Arch Linux as a lightweight audio platform

David Runge

Linux Audio Conference 2015

10.04.2015
What, Where, Why?

“Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]
What, Where, Why?

“Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]

https://www.archlinux.org
What, Where, Why?

- “Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]
- https://www.archlinux.org
- 36 developers, 40 trusted users, 8 support staff
What, Where, Why?

- “Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]
- https://www.archlinux.org
- 36 developers, 40 trusted users, 8 support staff
- ABS & AUR
What, Where, Why?

“Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]

- https://www.archlinux.org
- 36 developers, 40 trusted users, 8 support staff
- ABS & AUR
- pacman/ aura
What, Where, Why?

“Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]

- https://www.archlinux.org
- 36 developers, 40 trusted users, 8 support staff
- ABS & AUR
- pacman/ aura
- Archiso
“Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]

https://www.archlinux.org

36 developers, 40 trusted users, 8 support staff

ABS & AUR

pacman/ aura

Archiso

rolling release, systemd
What, Where, Why?

“Arch Linux exposes the user to the system without hiding any details.” [man 7 archlinux]

- https://www.archlinux.org
- 36 developers, 40 trusted users, 8 support staff
- ABS & AUR
- pacman/ aura
- Archiso
- rolling release, systemd
- lightweight, high customizability, easy build system
Preparations

Note: The following assumes you also configure, what you install!

- Choose supported hardware!!!
Preparations

**Note:** *The following assumes you also configure, what you install!*

- Choose supported hardware!!!
- Install Archlinux
Preparations

Note: The following assumes you also configure, what you install!

- Choose supported hardware!!!
- Install Archlinux
- Install cpupower
Preparations

Note: *The following assumes you also configure, what you install!*

- Choose supported hardware!!!
- Install Archlinux
- Install cpupower
- Install linux-rt
- Boot into realtime kernel
Preparations

Note: *The following assumes you also configure, what you install!*

- Choose supported hardware!!
- Install Archlinux
- Install cpupower
- Install linux-rt
- Install tuna & rt-tests
Preparations

Note: *The following assumes you also configure, what you install!*

- Choose supported hardware!!
- Install Archlinux
- Install cpupower
- Install linux-rt
- Install tuna & rt-tests
- Install rts & uenv

Choose supported hardware!!!

Install Archlinux

Install cpupower

Install linux-rt

Install tuna & rt-tests

Install rts & uenv
Preparations

Note: *The following assumes you also configure, what you install!*

- Choose supported hardware!!!
- Install Archlinux
- Install cpupower
- Install linux-rt
- Install tuna & rt-tests
- Install rts & uenv
- Install jack2
Note: The following assumes you also configure, what you install!

- Choose supported hardware!!!
- Install Archlinux
- Install cpupower
- Install linux-rt
- Install tuna & rt-tests
- Install rts & uenv
- Install jack2
- Install awesome
Note: The following assumes you also configure, what you install!

- Choose supported hardware!!!
- Install Archlinux
- Install cpupower
- Install linux-rt
- Install tuna & rt-tests
- Install rts & uenv
- Install jack2
- Install awesome
- Boot into realtime kernel
# Add your user to the audio group
```bash
gpasswd -a <username> audio
```
# Fully log out and log back in again

# Have a look at your hardware interrupts
```
cat /proc/interrupts
```
# Find the devices (and sometimes their parent devices),
# that you will use / need for your audio setup
# You can also use your audio card’s IRQ for the next test
# clone the realtimeconfigquickscan and run it
```bash
git clone https://github.com/raboof/realtimeconfigquickscan
cd realtimeconfigquickscan
./realtimeconfigquickscan
```
# Try and meet all criteria
# (except maybe the one for ‘fs.inotify.max_user_watches’)
Checks

- # Add your user to the audio group
  `gpasswd -a <username> audio`
  # Fully log out and log back in again

- # Have a look at your hardware interrupts
  `cat /proc/interrupts`
  # Find the devices (and sometimes their parent devices),
  # that you will use/need for your audio setup
  # You can also use your audio card’s IRQ for the next test
Checks

- # Add your user to the audio group
  gpasswd -a <username> audio
  # Fully log out and log back in again

- # Have a look at your hardware interrupts
  cat /proc/interrupts

  # Find the devices (and sometimes their parent devices),
  # that you will use/need for your audio setup
  # You can also use your audio card’s IRQ for the next test

- # clone the realtimeconfigquickscan and run it
  git clone https://github.com/raboof/realtimconfigquickscan
  cd realtimeconfigquickscan
  ./realtimeconfigquickscan
  # Try and meet all criteria
  # (except maybe the one for 'fs.inotify.max_user_watches')
Listing 1: Add systemd.setenv=REALTIME=true to your kernel command line

```plaintext
# syslinux: /boot/syslinux/syslinux.cfg
[..
LINUX ../vmlinuz-linux-rt
APPEND root=/dev/<your-root-fs> systemd.setenv=REALTIME=true rw
[..

# grub: /boot/grub/grub.cfg (or set stuff using /etc/default/grub)
[..
linux /boot/vmlinuz-linux-rt root=/dev/<your-root-fs> systemd.setenv=REALTIME=true rw
[..]
```
IRQs, cgroups & CPU settings

Figure: Sample tuna display of IRQs and kthreads on linux
IRQs, cgroups & CPU settings

Figure: Sample tuna display of IRQs and kthreads on linux-rt
Arch Linux as a lightweight audio platform

David Runge

Archlinux Installation Config Tests

Systemd Realtime Conditionals IRQ & kthread scheduling Awesome

Cyclic test and oscilloscope

Figure: Example Oscilloscope output (being fed by cyclic test)

Listing 2: Cyclic test feeding oscilloscope

```
# Start cyclic test (package: rt-tests) and feed it to oscilloscope (package: tuna)
cyclic test --smp -n -p99 -m -v | oscilloscope >/dev/null
```
systemd is a system and service manager for Linux operating systems. When run as first process on boot (as PID 1), it acts as init system that brings up and maintains userspace services. (man 1 init)
systemd is a system and service manager for Linux operating systems. When run as first process on boot (as PID 1), it acts as init system that brings up and maintains userspace services. (man 1 init)

systemctl start/stop/enable/disable *.{service,target,timer}
systemd is a system and service manager for Linux operating systems. When run as first process on boot (as PID 1), it acts as init system that brings up and maintains userspace services. (man 1 init)

- systemctl <start/stop/enable/disable> *.{service,target,timer}
- /run/systemd/system/*,
  /usr/lib/systemd/{system,user}/*,
  /etc/systemd/{system,user}/*
**systemd** is a system and service manager for Linux operating systems. When run as first process on boot (as PID 1), it acts as init system that brings up and maintains userspace services. (**man 1 init**)

**systemctl** `<start/stop/enable/disable> *.{service,target,timer}`

- `/run/systemd/system/*`,
  `/usr/lib/systemd/{system,user}/*`,
  `/etc/systemd/{system,user}/*`

- `/etc/systemd/journald.conf`,
  `/etc/systemd/logind.conf`,
  `/etc/systemd/system.conf`,
  `/etc/systemd/user.conf`
Realtime

- My Service Can't Get Realtime!

---

Let's ignore dbus as long as it is not kdbus

"[...]

When a process is in a cgroup in the "cpu" controller, and no RT budget is set for that cgroup, then RT is not available to it. This is very unfortunate. I'd love to assign an RT budget by default from systemd, but this isn't really doable, since there's no sane RT budget one could assign a cgroup given the current semantics of it (which require that all RT budgets of cgroups within another cgroup must sum up to less than 1/1...).

This is something that needs to be cleaned up in the kernel, and then we can expose this nicer in systemd.

For now, my recommendation would be to disable the RT cgroup stuff in the kernel, and thus forego the whole problem. [..."]

~Lennart Poettering

"[...]

anyone who wants to control cpu cgroups will be required to also become responsible for distributing RT scheduling."

~Tejun Heo

ControlGroup, ControlGroupAttribute directives removed in systemd > 205
Realtime

- My Service Can't Get Realtime!
- Let’s ignore dbus as long as it is not kdbus

"[...] When a process is in a cgroup in the "cpu" controller, and no RT budget is set for that cgroup, then RT is not available to it. This is very unfortunate. I'd love to assign an RT budget by default from systemd, but this isn’t really doable, since there’s no sane RT budget one could assign a cgroup given the current semantics of it (which require that all RT budgets of cgroups within another cgroup must sum up to less than 1/1...).

This is something that needs to be cleaned up in the kernel, and then we can expose this nicer in systemd. For now, my recommendation would be to disable the RT cgroup stuff in the kernel, and thus forego the whole problem. [...]"

~Lennart Poettering

"[...] anyone who wants to control cpu cgroups will be required to also become responsible for distributing RT scheduling." ~Tejun Heo

ControlGroup, ControlGroupAttribute directives removed in systemd > 205
Realtime

- **My Service Can’t Get Realtime!**

- Let’s ignore dbus as long as it is not kdbus

- “[…] When a process is in a cgroup in the “cpu” controller, and no RT budget is set for that cgroup, then RT is not available to it. This is very unfortunate. I’d love to assign an RT budget by default from systemd, but this isn’t really doable, since there’s no sane RT budget one could assign a cgroup given the current semantics of it (which require that all RT budgets of cgroups within another cgroup must sum up to less than 1/1…).

  This is something that needs to be cleaned up in the kernel, and then we can expose this nicer in systemd.

  For now, my recommendation would be to disable the RT cgroup stuff in the kernel, and thus forego the whole problem. […]” —Lennart Poettering
Realtime

- My Service Can't Get Realtime!

- Let’s ignore dbus as long as it is not kdbus

- “[…] When a process is in a cgroup in the “cpu” controller, and no RT budget is set for that cgroup, then RT is not available to it. This is very unfortunate. I’d love to assign an RT budget by default from systemd, but this isn’t really doable, since there’s no sane RT budget one could assign a cgroup given the current semantics of it (which require that all RT budgets of cgroups within another cgroup must sum up to less than 1/1…).

  This is something that needs to be cleaned up in the kernel, and then we can expose this nicer in systemd. For now, my recommendation would be to disable the RT cgroup stuff in the kernel, and thus forego the whole problem. […]“ ~Lennart Poettering

- “[…] anyone who wants to control cpu cgroups will be required to also become responsible for distributing RT scheduling.” ~Tejun Heo
Realtime

- My Service Can't Get Realtime!

- Let’s ignore dbus as long as it is not kdbus

- “[…] When a process is in a cgroup in the “cpu” controller, and no RT budget is set for that cgroup, then RT is not available to it. This is very unfortunate. I’d love to assign an RT budget by default from systemd, but this isn’t really doable, since there’s no sane RT budget one could assign a cgroup given the current semantics of it (which require that all RT budgets of cgroups within another cgroup must sum up to less than 1/1…).

  This is something that needs to be cleaned up in the kernel, and then we can expose this nicer in systemd. For now, my recommendation would be to disable the RT cgroup stuff in the kernel, and thus forego the whole problem. […]“ ~Lennart Poettering

- “[…] anyone who wants to control cpu cgroups will be required to also become responsible for distributing RT scheduling.” ~Tejun Heo

- ControlGroup, ControlGroupAttribute directives removed in systemd > 205
man 5 systemd.exec

[...]

**IOSchedulingClass** = Sets the IO scheduling class for executed processes. Takes an integer between 0 and 3 or one of the strings none, realtime, best-effort or idle. See ioprio_set(2) for details.

**IOSchedulingPriority** = Sets the IO scheduling priority for executed processes. Takes an integer between 0 (highest priority) and 7 (lowest priority). The available priorities depend on the selected IO scheduling class (see above). See ioprio_set(2) for details.

**CPUSchedulingPolicy** = Sets the CPU scheduling policy for executed processes. Takes one of other, batch, idle, fifo or rr. See sched_setscheduler(2) for details.

**CPUSchedulingPriority** = Sets the CPU scheduling priority for executed processes. The available priority range depends on the selected CPU scheduling policy (see above). For real-time scheduling policies an integer between 1 (lowest priority) and 99 (highest priority) can be used. See sched_setscheduler(2) for details.

**CPUSchedulingResetOnFork** = Takes a boolean argument. If true, elevated CPU scheduling priorities and policies will be reset when the executed processes fork, and can hence not leak into child processes. See sched_setscheduler(2) for details. Defaultstofalse.

**CPUAffinity** = Controls the CPU affinity of the executed processes. Takes a space-separated list of CPU indices. This option may be specified more than once in which case the specified CPU affinity masks are merged. If the empty string is assigned, the mask is reset, all assignments prior to this will have no effect. See sched_setaffinity(2) for details.

[...]
Custom JACK systemd --user service

**Listing 3:** /etc/conf.d/fw1

```bash
# Sample configuration file for a JACK systemd --user service, using a firewire device
NAME="default"
DRIVER="firewire"
DEVICE="/dev/fw1"
NOMLOCK=""
REALTIME="-R"
PORTMAX=512
UNLOCK="-u"
VERBOSE="-v"
DRIVER_SETTINGS="\n   -n 3 \n   -p 256\n   -r 48000"
```

**Listing 4:** /usr/lib/systemd/user/jack@.service

```ini
[Unit]
Description=JACK Audio
After=sound.target local-fs.target

[Service]
EnvironmentFile=/etc/conf.d/%i
EnvironmentFile=%h/.config/jack/%i
ExecStart=/usr/bin/jackd -n $NAME $REALTIME -p $PORTMAX -d $DRIVER -d $DEVICE $DRIVER_SETTINGS
CPUSchedulingPolicy=rr
CPUSchedulingPriority=70
LimitRTPRIO=71
LimitRTTIME=-1

[Install]
WantedBy=default.target
```
Conditional cpupower (cpupower-rt)

Listing 5: /usr/lib/systemd/system/cpupower-rt.service (package: uenv)

[Unit]
Description=Apply cpupower configuration
ConditionKernelCommandLine=systemd.setenv=REALTIME=true
After=cpupower.service

[Service]
Type=oneshot
ExecStart=/usr/lib/systemd/systemd/scripts/cpupower-rt
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
Listing 6: /etc/default/cpupower-rt

# Define CPUs governor
# valid governors: ondemand, performance, powersave, conservative, userspace.
governor='performance'

# Limit frequency range
# Valid suffixes: Hz, kHz (default), MHz, GHz, THz
#min_freq="2.25 GHz"
#max_freq="3.4 GHz"

# Specific frequency to be set.
# Requires userspace governor to be available.
# Do not set governor field if you use this one.
#freq=

# Utilizes cores in one processor package/socket first before processes are
# scheduled to other processor packages/sockets.
# See man (1) CPUPOWER-SET for additional details.
#mc_scheduler=

# Utilizes thread siblings of one processor core first before processes are
# scheduled to other cores. See man (1) CPUPOWER-SET for additional details.
#smp_scheduler=

# Sets a register on supported Intel processors which allows software to convey
# its policy for the relative importance of performance versus energy savings to
# the processor. See man (1) CPUPOWER-SET for additional details.
perf_bias=0

# vim:set ts=2 sw=2 ft=sh et:
Conditional Compositing

List 7: /usr/lib/systemd/user/compton.service

[Unit]
Description=Compton X Compositor
After=display-manager.service local-fs.target
ConditionFileIsExecutable=/usr/bin/compton
ConditionKernelCommandLine=!systemd.setenv=REALTIME=true

[Service]
Type=forking
ExecStart=/usr/bin/compton --config %h/.config/compton.conf -b
Restart=always

[Install]
WantedBy=default.target
Tuna scripting

With the help of **tuna** and **rts** you can create a customized IRQ and kthread scheduling setup!

**Listing 8:** /etc/rts/examples/w540-expresscard-firewire

```bash
# IRQ scheduling

# set priority for hardware clock rtc0
tuna -q 8 -p 90
# set priority for device with irq 18 (firewire_ohci, i801_smbus)
tuna -q 18 -p 85

# kthread scheduling
tuna -t 'irq/*/rtc0*' -p 90
tuna -t 'irq/*/i801_smb*' -p 88
tuna -t 'irq/*/firewire*' -p 86
```

**Listing 9:** /usr/lib/systemd/system/rts@.service

```iniset
[Unit]
Description=Apply Realtime-Kernel specific settings
After=multi-user.target sound.target
ConditionKernelCommandLine=systemd.setenv=REALTIME=true
ConditionFileIsExecutable=/usr/bin/tuna

[Service]
Type=oneshot
ExecStart=/usr/lib/systemd/scripts/rts %i
RemainAfterExit=true

[Install]
WantedBy=multi-user.target
```
“awesome is a highly configurable, next generation 
framework window manager for X. It is very fast, 
extensible and licensed under the GNU GPLv2 license.”
Awesome

“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
- Many themes available
“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
- Many themes available
- Auto-tiling!
Awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
- Many themes available
- Auto-tiling!
- Integrates well with Desktop/Session Managers

“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”
“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
- Many themes available
- Auto-tiling!
- Integrates well with Desktop/Session Managers
- Extensible via widgets
“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
- Many themes available
- Auto-tiling!
- Integrates well with Desktop/Session Managers
- Extensible via widgets
- Well documented, vast wiki and a helpful community
“awesome is a highly configurable, next generation framework window manager for X. It is very fast, extensible and licensed under the GNU GPLv2 license.”

- Using LUA, versatile and complex setups are possible
- Being keyboard-based this might or might not be what you want
- Many themes available
- Auto-tiling!
- Integrates well with Desktop/Session Managers
- Extensible via widgets
- Well documented, vast wiki and a helpful community
- Suited for work with lightweight and realtime environments
Questions, suggestions, blame?

Mail: dave@sleepmap.de
XMPP: dvzrv@sleepmap.de
IRC: dvzrv@{efnet,freenode,oftc}
References

**Julien Danjou.**
*Awesome window manager*, 2015.

**FreeDesktop Foundation.**
*Systemd Index*, 2015.

**Judd Vinet & Aaron Griffin.**
*Arch Linux*, 2015.
URL [https://www.archlinux.org](https://www.archlinux.org).

**JACK Audio Connection Kit.**
*JACK Audio Connection Kit*, 2015.

**Libre Music Production.**
*Libre Music Production*, 2015.

**David Runge.**
*rts*, 2015.
URL [http://sleepmap.de/projects/rts/](http://sleepmap.de/projects/rts/).

**David Runge.**
*uenv*, 2015.

**Lana Brindley & Alison Young.**