

# Access Batocera Linux via ssh

First, make sure that SSH is enabled on your box — by default, on a fresh install, it is enabled. Check your `/userdata/system/batocera.conf` file and make sure you have:

```
system.ssh.enabled=1
```

When SSH is enabled, the only Linux user on a Batocera system is root:

```
ssh root@batocera.local
```

- user is always root
- the default password is `linux`
- if you enabled **SYSTEM SETTINGS** → **SECURITY**, the root password is visible from this menu.

For passwordless authentication, you can add your public SSH keys (from `~/.ssh/id_rsa.pub` on your client machine) to the list of authorized keys on your Batocera box into the file `/userdata/system/.ssh/authorized_keys`.

Windows users are advised to use Putty to connect through ssh: <https://www.putty.org>  
Linux and MacOSX users can use command line ssh from a terminal.

## Basic SSH commands

Once you are connected to batocera, you can use the command-line to operate on the system.

Know that upon connecting, you will be put into the HOME folder of Batocera, this folder is called `~` and is located at `/userdata/system`.

Some command-line tools have an `--help` option describing how to use the program, for exemple, typing `cp --help` will print how the command is used in the command-line interface.

```
BusyBox v1.31.1 (2020-11-28 14:11:54 CET) multi-call binary.
```

```
Usage: cp [OPTIONS] SOURCE... DEST
```

```
Copy SOURCE(s) to DEST
```

```
-a      Same as -dpR
-R, -r  Recurse
-d, -P  Preserve symlinks (default if -R)
-L      Follow all symlinks
-H      Follow symlinks on command line
-p      Preserve file attributes if possible
-f      Overwrite
-i      Prompt before overwrite
-l, -s  Create (sym)links
-T      Treat DEST as a normal file
```

```
-u      Copy only newer files
```

Most command-line tools have a similar syntax: `CommandName -Option  
Parameter_of_the_option path/to/input/file path/to/output/file`

Using the command line with paths of files can be confusing at first, there are two types of paths:

- **Absolute paths** : they will always be the same ones regardless of the current directory you are in, and they start with a / character, for exemple `/userdata/saves`
- **Relative paths** : they are relative to you current position, for exemple, if you are in the `/userdata` directory, and you use `nano system/batocera.conf`, you will execute the command `nano` on the file `batocera.conf` located in the `/userdata/system` folder.

If a path, or a filename, contains special characters or spaces, you will need to put either single quotes `'` or double quotes `"` around it.

If a command needs to be manually closed, doing `Ctrl+C` will end it.

Here is a subset of the commands you might want to know:

## Basic file usage

A tool included into Batocera is Midnight commander, it is a powerful command-line file explorer, it should be enough for most users :

- `mc` : starts the command-line file explorer **M**idnight **C**ommander, this tool can be used to move, copy, delete, rename, and edit files, as well as create folders, make symbolic links, and change individual file permissions. This utility also supports a mouse input.

You can also use the followings command to manipulate files :

- `cd` : changes the current working directory [**C**hange **D**irectory], for exemple `cd /userdata/roms` puts you in the `/userdata/roms` folder
- `cp` : copies a given file or folder to another path [**C**o**P**y], for exemple `cp /userdata/system/batocera.conf /userdata/batocera.conf` will create a copy of `batocera.conf` in the `/userdata` folder.
- `du -sh` : gives you the size of the specified element [**D**isk **U**sage, **S**pecified, **H**uman-readable], for exemple `du -sh /userdata/roms/snes` gives you the size of the `snes` roms folder (If you use batocera on a PC, and access it from a windows computer with the file manager, the size of some elements may not be accurate, especially for `.wine` games, so this command lets you see the actual size)
- `ls` : lists the files and folders present in the current directory [**L**i**S**t].
- `mkdir` : creates a directory [**M**a**K**e **D**i**R**ectory], for exemple `mkdir content` will create a directory called `content` in the current working directory.
- `mv` : moves a given file to another path [**M**o**V**e], for exemple `mv /userdata/roms/gb/game.zip /userdata/roms/gbc` will move the file `game.zip` from `gb` to `gbc`; the `mv` command can also be used to rename files, for exemple, `mv /userdata/roms/gb/game.zip /userdata/roms/gbc/gb_game.zip` would rename the file when moving it.
- `nano` : opens a command-line text editor for the specified file, for exemple `nano /userdata/system/batocera.conf` opens the file `batocera.conf` to edit it, for more infos

see [this link \(external\)](#).

- `rmdir` : deletes a directory if it is empty [**ReMove** **DIR**ectory], for exemple `rmdir content` will delete the directory content if it is empty.
- `rm` : deletes a specified file [**ReMove**], for exemple `rm invaders-201226-124223.png` will erase the file `invaders-201226-124223.png` in the current working directory.
- `rm -r` : deletes a directory and all the files it contains [**ReMove**, **Re**cursive], use with caution!
- `unzip` : decompresses a given .zip file in the working directory, for exemple `unzip file.zip` will extract all the data in `file.zip` in the current working directory. You can ask to extract it elsewhere using the `-d` option [**DIR**ectory], for exemple, `unzip file.zip -d uncompressed` will extract all the data in `file.zip` inside a subdirectory called `uncompressed`.

## Debugging

The following commands can be used to debug Batocera, some of them are part of the `batocera-es-swissknife` command

- `aplay -l` : returns the list of playback hardware devices [**A**udio **PLA**yer, **L**ist] (can be used to debug audio issues on PC).
- `batocera-es-swissknife --arch` : prints the current **ARCH**itecture running.
- `batocera-es-swissknife --emukill` : **KILLS** any **EMU**lator running.
- `batocera-es-swissknife --restart` : **RESTARTS** EmulationStation.
- `batocera-es-swissknife --reboot` : **REBOOTs** the entire system.
- `batocera-es-swissknife --shutdown` : **SHUTDOWNS** the entire system.
- `emulationstation` : starts emulationstation.
- `kill` : kills a process with a given PID, for exemple if `pidof retroarch` returned 640, then doing `kill 640` will terminate the `retroarch` process.
- `pidof` : gives a list of processes identifiers (PID) for a running process name [**P**rocess **ID**entification **OF**], for exemple `pidof retroarch` returns a number when a `retroarch`-based emulator is running.
- `raspi2png` : Exclusive to **raspberry pi**, saves a screenshot of batocera in the current working directory, **to** a **.png** file.

## Internet fonctionnalités

- `batocera-upgrade` : let's you update batocera using the command-line with the correct URL, see [Manual upgrades/downgrades](#) for more details
- `pacman -Ss` : let's you search through the pacman packages using the command-line [**PAC**kage **MAN**ager, **S**ync], see [Batocera Package Manager \(pacman\)](#) for more infos.
- `pacman -S` : let's you install a package using it's name [**PAC**kage **MAN**ager, **S**ync, **S**earch], see [Batocera Package Manager \(pacman\)](#) for more infos.
- `pacman -Rsd` : let's you remove a package using it's name [**PAC**kage **MAN**ager, **R**emove, **re**curs**I**ve, skip **D**ependancies checks], see [Batocera Package Manager \(pacman\)](#) for more infos.
- `pacman -Scc` : clears the entire cache of the pacman manager [**PAC**kage **MAN**ager, **S**ync, clear **C**ache (extra **c** forces a complete clear)], see [Batocera Package Manager \(pacman\)](#) for more infos.
- `pacman -Sy` : Updates the pacman database [**PAC**kage **MAN**ager, **S**ync (extra **y** refreshes the database)], see [Batocera Package Manager \(pacman\)](#) for more infos.

## Logging to a file

Sometimes, you might prefer having the output of a command inside a separated text file instead of reading through the command line interface. to do that you can use the `>` and `>>` symbols followed by the path of a filename.

for exemple, the command `ls` gives you the list of all files and folders in your current working directory, if you want to gather this inside a text file named `list-files.txt` at `/userdata/system`, you simply need to use:

- `ls > /userdata/system/list-files.txt`, the command will return nothing, and instead the files will be listed inside a new file called `list-files.txt` located in `/userdata/system`. Running the command again will replace the content of `list-files.txt`.
- `ls >> /userdata/system/list-files.txt` however will add the output of the command to the existing file, without removing the previous infos.

## Custom aliases

While not mandatory, it is good to know you can create aliases of commands, they let you launch a command with a simple keyword. One way to do this can be done by creating a text file in `/userdata/system`, that file must be named `.profile` : it will contain aliases commands and will execute them when Batocera is launched, so be careful of what you do.

The alias command syntax is as follow:

```
alias customname='du -sh /userdata/system/batocera.conf'
```

In this example, once batocera has been restarted, entering `customname` in the command line will do the same as entering `du -sh /userdata/system/batocera.conf` (in this example, it basically returns the size of the `batocera.conf` file in the command-line interface)

a more useful, yet complicated example, would be to try and use `ffmpeg` to save a screenshot of Batocera in the screenshots folder for `x86_64` devices, as devices other than raspberry pies don't have an equivalent of `raspi2png`, in this example I call the custom alias "`pc2jpeg`".

```
alias pc2jpeg='ffmpeg -hide_banner -loglevel error -f x11grab -i :0.0 -frames:v 1 /userdata/screenshots/$(date +%y-%m-%d_%H-%M-%S).jpg'
```

This commands does the following :

- `ffmpeg` : an command-line encoder included inside batocera.
- `-hide_banner` : prevents `ffmpeg` from printing the copyright notice when running this command.
- `-loglevel error` : will only alert in the command-line if there is an actual error preventing the command from working.
- `-f x11grab -i :0.0` : uses `x11grab` to capture the screen 0 (this lets you use the screen as the source).
- `-frames:v 1` : the number of video frames to capture.
- `/userdata/screenshots/$(date +%y-%m-%d_%H-%M-%S).jpg` : the full path to the file we want to save.

- `$(date +%y-%m-%d_%H-%M-%S)` : executes the `date +%y-%m-%d_%H-%M-%S` command and returns its result in the command line (for example, it will return `20-12-31_21-52-19` the 31st december of 2020 at 21h52m19s), so the path before would be read as `/userdata/screenshots/20-12-31_21-52-19.jpg`.

So by using the `pc2jpeg` custom command, a `.jpeg` file will be created with a filename based off the date of the screenshot.

## Advice

Whenever you are using a command-line, try to understand what you are doing, even if it doesn't seem like it, some of them can be used in harmful ways. for example, putting simply `> empty.txt` without any command will create an empty file called `empty.txt` in the current working directory, however if you already had a useful file with this name, it'll be empty after this command.

From:

<https://wiki.batocera.org/> - **Batocera.linux** - Wiki

Permanent link:

[https://wiki.batocera.org/access\\_the\\_batocera\\_via\\_ssh?rev=1610116670](https://wiki.batocera.org/access_the_batocera_via_ssh?rev=1610116670)

Last update: **2021/01/08 15:37**

