Basic Linux Commands

Contents

Introduction

This article will go over basic Linux commands that are commonly used with the AXC F 2152.

NOTE: This document will only contain basic commands and will **not** contain all possible Linux commands. To learn more Linux commands, there are multiple tutorials available online.

Objectives

- Navigate the file system
- Create a file
- Edit a file
- Change permissions of a file
- Run a script file (.sh file extension)

Basic Commands

This section will go each of the basic commands, what they mean and how to use them.

1 – Navigating the file system

The filesystem works just like the filesystem on your computer, but navigating directories is done by commands rather than clicking.

<u>1.1 – Your current location in the filesystem:</u> The current location you are in the filesystem is the prefix to the command line input which is circled in Figure 1.1-1.

NOTE: If you are logged in as admin, the /opt/plcnext/ directory will show as the / directory!

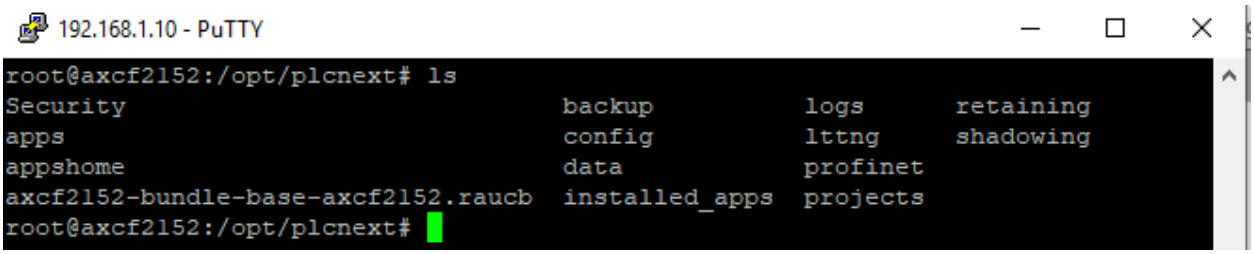
If you do not know how to create/login to a root user refer to Appendix 4.

Figure 1.1-1 Showing the current directory you are located as root

<u>1.2 – ls :</u> List command. This command will list all of the files in the current directory. This command is very useful when you do not know what files exist in the current directory.

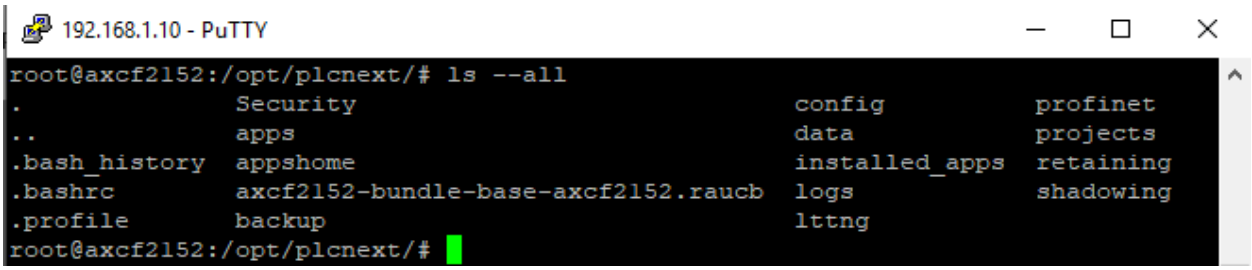
1) To view files in the current directory type the command "`ls`" as shown in figure 1.1.2

Figure 1.1-2 ls command output



2) To view all files (including hidden files) type the command `ls --all` as shown in figure 1.1-3

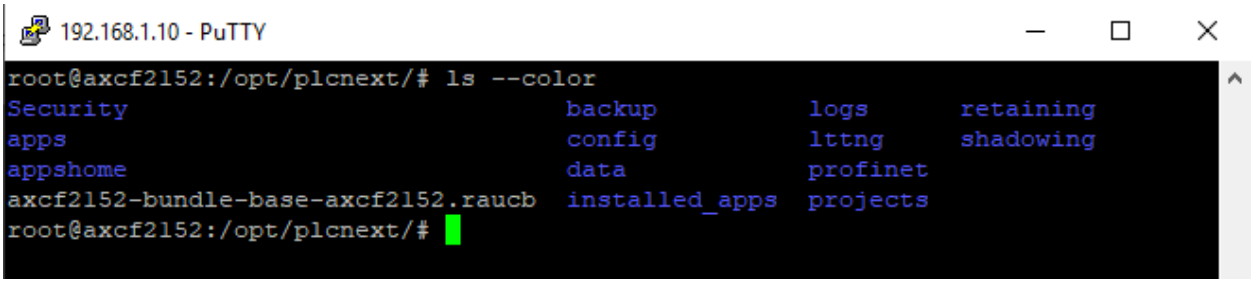
Figure 1.1-3 ls --all command output



3) To view all files in color which explains the rights for each file, type `ls --color` as shown in figure 1.1-4.
   The color code is as follows:

   - **Uncolored (white)**: file *or* non-filename text
   - **Bold blue**: directory
   - **Bold cyan**: symbolic link
   - **Bold green**: executable file
   - **Bold red**: archive file
   - **Bold magenta**: image file, video, graphic, etc. *or* door *or* socket
   - **Cyan**: audio file
   - **Yellow with black background**: pipe (AKA FIFO)
   - **Bold yellow with black background**: block device *or* character device
   - **Bold red with black background**: orphan symlink *or* missing file
   - **Uncolored with red background**: set-user-ID file
   - **Black with yellow background**: set-group-ID file
   - **Black with red background**: file with capability
   - **White with blue background**: sticky directory
   - **Blue with green background**: other-writable directory
   - **Black with green background**: sticky and other-writable directory

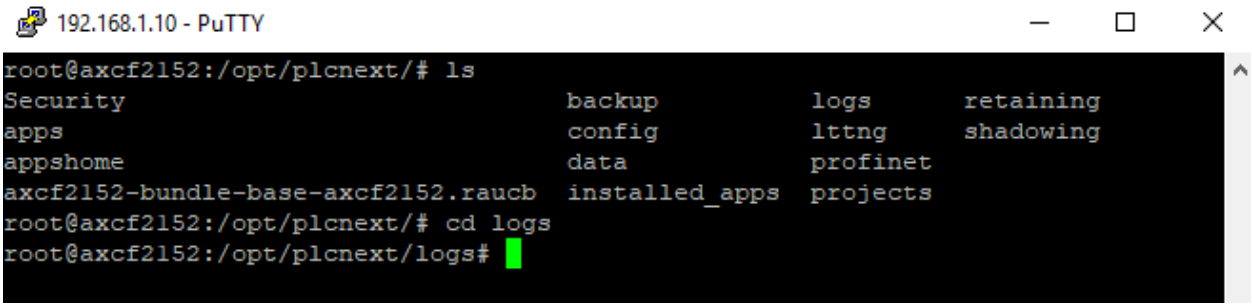
Figure 1.1-4 ls --color command output



1.3 – cd : Change directory. This command allows the user to change to a different directory of the filesystem. On the AXC F 2152 this is the most used command. Examples are below:

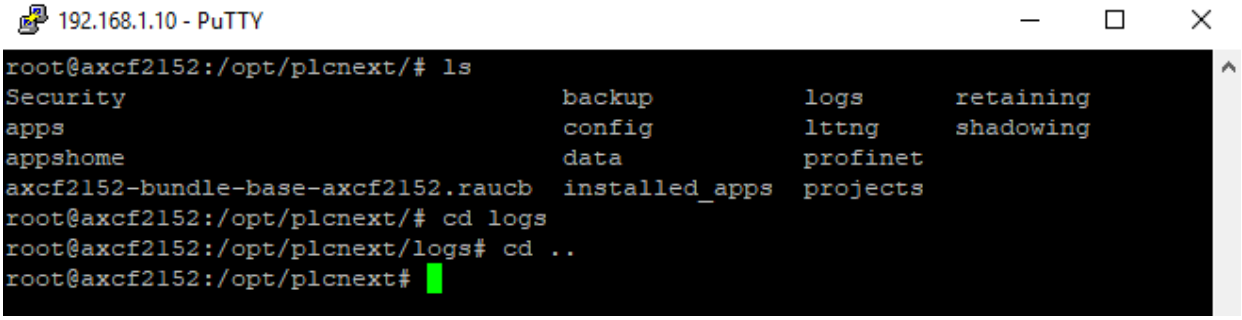
1) To change to a directory that is listed when running the ls command, type cd (directory name) to change to that directory. See figure 1.1-5.

Figure 1.1-5 Changing to a directory located in current directory



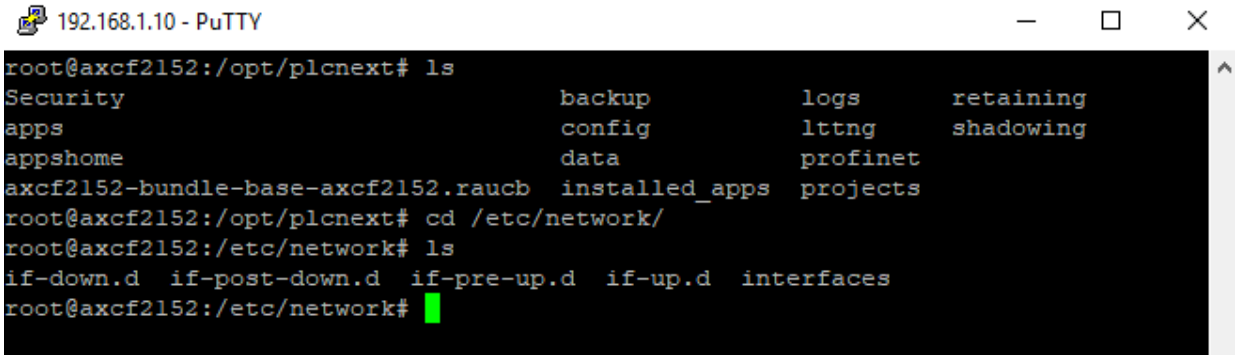
2) To change to a directory above the current directory, type "`cd ..`" as shown in figure 1.1-6

Figure 1.1-6 Going up a directory from the current directory



3) To change to a directory that is not listed when running the ls command, type `cd full_path_to_location` to change to that directory. See figure 1.1-6.

Figure 1.1-6 Changing to a directory using absolute path



1.4 – clear: Clears the command line, as shown in figure 1.4-1.

Figure 1.4-1 clear command before and after

```
192.168.1.10 - PuTTY
root@axcf2152:/opt/plcnext/# ls
Security                              backup          logs       retaining
apps                                  config          lttng      shadowing
appshome                              data            profinet
axcf2152-bundle-base-axcf2152.raucb   installed_apps  projects
root@axcf2152:/opt/plcnext/# cd logs/
root@axcf2152:/opt/plcnext/logs# cd ..
root@axcf2152:/opt/plcnext# clear
```

```
192.168.1.10 - PuTTY
root@axcf2152:/opt/plcnext#
```

2 – Creating a file

This section will go over how to create a file in PuTTY.

2.1 – touch: The touch command will create a file in the filesystem with the name and extension of your choice. Refer to figure 2.1-1 for examples.

Figure 2.1-1 creating files with touch

```
192.168.1.10 - PuTTY
root@axcf2152:/opt/plcnext# touch textfile.txt
root@axcf2152:/opt/plcnext# touch table.csv
root@axcf2152:/opt/plcnext# touch script.sh
root@axcf2152:/opt/plcnext# ls --color
Security                              config          profinet    table.csv
apps                                  data            projects    textfile.txt
appshome                              installed_apps  retaining
axcf2152-bundle-base-axcf2152.raucb   logs            script.sh
backup                                lttng           shadowing
root@axcf2152:/opt/plcnext#
```

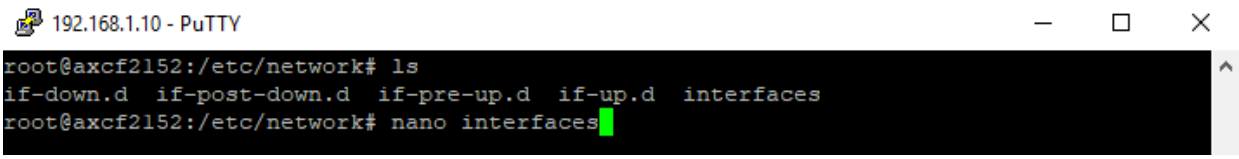
3 – Editing a file

This section will go over how to use nano to edit a file. Vim is also available to edit files, but we will be showing nano.

<u>3.1 – nano:</u> nano is used to open a file a view the contents, as well as make changes to the file.

1) Opening a file: To open a file, type `nano file_name_here`. See figure 3.1-1

Figure 3.1-1 opening a file with nano



2) Understanding the environment: Once nano has opened a file, it displays file into 3 separate areas as shown in figure 3.1-2:

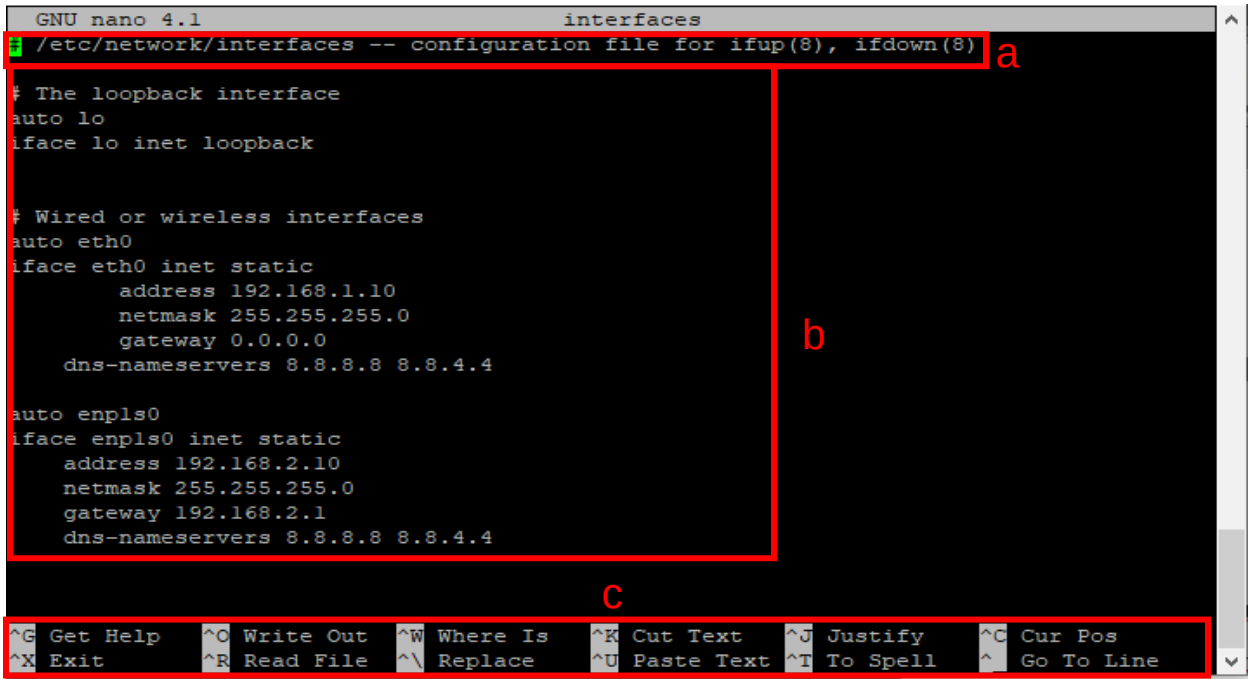
a) Description. Shows what file is being edited as well as a description if it applies.

b) Contents of the file

c) Editor options. The "^" symbol refers to the control key. For example, to exit press control and X at the same time.

NOTE: To save the changes to the file, press control and S at the same time.

Figure 3.1-2 nano editor space

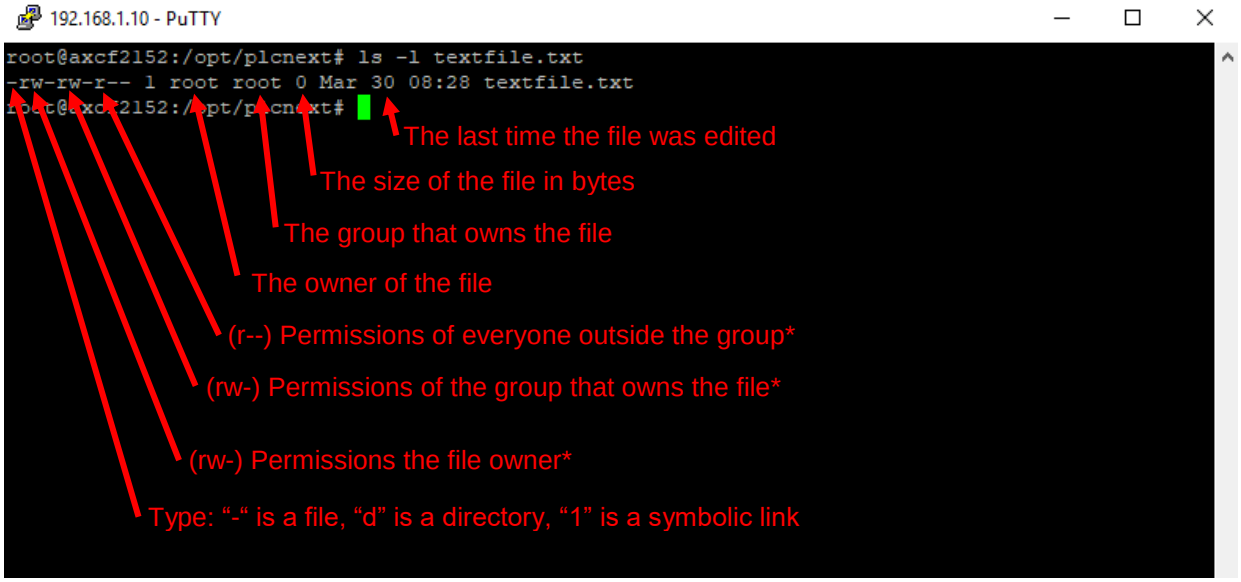


4 – Changing permissions of a file

This section will go over how to change permissions of a file, as well as why this is necessary.

<u>4.1 – ls -l:</u> This command lists the current permissions of a file. Figure 4.1-1 explains each part of the returned string.

Figure 4.1-1 Current permissions of a file



<u>4.2 – File Permission Types:</u> This section goes over the permission types for files and directories. The permissions of the file are the red characters in the line below. The description of who the permissions apply to are listed in figure 4.1-1.

-**rw-rw-r—**1 root root 0 Mar 30 08:28 textfile.txt

r: The file is readable

w: The file can be changed/modified

x: The file can be executed

Based on the red characters above, we have read/write privileges as the root user and users in the root group; and all other users have read privileges only.

4.3 – Writing new permissions to a file: This section is going to explain the options for changing a file's permissions, and how to easily determine the code to write when changing permissions of a file.

Each file permission type has a number value associated with it:

r-- (read) = 4

-w- (write) = 2

--x (executable) = 1

rw- (read and write) = 4 + 2 = 6

rwx (read, write and execute) = 4 + 2 + 1 = 7

No permissions = 0

Each digit (ones, tens, hundreds) is the user/user group that gets the permissions. The hundreds digit is the owner of the file, the tens digit is the group, and the ones digit is all other users.

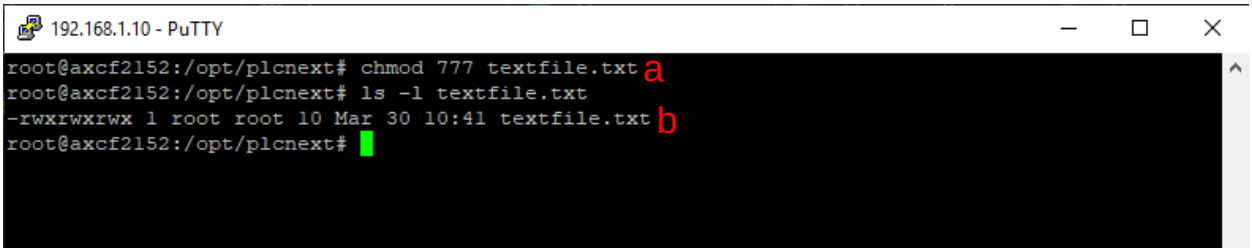
Therefore, if you want to change the permissions to allow for the owner to read/write/execute, and no one else to view/edit/execute the file; the code would be 700.

To change the permissions of a file, type the command below (a), and reference figure 4.3-1.

```
chmod 3_digit_code_here file_name_here
```

As seen in figure 4.3-1, the permissions have changed to allow everyone read/write/execute access to testfile.txt (b).

Figure 4.3-1 chmod example



5 – Running a script file (.sh)

This section will explain how to make sure the script file is executable, and to execute the script.

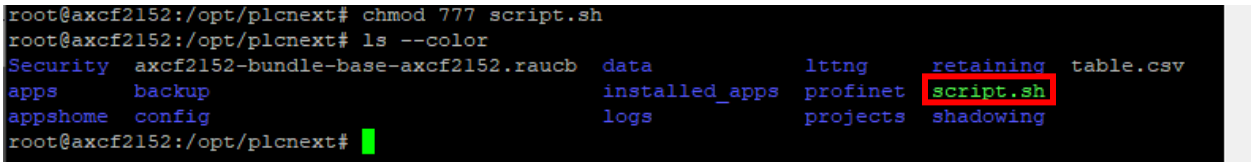
<u>5.1 – Ensuring the script is executable:</u> Before running the script, the file needs to be executable from the user.

To check, navigate to the folder the script file is in, and type the command below:

```
ls --color
```

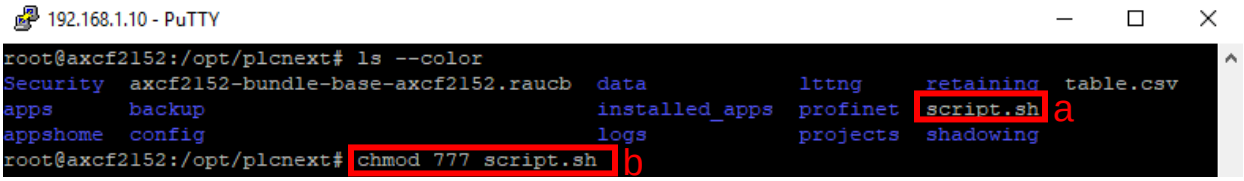
If the script file looks like the file in figure 5.5-1 then skip to 5.2.

Figure 5.5-1 Showing executable .sh file



If the file is not executable (a), you need to change the permissions of the file (b). For this example, we are going to give all users all permissions as shown in figure 5.5-2.

Figure 5.5-2 Changing permissions on .sh file



Once the script file is green as shown in figure 5.5-1, we can execute the file. The type of script that we will be showing in this example is a bash script.

<u>5.2 – Executing the bash script:</u> Once the file is green (shown in 5.1), we can execute the script file.

To execute type the below command, and refer to figure 5.5-3 for more information.

```
./filenamehere
```

The example scripts contents are below:

```
#!/bin/bash

echo "Script ran!"
```

To learn more about bash scripts, refer to https://linuxconfig.org/bash-scripting-tutorial

Figure 5.5-3 Executing a bash script