

Your Ultra.cc Shell - A Beginner's Guide

The terminal is your gateway to the inner workings of your Ultra.cc slot. It is essentially a remote computer running Linux, and when you SSH into it, you are connecting to that computer's terminal. For instructions on how to connect to your Ultra.cc slot via SSH, follow [this link](#).

A terminal is a powerful tool where you can execute written commands to perform many simple or even very complicated tasks. This guide will help you understand a few simple commands to get you started navigating and using the Linux terminal; it can also serve as a cheat sheet later down the track if you need to recall anything you may have forgotten.

We are also keeping a curated list of pre-installed utility tools that you can use on your Ultra.cc slot. When you have familiarized yourself with the guide you are reading right now. The next step would be to check out the list of pre-installed tools, and it can be found [here](#).

Introduction

Once you have connected to your Ultra.cc service via [SSH](#), your terminal prompt will be placed in your home directory. This can be confirmed with the `pwd` command, which stands for "print working directory". By executing this command, you will see the absolute path of the current working directory, as shown below.

```
ultradocs@spica: ~$ pwd
/home/ultradocs
```

However, the `/home/ultradocs` path is just a symbolic link. As there are multiple users on each server, and all of them require their own home directory. To see the physical home directory path of your home directory, you need to append the `-P` option to the `pwd` command. As you can see from the below output, the physical home directory path is different.

```
ultradocs@spica: ~$ pwd -P
/home12/ultradocs
```

Just as in the Windows operating system where each local user have their files stored in `C: /Users/username`, Linux also has local users. Each user have their own individual home directory, which is `/home/username` where `username` is your Ultra.cc service username.

The absolute path of the Linux home directory is `/home/username`, but it can also be shortened to `~/`, which can be useful when navigating the terminal. However, while editing scripts, or configuring application settings, it is always best to use the absolute path to avoid problems.

Default File/Folder Structure

In the home directory of your Ultra.cc service, there will be a set of default files and folders. Some of them are standard files that are included in every session of Bash, while some are unique to Ultra.cc. In your Ultra.cc service, the default file/folder structure would be what is stated below.

- `~/.bashrc`
- `~/.profile`
- `~/bin`
- `~/apps`
- `~/downloads`
- `~/media`
- `~/www`

These files and folders are an integral part to keep your service functional, and should not be deleted. However, should you by mistake delete any of these files and folders, they can be restored by following the instructions in [this guide](#).

As you can see in the above list of files and folders, some of them have a `.` prefixed in their name. These files are commonly known as "dotfiles" and usually serves the purpose of being important configuration files for your system. The dot signifies that a file or folder is hidden, and will therefor not show up by default when you browse the folder structure in your FTP client, or when executing the `ls` command. However, with `ls -a`, hidden files will be shown in the output.

Bash Startup Files

Whenever you are interacting with the terminal of your Ultra.cc service, you are using a shell called Bash aka Bourne Again Shell. To initialize Bash when you connect via SSH, your home directory must include a set of Bash startup files. The two most important are `.bashrc` and `.profile`. These files can be identified by having a prefixed `.` in their names. Below is examples of Bash files that you might find in your home directory.

- `~/.bashrc`
- `~/.profile`
- `~/.bash_profile`
- `~/.bash_logout`
- `~/.bash_history`

Navigation

In this section, we will detail the commands used to move between directories in Linux; we will also cover how to find your location and the structure of your slot.

cd

The first thing you are going to want to do in your terminal is to navigate between directories. All tasks are performed within different directories, and your current directory affects how a command may run or how you may need to use a command.

When you first SSH into your slot, you will be in your personal **home** directory. This is the location of all your files; anything you wish to accomplish must be done within the **home** directory as this is the only folder you have read, write and execute permissions for. If, while navigating your terminal, you wish to return to your **home** directory, you can do so by typing `cd ~`.

This brings us to the command you will likely find yourself using a large amount of the time; `cd`. This command is used to navigate between different directories in Linux and stands for *change directory*. To use this command, you simply type `cd` followed by the directory you wish to navigate to.

In Linux, when navigating using the `cd` command, there are multiple ways to tell the operating system where exactly you want to be. If, for example, you want to move one directory deeper from a directory you are already in, you would use `cd [directory]`. If you know the directory structure already and you simply want to go to a path relative to your current location, you can use `cd path/to/directory` where each `/` stands for a directory deeper. Just like `~` refers to the **home** directory `/` refers to the root of the filesystem. If, for example, you wanted to navigate to a folder called **media** in your **home** directory, you would use the command `cd ~/media`.

It is also worth noting that in Linux, `.` refers to the current working directory and `..` refers to the directory above. This is useful in if you want to go back one directory you can type `cd ..`, or if you want to go back two you can use `cd ../..` and so on.

ls

Knowing how to navigate your directories is all well and good, but how do you know where you actually want to go? For this, we use the `ls` command. Typing `ls` within any directory in Linux will instantly show you all the folders in said directory. You can also use `ls` to list the directories inside a different directory using `ls path/to/directory`.

pwd

The `pwd` command is used to show you the exact location you are currently in, fairly simple, really. Running `pwd` will print the full path to your current working directory. Very useful for when you may be lost.

File and Directory Manipulation

At some point, you are going to want to move files/directories around, copy or delete them. Here we will cover the commands involved in achieving this.

mv

The `mv` command stands for *move*; you can use it to move a file or directory to another directory. It is as simple as using the command `mv [file or directory you wish to move] [where you wish to move it]`. For example, if you wanted to move a directory called **media** to a directory called **files**, you would use the command `mv media files`. If, however, you wish to move all the files from the directory called **media** into the directory called **files**, you would use `mv media/* files`. This tells the operating system to move all the files (signified by `/*`) located in **media** directory.

cp

The `cp` command stands for *copy*; it is used in much the same way as `mv`; however, there are a few little details worth covering. The general Syntax is the same as `mv`; if you wish to copy a file in your current directory named **Movie1.mkv** to a subdirectory called **media** you'd used the command `cp Movie1.mkv media`. The difference

comes when you wish to copy an entire directory, when you want to copy an entire directory, you need to tell the `cp` command to copy *recursively*. This is signified with a `-r` after the `cp`. For example, if you wished to copy the directory called **media** to **files** similar to the previous example, you'd use the command `cp -r media files`. Copying all the files within **media** to **files** is pretty much exactly the same; `cp media/* files`. However, if there are any sub-directories within **media** that you also wish to copy, you again need to use the `-r` flag; `cp -r media/* files`. The `-r` flag is used whenever you need to copy an *entire directory*.

rm

The `rm` command stands for *remove*, it is used to delete files or directories you no longer wish to use. **It is very important to keep in mind that there is no recycling bin in Linux, any files you delete are permanently lost.** Syntax for `rm` is much the same as `cp` if you wish to remove a file named **Movie1.mkv** you'd simply type `rm Movie1.mkv`. If you wish to remove an entire directory, once again, you need to use the `-r` flag; `rm -r media` would delete the entire **media** directory. If you wanted to remove all the files inside the **media** directory, you'd use `rm media/*` or `rm -r media/*` if there are directories inside the **media**.

unzip

The `unzip` command can be used to manually decompress a ZIP file via SSH terminal. The steps to unzip a file are simple, simply `cd` to a path where the ZIP file is located. For example, `cd /path/to/location` and execute `unzip archive-name.zip`. This will unpack the ZIP file in the current directory. Check the content using the `ls` command.

unrar

The `unrar` command can be used to manually decompress a RAR file via SSH terminal. The steps to unrar a file are simple, simply `cd` to a path where the RAR file is located. For example, `cd /path/to/location` and execute `unrar x archive-name.rar`. This will unpack the RAR file in the current directory. Check the content using the `ls` command.

For a full list of pre-installed utility tools, see [this guide](#).

Further Info

For any commands, you need to find more information on; you can use the command `man [command]`. This displays a manual page for the specified command.

Ultra.cc Specific Commands

Application Management

When interacting with your slot through Secure Shell, you may wish to manipulate the applications we offer one-click installers for; to do this, you use the `app-[appname]` commands.

Installing and Uninstalling Apps

To install an application, we can use the command `app- [appname] install`. This can be helpful to gain further knowledge as to why certain installations are failing. When installing an application, you will be presented with an output in the SSH prompt showing either true or false.

Likewise, to uninstall an application, we can use the command `app- [appname] uninstall`. This is especially useful in the case of an application saying it's installed while not being accessible from the UCP.

Stop, Start and Restarting Apps

To change the running state of an application we use `app- [appname] [start| stop| restart]`. These commands will be useful for you throughout the usage of your USB slot for troubleshooting or further customize your experience.

Checking Used Space

To display your folder structure and space taken by individual files, the `ncdu - x` command is included in every Ultra.cc slot. After a short scan largest items are displayed top, and size is in descending order

Further Info

For further information regarding specific commands for a particular application run `app- [appname] help`. This will provide all sub-commands available for the said app.