Batocera scripts

You can opt to have scripts launched at various points in Batocera. This will change the location you save the script to and how it opens.

Launch a script, one time only during boot and shutdown

Sometimes you just want to fire up one script after successfully booting, for example when you want to start up a VPN or other after-boot tasks. In order to do this, create a script text file at /userdata/system/custom.sh. Be aware that the script will be executed independently of the executable (x) attribute being set to the script file or not!

This script is the very last one that will be invoked by Batocera at boot time (check /etc/init.d/ to see all the modules that are loaded before then, S99custom is when the user script is launched). The filename must be custom.sh otherwise it will not be checked.

Make sure your script ends with Unix line terminators (LF), not Windows-style line terminators (CR/LF) otherwise the script will not launch. Use a real text editor to edit your scripts, especially if you edit them under Windows.

To distinguish between different events, the first argument parsed in the script will change:

- **start** Only when Batocera is booting.
- **stop** Only when Batocera is shutting down.
- **restart**
- **reload**

Putting code outside of cases which check these arguments will always be executed on both boot and shutdown.

A simple one time script as an example

```
custom.sh

#!/bin/bash
# Code here will be executed on every boot and shutdown.

# Check if security is enabled and store that setting to a variable.
securityenabled="$(/usr/bin/batocera-settings-get system.security.enabled)"

case "$1" in
    start)
```

Launching a script, one time only early in the boot process

In Batocera v32 and higher, you can instead opt to launch a script early in the boot process, before much of Batocera has even begun loading. This is done by storing the script in the boot partition at /boot/boot-custom.sh. Keep in mind that you will be limited to more basic commands/modules, as most things have not loaded yet. The filename must be boot-custom.sh otherwise it will not be checked.

Precisely, this is the first thing executed once init.d has begun. This is before the network/share population/IR remote daemon have even loaded so capabilities are limited. Check /etc/init.d to see the exact order of all the modules have been loaded in (S00bootcustom is when the /boot/boot-custom.sh is executed).

This can be used to effectively patch and/or override the modules loaded by Batocera. Here are some examples that are only possible with this method:

- Customize or disable S33disablealtfn to allow a certain text mode console on a particular
TTY session.
- Tinker with hwclock to save local time instead of universal time to RTC.
- Run fsck -a or e2fsck -p to harden the file system against data corruption in the case of a power cut/disconnection.

Watch for a game start/stop event

Batocera 5.23 and higher supports running a script right before a game launches and/or after a game is exited. Here are some examples:

- Automatic controller setup
- Automatic scraping for new games
- Change screen resolution (though it might be better to use switchres for this)
- Sync savestates to an external/network device
- ... give me your idea here

Place an executable script (can have any filename) with the correct setted shebang (first line!) or an executable binary into directory /userdata/system/scripts/.

You can add as many subfolders as you want, every script placed there will be executed. To distinguish between START or STOP condition the first argument parsed in the script have either of these flags:

- gameStart is passed to your scripts if you select a game from EmulationStation (Game Starts!)
- gameStop is passed to your scripts if you are going back from a game to EmulationStation (Game Stops!)

If you do not set cases for first argument, then the script is executed on every start and on every end of a game.

What is parsed

Table of parsed arguments in correct order and their functions:

<table>
<thead>
<tr>
<th>Arg no.</th>
<th>Parsed Parameter</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>gameStart or gameStop</td>
<td>To distinguish between START or STOP condition</td>
</tr>
<tr>
<td>2</td>
<td>systemName</td>
<td>The system shortname as is in es_system.cfg, eg. atari2600</td>
</tr>
<tr>
<td>3</td>
<td>system.config['emulator']</td>
<td>The emulator settings, eg. libretro</td>
</tr>
<tr>
<td>4</td>
<td>system.config['core']</td>
<td>The emulator core you have chosen, eg. stella</td>
</tr>
<tr>
<td>5</td>
<td>args.rom</td>
<td>The full rom path, eg. /userdata/roms/atari2600/Mysterious Thief, A (USA).zip</td>
</tr>
</tbody>
</table>

Simple game start/stop script as an example

At first create the directory where the scripts need to be set up. Connect through SSH and type
mkdir /userdata/system/scripts. After this we can set up our first script by typing nano /userdata/system/scripts/first_script.sh. The filename can be anything readable by bash.

Here’s the template script:

```bash
#!/bin/bash
# This is an example file how Events on START or STOP can be uses
#
# Set logfile location and filename
logfile=/tmp/scriptlog.txt

# Case selection for first parameter parsed
case $1 in
gameStart)
    echo "START" > $logfile
    echo "$@" >> $logfile
    ;;

gameStop)
    echo "END" >> $logfile
    ;;
esac
```

Now you can see a log file created /tmp/scriptlog.txt, that parsed all arguments in this file. This is just a small test of course. You can check out the SSH page and the usage of batocera-settings page for general and Batocera-specific commands.

**EmulationStation scripting**

In case Batocera’s provided scripting functionality is not sufficient, ES itself also triggers scripts of its own volition. Extra scripts can be created at /userdata/system/configs/emulationstation/scripts/<event name>/<script>.sh and be marked as executable with the chmod +x command. For example:

```
nano /userdata/system/configs/emulationstation/scripts/game-start/first_script.sh
# write in your script, save and exit
chmod +x /userdata/system/configs/emulationstation/scripts/game-start/first_script.sh
```

Unlike with scripts that are directly called by Batocera, EmulationStation requires you to add the executable bit to your intended scripts. This means that if your userdata partition
is of the NTFS, exFAT or older file system, it will fail to execute.

<table>
<thead>
<tr>
<th>Event name</th>
<th>Arguments</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>game-start</td>
<td>ROM name rom, ROM path basename</td>
<td>When a game starts. Nearly identical to Batocera's gameStart, however this doesn't include as much information and only triggers when it's ES itself that starts it.</td>
</tr>
<tr>
<td>game-end</td>
<td>N/A</td>
<td>When a game ends. Nearly identical to Batocera's gameStop, however this only triggers when it's ES itself that ends it.</td>
</tr>
<tr>
<td>game-selected</td>
<td>getSourceFileData()→getSystem()→getName(), getPath(), getName()</td>
<td>New to Batocera v33. Whichever game is currently being hovered over. Includes games shown during the screensaver.</td>
</tr>
<tr>
<td>system-selected</td>
<td>System getSelected()→getName()</td>
<td>New to Batocera v33. Whichever system is currently being hovered over in the system list.</td>
</tr>
<tr>
<td>theme-changed</td>
<td>Theme being switched to theme_set→getSelected(), Previous theme oldTheme</td>
<td>When a different theme is selected. Mostly used for theme/element reloading.</td>
</tr>
<tr>
<td>settings-changed</td>
<td>N/A</td>
<td>When a system setting is saved.</td>
</tr>
<tr>
<td>controls-changed</td>
<td>N/A</td>
<td>When a controller mapping is saved from the CONTROLLER MAPPING menu.</td>
</tr>
<tr>
<td>config-changed</td>
<td>N/A</td>
<td>Whenever any configuration, be it system settings or controller mappings, is changed.</td>
</tr>
<tr>
<td>quit</td>
<td>N/A</td>
<td>When the system is told to do a regular shutdown.</td>
</tr>
<tr>
<td>Event name</td>
<td>Arguments</td>
<td>Notes</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>reboot</td>
<td>N/A</td>
<td>When the system is told to do a reboot.</td>
</tr>
<tr>
<td>shutdown</td>
<td>N/A</td>
<td>When the system is told to do a fast shutdown.</td>
</tr>
<tr>
<td>sleep</td>
<td>N/A</td>
<td>When the system is told to sleep. This is currently unused in Batocera.</td>
</tr>
<tr>
<td>wake</td>
<td>N/A</td>
<td>When the system is told to wake from sleep. This is currently unused in Batocera.</td>
</tr>
</tbody>
</table>

**Simple EmulationStation script example**

```bash
#!/bin/bash

# Save the arguments into variables.
system="${1}"
rom="${2}"
romname="${3}"

# Convert an argument into another value.
if [[ "${system}" == "fbneo" ]]; then
    system="mame"
fi

case ${system} in
    fbneo)
        system="mame"
        ;;
    scummvm)
        rom="${rom%.*}"
        ;;
    esac

# Execute this part every time this event triggers.
curl -G
    --data-urlencode "t=${romname}" \
    http://127.0.0.1:8080/arcade/stream/${system}/`basename ${rom}`
```
Real use cases

Batocera scripts

- Output text to a file

```bash
#!/bin/bash
# by cyperghost for batocera

testfile="/userdata/system/scripts/output.txt"
echo "END Script!" >> $testfile
echo "Parameter: $@" >> $testfile
echo "Systemname: $1" >> $testfile
echo "Emulatorcore: $2" >> $testfile
echo "ROM: $(basename "$3")" >> $testfile
[[ "$4" == "libretro" ]] && echo "You are free!" >> $testfile || echo "No hotkey is pure hell eh??" >> $testfile
```

- Method to not lose SRM and other meta-data

```bash
#!/bin/bash
# custom.sh - place to /userdata/system
# by cyperghost 23/11/19
#

if [[ $1 == stop ]]; then
    batocera-es-swissknife --emukill
fi
```

- Splash video during game load (x86/x86_64) (credit to @algernon)

```bash
#!/bin/bash
# Adds system specific loading screen videos to x86 builds of batocera via VLC
# Add your .mp4 videos to /userdata/loadingscreens naming them each with the same as in your roms folder i.e named snes.mp4 to play a video before for each snes game.
# Add this script in /userdata/system/scripts then chmod+x /userdata/system/scripts/videolaunchx86.sh in terminal to give this script permissions to run
```
do_videostart ()
{
    video="$1"
    vlc play --fullscreen --no-video-title-show --play-and-exit $video
}

# Comment out the above phrase and uncomment this one to enable playing
# loading videos concurrently with emulator launch instead of before
# (experimental)
#do_videostart ()
#{
#    video="$1"
#    vlc_opt="play --fullscreen --no-video-title-show --play-and-exit"
#    vlc $vlc_opt $video &
#    PID=$!
#}

videopath="/userdata/loadingscreens"

if [[ "$1" == "gameStart" && -n "$2" ]] ; then
    video="${videopath}/$2.mp4"
    [[ -f "$video" ]] || exit 1
else
    exit 1
fi

do_videostart "$video"
#wait $PID
exit 0

# Replace or comment out all the other code above except for the first
# bash line then uncomment the following phrase to enable a single
# loading splash for all systems. It should be named default.mp4 in the
# same folder as above.
#default="/userdata/loadingscreens/default.mp4"
#case $1 in
# gameStart)
#    vlc play --fullscreen --no-video-title-show --play-and-exit
#    $default
#    ;;
#    #
#    #esac
#    #
#    #exit 0

- Splash video during game load (Raspberry Pi)

videolaunchpi.sh
#!/bin/bash

do_videostart ()
{
  video="$1"
  # Launch the video
  omx_fnt="--font=/usr/share/fonts/dejavu/DejaVuSans-BoldOblique.ttf"
  omx_opt="--no-keys --layer=10000 --aspect-mode=fill"
  omx_srt="--no-ghost-box --lines=1 --align=left $omx_fnt --font-size=20 --subtitles=/usr/share/batocera/splash/splash.srt"
  
  # Disable sound
  omx_nosound="-n -1"
  # -911 = Volume set to 35% (On omxplayer)
  # 1/(10^(911/2000)) = 0.35034828830157
  
  /usr/bin/omxplayer -o both --vol -2500 $omx_opt $omx_srt
  $omx_nosound $video &
  PID=$!
}

videopath="/userdata/videoloadingscreens"

if [[ "$1" == "gameStart" && -n "$2" ]]; then
  video="${videopath}/$2.mp4"
  # Filecheck
  [[ -f "$video" ]] || exit 1
else
  exit 1
fi

do_videostart "$video"
#wait $PID
exit 0


• OGST Screen Support (original post)

OGST_screen.sh

#!/bin/bash
# Install videos of ZIP file to /userdata/OGST/
# Install scripts to /userdata/system/custom.sh and
# /userdata/system/scripts/OGST_screen.sh to use this script.

logo_folder="/userdata/OGST"
default_logo='start.mp4'
logo_file=""
loop=false
default_logo_loop=false
cache_file="/userdata/system/.cache/ffmpeg_PID'
clean_cache () {
  if [ -f $cache_file ]; then
    rm -f $cache_file
  fi
}

case "$1" in
  start)
    # init TFT screen
    [ `sbin/lsmod | grep -c spi_s3c64xx` -ge 1 ] && rmmod spi_s3c64xx
    modprobe spi_s3c64xx force32b=1
    modprobe fbtft_device name=hktft9340 busnum=1 rotate=270
    force32b=1
    # wait fb1 loaded
    N=0
    while ! test -e /dev/fb1 -o $N -gt 15; do
      sleep 1
      let N++
    done
    # rename fb1 to fb_ (trick to avoid conflicts with some emulators like reicast...)
    mv /dev/fb1 /dev/fb_
    clean_cache
    logo_file=$default_logo && loop=default_logo_loop
    ;;
  stop)
    clean_cache
    logo_file='stop.mp4' && loop=false
    ;;
  gameStop)
    logo_file=$default_logo && loop=default_logo_loop
    ;;
  gameStart)
    logo_file=$default_logo && loop=default_logo_loop
    case "$2" in
      3do) logo_file='3do/video.mp4' && loop=true
        ;;
      amiga) logo_file='amiga/video.mp4' && loop=true
        ;;
      amigacd32) logo_file='amigacd32/video.mp4' && loop=true
        ;;
      amstradcpc) logo_file='amstradcpc/video.mp4' && loop=true
        ;;
  esac
}
loop=true ;
apple2)  logo_file='apple2/video.mp4' &&
loop=true ;
atar1800) logo_file='atar1800/video.mp4' &&
loop=true ;
atar2600) logo_file='atar2600/video.mp4' &&
loop=true ;
atar15200) logo_file='atar15200/video.mp4' &&
loop=true ;
atar7800) logo_file='atar7800/video.mp4' &&
loop=true ;
atarist)  logo_file='atarist/video.mp4' &&
loop=true ;
atomiswave) logo_file='atomiswave/video.mp4' &&
loop=true ;
c64)      logo_file='c64/video.mp4' && loop=true
loop=true ;
cavestory) logo_file='cavestory/video.mp4' &&
loop=true ;
colecovision) logo_file='colecovision/video.mp4' &&
loop=true ;
dreamcast) logo_file='dreamcast/video.mp4' &&
loop=true ;
fba)      logo_file='fba/video.mp4' && loop=true
loop=true ;
fds)      logo_file='fds/video.mp4' && loop=true
loop=true ;
gameandwatch) logo_file='gameandwatch/video.mp4' &&
loop=true ;
gamegear) logo_file='gamegear/video.mp4' &&
loop=true ;
gb)       logo_file='gb/video.mp4' &&
loop=true ;
gba)      logo_file='gba/video.mp4' && loop=true
loop=true ;
gbc)      logo_file='gbc/video.mp4' && loop=true
loop=true ;
gx4000)   logo_file='gx4000/video.mp4' &&
& loop=true ;
intellivision) logo_file='intellivision/video.mp4'
loop=true ;
jaguar)   logo_file='jaguar/video.mp4' &&
loop=true ;
lynx)     logo_file='lynx/video.mp4' &&
loop=true ;
mame)     logo_file='mame/video.mp4' &&
loop=true ;
mastersystem) logo_file='mastersystem/video.mp4' &&
loop=true ;
megadrive) logo_file='megadrive/video.mp4' &&
msx) logo_file='msx/video.mp4' && loop=true
n64) logo_file='n64/video.mp4' && loop=true
naomi) logo_file='naomi/video.mp4' &&
loop=true ;; neogeo) logo_file='neogeo/video.mp4' &&
loop=true ;; neogeo) logo_file='neogeo/video.mp4' &&
loop=true ;; nes) logo_file='nes/video.mp4' && loop=true
;; ngp) logo_file='ngp/video.mp4' && loop=true
;; ngpc) logo_file='ngpc/video.mp4' &&
loop=true ;; o2em) logo_file='o2em/video.mp4' &&
loop=true ;; pcenginecd) logo_file='pcenginecd/video.mp4' &&
loop=true ;; pcengine) logo_file='pcengine/video.mp4' &&
loop=true ;; pcfx) logo_file='pcfx/video.mp4' &&
loop=true ;; pokemini) logo_file='pokemini/video.mp4' &&
loop=true ;; prboom) logo_file='prboom/video.mp4' &&
loop=true ;; psp) logo_file='psp/video.mp4' && loop=true
;; psx) logo_file='psx/video.mp4' && loop=true
;; satellaview) logo_file='satellaview/video.mp4' &&
loop=true ;; saturn) logo_file='saturn/video.mp4' &&
loop=true ;; scummvm) logo_file='scummvm/video.mp4' &&
loop=true ;; sega32x) logo_file='sega32x/video.mp4' &&
loop=true ;; segacd) logo_file='segacd/video.mp4' &&
loop=true ;; sg1000) logo_file='sg1000/video.mp4' &&
loop=true ;; snes) logo_file='snes/video.mp4' &&
loop=true ;; sufami) logo_file='sufami/video.mp4' &&
loop=true ;; supergrafx) logo_file='supergrafx/video.mp4' &&
loop=true ;; thomson) logo_file='thomson/video.mp4' &&
loop=true ;;
  vectrex)    logo_file='vectrex/video.mp4' &&
loop=true ;;
  virtualboy) logo_file='virtualboy/video.mp4' &&
loop=true ;;
  wswan) logo_file='wswan/video.mp4' &&
loop=true ;;
  wswanc) logo_file='atari800/video.mp4' &&
loop=true ;;
  x68000) logo_file='x68000/video.mp4' &&
loop=true ;;
  zx81) logo_file='zx81/video.mp4' &&
loop=true ;;
  zxspectrum) logo_file='zxspectrum/video.mp4' &&
loop=true ;;
  *)
    esac
;
esac

draw_logo () {
    loop=0
    if [ $3 = true ]; then loop=-1; fi
    dd if=/dev/zero of=/dev/fb_2>/dev/null >/dev/null
    ffmpeg -hide_banner -re -stream_loop $loop -i $1/$2 -c:v rawvideo -pix_fmt rgb565le -f fbdev /dev/fb_2>/dev/null >/dev/null &
    echo $! > $cache_file
}

# draw logo
if [ "$logo_file" != "" ]; then
  # kill last played video if running
  if [ -s $cache_file ]; then
    kill -9 $(cat $cache_file)
  fi
  # draw
  draw_logo $logo_folder $logo_file $loop
fi

exit 1

- Change LED colors while running games, and shutdown on button press

shutdownsimple.sh

#!/bin/bash
LED1=22
LED2=27
SHUTDOWN=3
case "$1" in
  start)
    # LED init
    echo "$LED1" > /sys/class/gpio/export
    echo "$LED2" > /sys/class/gpio/export
    echo out > /sys/class/gpio/gpio$LED1/direction
    echo 0 > /sys/class/gpio/gpio$LED1/value
    echo out > /sys/class/gpio/gpio$LED2/direction
    echo 1 > /sys/class/gpio/gpio$LED2/value

    # Button init
    echo "$SHUTDOWN" > /sys/class/gpio/export
    echo "in" > /sys/class/gpio/gpio$SHUTDOWN/direction
    # This loop continuously checks if the shutdown button was pressed
    # It sleeps as long as that has not happened.
    buttonstate1=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
    shutdownSignal=$(cat /sys/class/gpio/gpio$SHUTDOWN/value)
    while [ $shutdownSignal = $buttonstate1 ]; do
      sleep 0.5
      done
    shutdown -h now
  ;;
  stop)
    unexport all GPIOs
    #echo "$SHUTDOWN" > /sys/class/gpio/unexport
    echo "$LED1" > /sys/class/gpio/unexport
    echo "$LED2" > /sys/class/gpio/unexport
  ;;
esac
exit $?

- Load latest created savestate automatically

loadlatestsave.sh

#!/bin/bash

# disable auto load/save state inside ES
# add 'global.retroarch.savestate_auto_load=true'
# to your batocera.conf
# by cyperghost aka lala for BATOCERA
#
# download this script to '/userdata/system/scripts' and set executable bit
#
rom_no_ext="$(basename "${5%.*}")"
sav_path="/userdata/saves/$2"
EmulationStation scripts

Play a video on a second screen

Original forum post with a deeper explanation. Artwork needs to be sourced and placed in the appropriate Marquee and roms/Marquee folders first.

Place game.sh into system/configs/emulationstation/scripts/game-selected

```bash
#!/bin/bash
System=$1 #system name
Romname=${2%.*} #romname
rom=${Romname##*/}
/userdata/marquee.sh Gameselected $System "$rom"
```

Place system.sh into system/configs/emulationstation/scripts/system-selected

```bash
#!/bin/bash
System=$1 #System name
/userdata/marquee.sh Systemselected $System &
```

Place marquee.sh in /userdata

```
marquee.sh
```
#!/bin/bash

case $1 in
  Start)
    Romname=$3
    Gamepath=$2
    marqueeimage=$Gamepath/images/$romname-marquee.png
    if [ -f "/userdata/roms/Marquee/videos/$Romname.mp4" ];
      then ffmpeg -i /userdata/roms/Marquee/videos/$Romname.mp4 -vf scale=1280:720 -sws_flags bilinear -pix_fmt rgb565le -f fbdev /dev/fb0
    fi

    if [ -f "/userdata/roms/Marquee/hires/$Romname.jpg" ];
      then fnv /userdata/roms/Marquee/hires/$Romname.jpg -fer
    elif [ -f "$marqueeimage" ];
      then fnv $marqueeimage -fer
    else fnv /userdata/roms/Mame/images/mame.png -fer
    fi

  Gameselected)
    System=$2 #system name
    Romname=$3 #romname

    if [ -f "/userdata/roms/Marquee/$Romname.png" ];
      then fnv /userdata/roms/Marquee/$Romname.png -fer
    elif [ -f "/userdata/roms/$System/images/$Romname-marquee.png" ];
      then fnv "/userdata/roms/$System/images/$Romname-marquee.png" -fer
    else fnv /userdata/roms/Marquee/mame.png -fer
    fi

  Systemselected)
    imagepath="/userdata/roms/sysimages/$2"
    if [ -f "$imagepath.png" ];
      then fnv "$imagepath.png" -fer
    else fnv /userdata/roms/Mame/images/mame.png -fer
    fi

  ;;

#system name
Romname=$3 #romname

if [ -f "/userdata/roms/Marquee/$Romname.png" ];
  then fnv /userdata/roms/Marquee/$Romname.png -fer
elif [ -f "/userdata/roms/$System/images/$Romname-marquee.png" ];
  then fnv "/userdata/roms/$System/images/$Romname-marquee.png" -fer
else fnv /userdata/roms/Marquee/mame.png -fer
fi

;.;
Place `script.sh` in `system/scripts`

```bash
#!/bin/bash

case $1 in
  gameStart)
    gamepath=${5%/*}
    romname=${5##*/}
    /userdata/marquee.sh Start $gamepath ${romname%.*} &
    ;;

  gameStop)
    killall ffmpeg
    ;;

esac
```

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

Permanent link:
https://wiki.batocera.org/launch_a_script

Last update: 2022/02/20 07:44