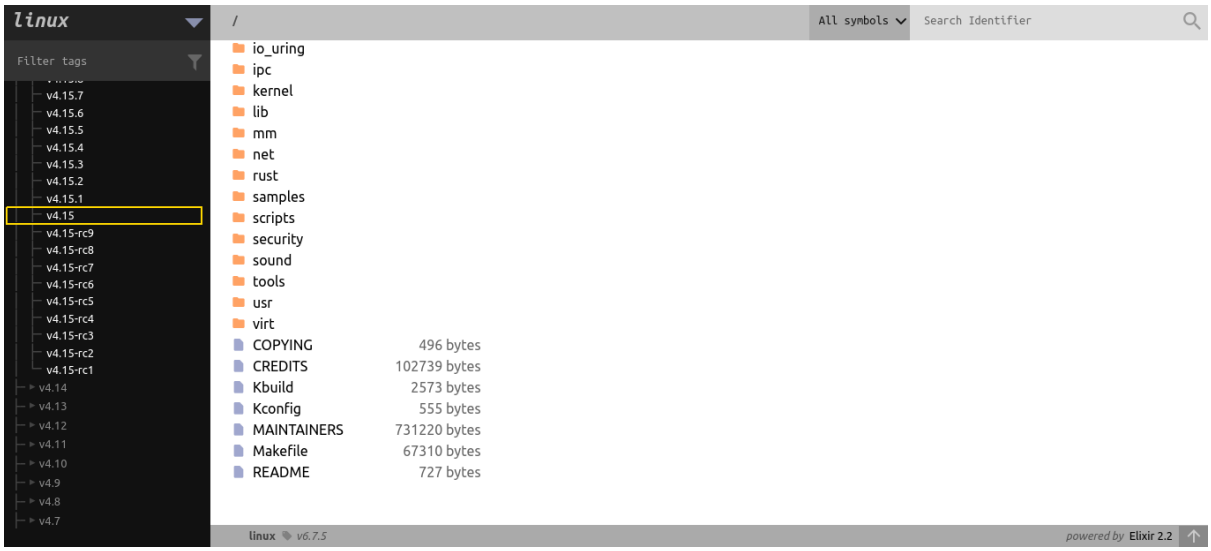

Figure 3: Detailed menu for choosing the kernel source version

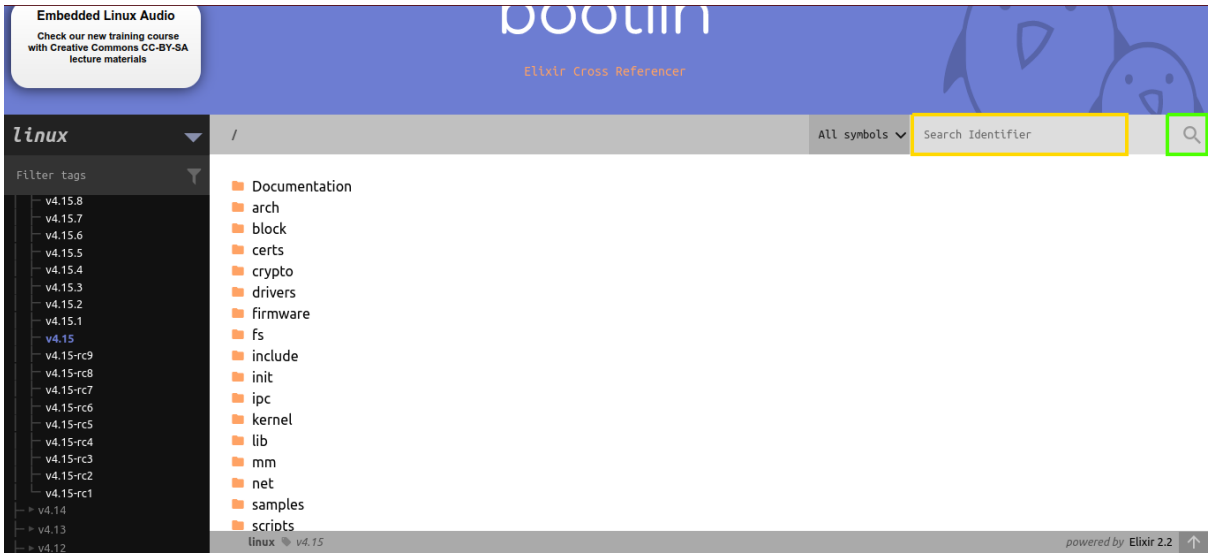

Figure 4: Name search

Figure 5: Results of the name search

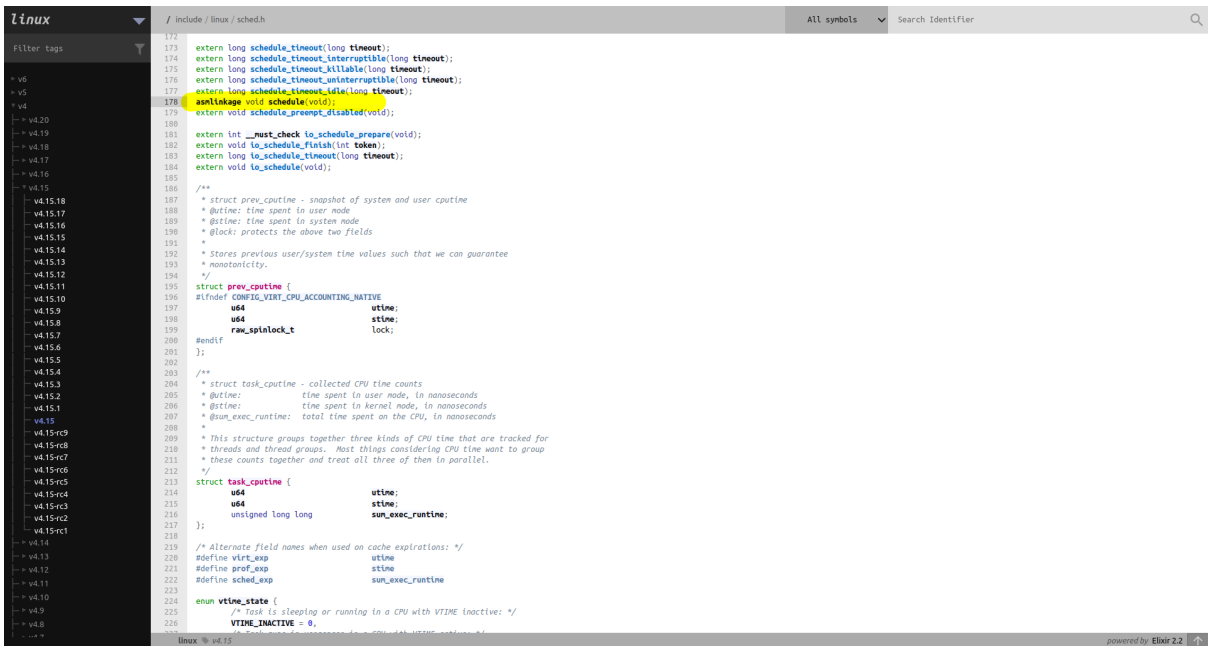

Figure 6: Declaration of the `schedule()` function

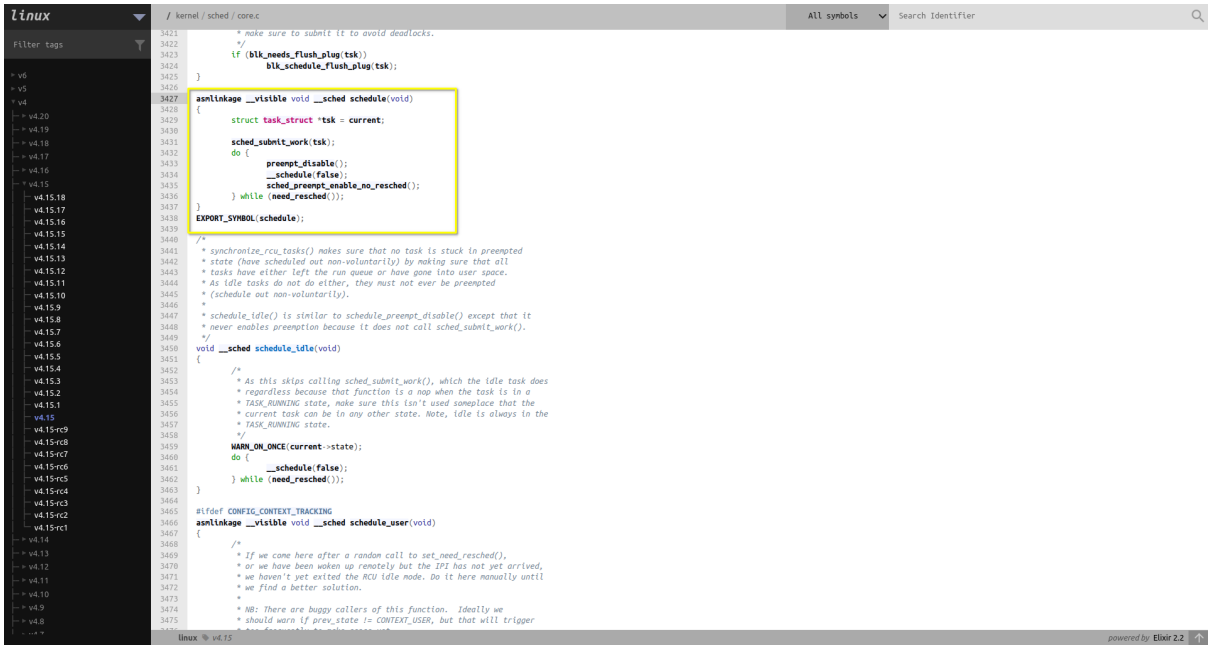

Figure 7: Definition of the `schedule()` function