Ports (native to Linux)

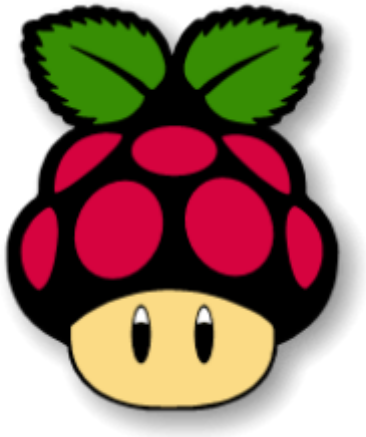
Ports are great. You can just straight up run Linux games on Batocera, if the game itself carries all of its dependencies. Fun fact: most don't. You'll typically need to refer to the game's documentation or scour around the internet (the Arch Wiki and the PCGamingWiki are pretty good resources for this) to find out how to get it working in Batocera.

If the game is available as an AppImage, try that as it has the highest chances of working out of the box.

In Batocera **v32** and lower, you would have to create the `roms/ports/` subfolder yourself first in order to start using it.

This system scrapes metadata for the “ports” group(s) and loads the `ports` set from the currently selected theme, if available.

Grouped with the “ports” group of systems.



Quick reference

- **Accepted ROM formats:** `.sh`
- **Folder:** `/userdata/roms/ports`

Program files

In many cases a physical (USB) keyboard is required to configure the native Linux game for the first time (most times to set up the physical controller once in-game if not automatically configured).

Place the data of the game (whatever that may be) into `roms/ports/.data`. This will automatically

hide any SH scripts that might be packaged with the game.

Then create an appropriate SH file to launch the game's actual executable file with. Take the file below and fill it as appropriate:

Your game name here.sh

```
# These events will be executed before the game begins.
# Read the current directory this script is being executed from and
save to variable DIR.
DIR="$(dirname "$(readlink -f "$0")")"

# Change the current directory to that of the game's data folder.
cd "${DIR}/.data/<folder of game here>"

# Show the mouse cursor. It's a good idea to first show it when setting
up the game, in case it's needed.
unclutter-remote -s
./<filename of game executable here>

# These events will be executed once the game terminates.
# Hide the mouse cursor as we are going back to ES.
unclutter-remote -h
```

Save this script to the `roms/ports` folder, refresh the game list, and then launch the game. Most 3D games will spend some time generating shader caches when first booting up, so be patient at least on the first launch. For sanity's sake, you can check that your system has not frozen by moving your mouse around (if you have one connected). Shader compilation should not take longer than five minutes at most.

If using the SquashFS compression, this script file must be named `run.sh` and included in the root folder of the game's data itself in the `.squashfs` archive.

If you'd like to see the actual files, you can install some native Linux ports in the content downloader. Read below for some setup examples:

Todo: Example of a freeware game so users can test it out!

Self-contained itch.io game example: Skatebird

Skatebird can be purchased and downloaded from its itch.io page. Make sure it's the “Linux” version!

1. Once downloaded, extract and then copy the `skatebird-linux` folder from the ZIP file to `roms/ports/.data` (you may need to enable “Show hidden folders” in your file explorer to see this).
2. Download the following script and save it to `roms/ports/Skatebird.sh`:

Skatebird.sh

```
DIR="$(dirname "$(readlink -f "$0")")"

cd "${DIR}/.data/skatebird-linux"

unclutter-remote -s
./SkateBIRD.x86_64
unclutter-remote -h
```

The `unclutter-remote -s` and `unclutter-remote -h` commands are to **s**how and **h**ide the mouse cursor respectively. It's a good idea to first show it when setting up the game, in case it's needed.

3. Press [START] → **GAME SETTINGS** → **REFRESH GAMELISTS**.
4. Navigate to the “Ports” system, then launch “Skatebird”.

This game in particular may take a very long time to initially load with its default settings (which are at maximum fidelity), during which time you'll only see a blank, grey screen. Once in-game these can be lowered down, resulting in much faster launch times in the future.

Self-contained popular musical game: Clone Hero

Clone Hero is a free, but not opensource, Guitar Hero/Rock Band clone that works with many accessories. They provide a build for Linux x86_64, you won't be able to install it on a Raspberry Pi or other ARM-based boards.

1. Go to the download page and download it on your Batocera machine.
2. Once downloaded, extract the '.tar' archive into `roms/ports/CloneHero` and you will end up with all the files installed into `roms/ports/CloneHero/Standalone`.
3. Make sure the main app is exectuable (`chmod +x /userdata/roms/ports/CloneHero/Standalone/clonehero`)
4. Download the following script and save it to `roms/ports/CloneHero.sh`:

CloneHero.sh

```
DIR="$(dirname "$(readlink -f "$0")")"

cd "${DIR}/CloneHero/Standalone"

unclutter-remote -s
./clonehero
unclutter-remote -h
```

The `unclutter-remote -s` and `unclutter-remote -h` commands are to **s**how and **h**ide the

mouse cursor respectively. It's a good idea to first show it when setting up the game, in case it's needed.

1. Press [START] → **GAME SETTINGS** → **REFRESH GAMELISTS**.
2. Navigate to the “Ports” system, then launch “CloneHero”.

Complicated itch.io game example: Celeste

As already mentioned above, the `/userdata/roms/ports/.data` folder is where you want to place your game's main folder which contains the according native Linux game. The following example will show how you can make the popular game Celeste running on Batocera.

First of all, you will have to buy/download the native Linux version of Celeste, which will give you a single package file, e.g. `celeste-linux.zip`. After successfully downloading, copy the file over from your local computer to your Batocera system (for example via WinSCP) to the following path: `/tmp/celeste-linux.zip`.

Now SSH into your Batocera system and execute the following command to extact the package file's content to the correct path:

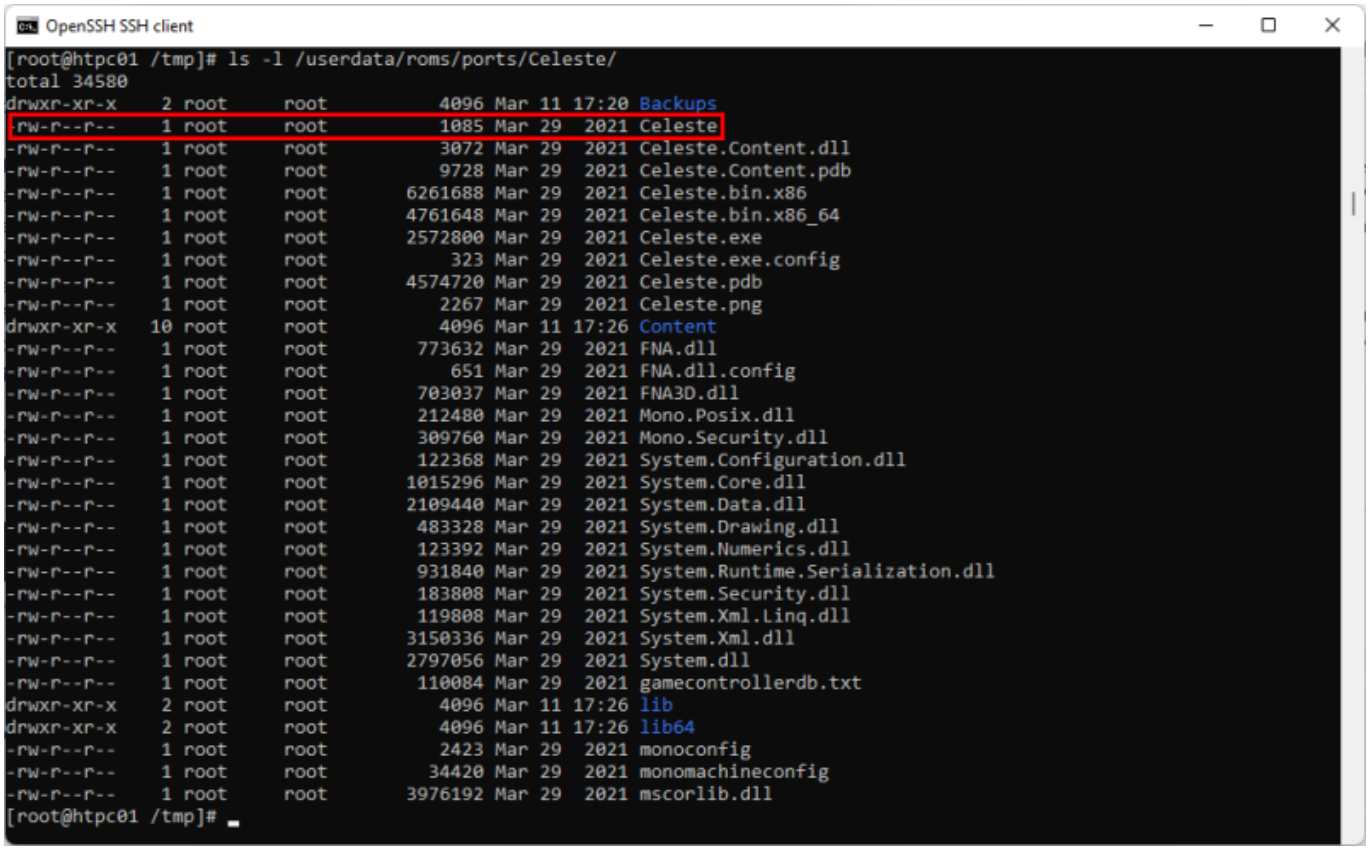
```
unzip -d /userdata/roms/ports/Celeste /tmp/celeste-linux.zip
```

In most cases, depending on where you've downloaded your Celeste game sources, the game is well “prepared” for running it as a native Linux game. In other words: the downloaded game package should contain all files/dependencies without the hassle of having to search/add manually any external files/dependencies. Unfortunately, as already mentioned above, this definitely is not the case for all native Linux games!

Now check the content of the game's directory…

```
ls -l /userdata/roms/ports/Celeste
```

…which should give you the following similiar output:



For Celeste, in his case, the correct executable file to start the game is `/userdata/roms/ports/Celeste/Celeste` as marked in the screenshot above. Now to make the game show up as “Celeste” in your Batocera GUI's “Ports” section, you would have to have a file called `/userdata/roms/ports/Celeste/Celeste.sh`. You could indeed rename the game's main executable file from `/userdata/roms/ports/Celeste/Celeste` to `/userdata/roms/ports/Celeste/Celeste.sh` and adjust it afterwards, but a better, more flexible and more reliable way would be to leave the source files untouched and create a separate executable file which will call the main executable file. To do so, just run the following command:

```
nano /userdata/roms/ports/Celeste/Celeste.sh
```

Now paste the following content:

Celeste.sh

```
#!/bin/bash
cd /userdata/roms/ports/.data/Celeste && export DISPLAY=:0.0; ./Celeste
```

Save the file and quit the editor.

As you may have mentioned on the screenshot above, none of the executable files are marked as executable yet, which will be required to run “Celeste” sucessfully. So let's do this by executing:

```
chmod +x /userdata/roms/ports/Celeste/Celeste.sh
/userdata/roms/ports/Celeste/Celeste
/userdata/roms/ports/Celeste/Celeste.bin*
```

The required executables are different for every native Linux game. There's no general rule you can follow, you have to find out by yourself by searching through the internet or by just trying to execute the executables via command line and analyze the errors that may occur (see the troubleshooting section below).

Now update the gamelist via the Batocera GUI and the game should show up in the “Ports” section. That's it, start the game from there and have fun! 😬

Vulkan dependent game example: "Teenage Mutant Ninja Turtles: Shredder's Revenge" (Linux patched version)

Some games are dependent on the Vulkan graphics API which can get enabled by command line when starting the game's executable as shown in the following example game's startup script.

Note that a game is dependent on Vulkan, the graphics card needs to support it. In other words: If a graphics card doesn't supports the Vulkan API natively, it won't be possible to run the game. There's no way to emulate graphics cards with Vulkan API support (yet) if the game is meant to run directly on a bare-metal computer environment. It *might* work in a virtual computer environment with a virtual GPU (vGPU) but is barely supported by any hypervisor. Only VMware hypervisors are supporting Vulkan for their virtual GPUs (vGPUS) these days, which might change for other hypervisor vendors in the future. But to run Batocera in a VM is not suggested anyway.

TMNT.sh

```
#!/bin/bash
export LC_ALL=C
export MONO_IOMAP=all
# Move to the game directory
cd "$(dirname "$(realpath "$0")")" || exit
# Run the game
chmod +x ./TMNT.bin.x86_64
exec ./TMNT.bin.x86_64 /gldevice:Vulkan "$@"
exit 0
```

Make the script executable:

```
chmod +x /userdata/roms/ports/TMNTSR/TMNT.sh
```

Update the gamelist and start the game via the ES its “Ports” section.

Library-dependent game: Final Fight LNS Ultimate

Final Fight LNS Ultimate is a game which is dependent on the libvpx library package that is not included with Batocera by default. This can be provided alongside the game.

Copy the contents of the game's data to the `roms/ports/Final Fight LNS Ultimate/` folder, then create a new subfolder named `LIBS` at `Final Fight LNS Ultimate/LIBS`. Download the libvpx as appropriate to your achitecture and place it in that folder.

Back in the `Final Fight LNS Ultimate/` folder, save the following script file:

Final Fight LNS Ultimate.sh

```
#!/bin/bash

# Only applied to version FFLNSU V03, for V04 comment those export
lines!
export LC_ALL=C
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./LIBS/

# Move to the game directory
cd "$(dirname "$(realpath "$0")")/" || exit

# Run the game
chmod +x "Final Fight LNS Ultimate" #For version V04, use Final Fight
LNS Ultimate.AppImage instead
exec "./Final Fight LNS Ultimate" "$@" #For version V04, use Final
Fight LNS Ultimate.AppImage instead
```

Note: For Final Final LNS Ultimate (V04), use the same script, but point to the Final Fight LNS Ultimate.AppImage and remove the lines about LIBS.

The final file structure should look like this (your installation may have more or less, but overall structure should be similar):

```
roms/
└─ ports/
   └─ Final Fight LNS Ultimate/
      ├─ LIBS/
      │  └─ libvpx.so.5 >> (For V03 only, not needed for V04)
      ├─ Logs/
      ├─ Paks/
      │  └─ Final Fight LNS.pak (Game file, V03 or V04)
      ├─ reshade-shaders (from Shaders package - optional)
      │  ├─ Shaders
      │  └─ Textures
      ├─ Saves/
      ├─ ScreenShots/
      ├─ 1920.ini (from Shaders package - optional)
      ├─ CAMBIOS - Final Fight LNS Utimate V03/V04.txt
      ├─ Final Fight LNS Ultimate >> (Final Fight LNS Ultimate.AppImage for
V04 instead - game launcher)
      ├─ Instrucciones.txt (from Shaders package - optional)
      ├─ Legal.txt
      ├─ opengl32.dll (from Shaders package - optional)
```

```
    ├─ ReShadeGUI.ini (from Shaders package - optional)
    ├─ ReShade.ini (from Shaders package - optional)
    └─ Final Fight LNS Ultimate.sh
```

Update gamelists and launch!

Old 32-bit game

For games which require 32-bit libraries and can't find them on their own:

run.sh

```
#!/bin/sh
cd "$(dirname "$(readlink -f "$0")")"

unclutter-remote -h
export LD_LIBRARY_PATH=/lib32
export LIBGL_DRIVERS_PATH=/lib32/dri
export SPA_PLUGIN_DIR="/lib32/spa-0.2:/usr/lib/spa-0.2"
export PIPEWIRE_MODULE_DIR="/lib32/pipewire-0.3:/usr/lib/pipewire-0.3"
exec ./start.sh
```

Old Game Maker Studio game: Maldita Castilla

Games written in older versions of Game Maker Studio will depend on a collection of 32-bit libraries not included with Batocera. (If the game's executable is just named `runner`, that's a good sign that you have a GMS game... though some developers will rename it.)

More broadly, if the `file` command on a more general-purpose Linux setup says that the game executable is an `ELF 32-bit LSB executable, Intel 80386`, then these instructions will likely get it working, though they may be overkill.

This blog post details how to build the required library bundle and includes a mention that, if you trust the author, you can just grab the `lib` folder and adapt the wrapper script from their River Raid Squadron game on Itch.io.

(If both of those links die, you can just build the libraries the error messages complain about one-by-one until the game works.)

If you place the `lib` folder inside the game folder, a suitable SH file will look something like this:

Maldita Castilla.sh

```
#!/bin/sh
DIR="$(dirname "$(readlink -f "$0")")"

cd "${DIR}/.data/MalditaCastilla"
```

```
unclutter-remote -h
export LD_LIBRARY_PATH=./lib:/lib32
export LIBGL_DRIVERS_PATH=/lib32/dri
export SPA_PLUGIN_DIR="/lib32/spa-0.2:/usr/lib/spa-0.2"
export PIPEWIRE_MODULE_DIR="/lib32/pipewire-0.3:/usr/lib/pipewire-0.3"
./runner
```

Installing the Java runtime environment

Some games may require the Java runtime environment. Ordinarily, this would just be installed from the operating system's package manager, however Batocera has no repository which includes or maintains the JRE. However, there are portable packages that allow for JRE to run in a portable form.

Download the JRE appropriate to your system from https://github.com/alinke/pixelcade-jre/ (as of writing, x86_64, x86, aarch32 and aarch64 have packages) and extract it to the `system/java` folder.

Either the executable can be pointed toward this location (if it has the appropriate arguments), or run the following SSH command to allow `java` to be run as if it were installed like a regular component:

```
export PATH=/userdata/system/java/usr/bin:$PATH
batocera-save-overlay
```

This will be required to be run every update.

In case the latest version of JRE is required (or your exact platform isn't available from the prepared packages above), refer to the manual method:

Manual method (click to expand)

Instructions taken and modified from https://docs.azul.com/core/zulu-openjdk/install/rpm-based-linux#install-from-binary-tar-gz-file

1. Go to https://www.azul.com/downloads/?package=jdk
2. Scroll down and download the latest `.tar.gz` package available for the appropriate platform
3. Extract the `.tar.gz` file to anywhere on your userdata (from SSH, run `tar -xvf <zulu_package>.tar.gz` to extract it)
4. Run `<zulu_package>/bin/java -version` to check that it was extracted successfully
5. Add the file to `/usr/bin` directory to use the `java` command from anywhere: `export PATH=<installation_folder>/usr/bin:$PATH` and run `batocera-save-overlay` to keep the changes permanent (until next upgrade)

If you'd like a single line command that does all of this automatically:

- **For aarch64:** `curl -kLo - https://cdn.azul.com/zulu-embedded/bin/zulu8.58.0.13-ca-jdk8.0.312-linux_aarch64.tar.gz | gunzip -c | tar -x --strip-components=1 && batocera-save-overlay`

A note about integrated ports

Batocera-integrated ports (recognized as their own unique system, but are grouped under the “Ports” system in EmulationStation along with native Linux ports by default) usually contain an `_info.txt` file in their folder explaining how to install them and sometimes have their own page in system overview page under the “Port” header.

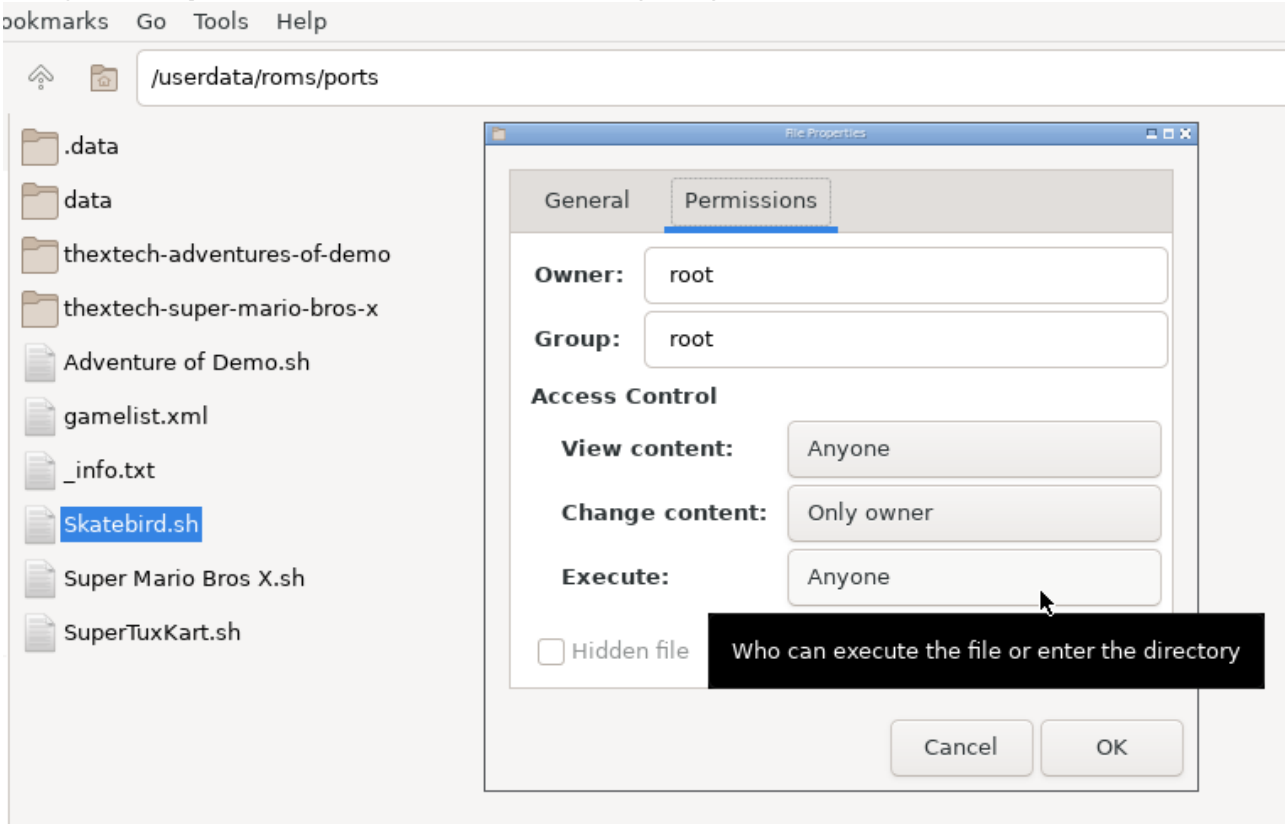
Troubleshooting

My game isn't launching

Native Linux games not running on Batocera can have many different reasons. The simplest solution is usually revealed in the error log located at `system/logs/es_launch_stderr.log` or `system/logs/es_launch_stdout.log` For testing purposes, you can use Batocera's file manager (`[F1]` on the system list) to navigate to the `roms/ports` folder and launch the game (simply double-click it as you normally would) to see if any error messages pop up.

You may need to enable the executable bit for the game in question when launching outside of ES. To do so:

1. Back in Batocera, hit `[F1]` and navigate to `roms/ports/.data`, right-click the `Skatebird.sh` file, go to **Properties** and make it executable by “Anyone”.



2. Press `[Ctrl] + [Q]` to exit back to Batocera.

If no meaningful error message comes from that, you can instead use the terminal/SSH in and try

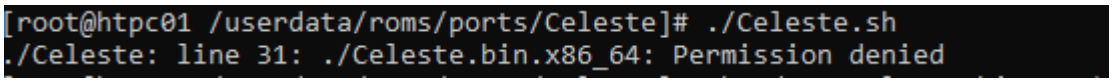
running the commands to launch the game there (pre-emptively run `export DISPLAY=:0.0` to get the game to be able to "see" your screen), this might help you see if there are any error messages being printed to the console. It's advised to be sitting in the file manager instead of EmulationStation to avoid video display issues.

Alternatively, run `/etc/init.d/S31emulationstation stop` to kill EmulationStation and `batocera-es-swissknife --restart` to bring it back to life!

For example:

```
export DISPLAY=:0.0
cd "/userdata/roms/ports/Celeste"
/bin/sh Celeste.sh
```

Sometimes there are multiple executable files involved in the process of running a game, so in many cases there may be only permission issues causing the game not to start, which would give you similar output to this:



In such a case, just set the required permission(s) (`chmod +x <yourExecutableFile(s)>`) to the according executable file(s) and you should be ready to go. In this case, it's `chmod +x Celeste.bin.x86_64`

I'm confident I'm launching it correctly but the game still isn't booting

It could be that the game is searching for a particular library on the system, but is failing to find it. Batocera includes most libraries for 64-bit and 32-bit games, but it is possible that the game itself is using an outdated name or a different method altogether.

If you've found an appropriate replacement library, or have found that it exists in Batocera just under a different name to what the game is searching for, you can (usually) tell the game to search for that particular library by adding the following to before the game is executed in its SH script:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./<path to library files>
```

For example, it's common for 32-bit games to incorrectly search in the `$LD_LIBRARY_PATH:./x86` folder. This can be corrected with the following:

```
export LD_LIBRARY_PATH=/usr/lib32
```

An example SH file for a game requiring library packages stored in a subfolder of the game would look like:

game.sh

```
#!/bin/bash
```

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:./LIBS/

exec "./game launcher" "$@"
```

Batocera's libraries are stored in `/usr/lib/`, some games may also be hard-coding their paths and try searching in `/lib` for instance, necessitating changing the path told to the game to be inside of `/usr` instead.

Other Known Bugs in Old Humble Bundle Linux Releases

- Some games (eg. Wizorb) will crash on startup if more than four joystick devices are present. (eg. If you have two dual-port controller adapters with one controller plugged into each, plus an X-Box 360 controller, and the adapters always make their ports visible to the host whether or not a controller is actually connected.)
- Some games (eg. Jamestown) which include both 32-bit and 64-bit executables will require you specifically use the 32-bit version as the 64-bit binary crashes with no known fix.
- If your game is XNA-based and you're getting a crash with the following error on startup, add an `export TERM=vt100` line to the SH file.

  ```
  System.TypeInitializationException: The type initializer for
  'System.Console' threw an exception. --->
  System.TypeInitializationException: The type initializer for
  'System.ConsoleDriver' threw an exception. ---> System.Exception: Magic
  number is wrong: 542
  ```

  The terminal type database's format got extended at some point and that error means that your game is too old to understand the extended-format entries that your terminal emulator is using to announce its capabilities.

I'm stuck and I can't quit the game!

Most native games should offer an option in the menu to “Quit back to desktop” or otherwise close the game. However, some may not or the game might freeze during execution.

Such cases can usually be escaped with good ol' trusty `[Alt] + [F4]`. But in case that doesn't work either (or you don't have your keyboard handy) you might want to try SSH'ing in and killing the host task with `htop`. Hover over the host task (indicated by being colored white) and press `[F9]` followed by `[Enter]`.

My game runs really slowly/laggy

Is your hardware powerful enough? You can usually find “recommended” requirements listed on the game's page, and if not then it might be listed on their documentations. If there's a complete absence of this information, then it usually means you don't need particularly much to run the game (such is the case with many 2D titles).

If you're confident that your hardware is powerful enough and your slowdown/lag is unusual, then try lowering the graphical settings inside of the game itself. These vary dramatically between games/engines, but usually the most influential settings are the screen resolution, anti-aliasing (MSAA, SSAA, etc.) and post-processing effects (depth-of-field, motion blur, etc.). A valuable resource for fine-tuning games in the PCGamingWiki, which may also provide more direct solutions for a game's particular problems.

Further troubleshooting

For further troubleshooting, refer to the generic support pages.

If you need further help, feel free to ask on our Discord server or forum.

From:
https://www.wiki.batocera.org/ - **Batocera.linux - Wiki**

Permanent link:
https://www.wiki.batocera.org/systems:ports?rev=1779212490

Last update: **2026/05/19 17:41**