Chino
scripted meta-applications for Linux audio

David Adler

Linux Audio Conference 2013
Modular sessions

Some programs, interconnected via:

- Jack audio
- Jack Midi
- Alsa Midi
Connection graphs

Assume a system with

\[ P \] number of port-types, each having
\[ O_p \] output ports and
\[ I_p \] input ports.

The number of all possible connection graphs \( \Omega \) then is

\[ \Omega = 2^{\sum_{p=1}^{P} O_p I_p}. \]
Connection graphs

\[
P = 2 \quad \text{two port types (audio and Midi)}
\]
\[
O_{\text{audio}} = 60 \quad \text{audio out}
\]
\[
I_{\text{audio}} = 40 \quad \text{audio in}
\]
\[
O_{\text{Midi}} = 10 \quad \text{Midi out}
\]
\[
I_{\text{Midi}} = 24 \quad \text{Midi in}
\]
\[
\Omega = 2^{60 \times 40 + 10 \times 24}
\]
\[
\approx 5.2383 \times 10^{794}
\]
Session management... is desirable.
Once upon a time [...] 

**LADCCA (2003)**  
Linux Audio Developers Configuration and Connection API  
[https://savannah.nongnu.org/projects/ladcca](https://savannah.nongnu.org/projects/ladcca)

superseded by

**LASH (2005)**  
Linux Audio Session Handler  
[https://savannah.nongnu.org/projects/lash](https://savannah.nongnu.org/projects/lash)

which is still alive.
In QJackCtl, non-compliant applications can be added as “Infra-clients”.

Frontends:
- pyjacksm
- QjackCtl
- and ...
LADISH

Linux Audio Desktop Integration Session Handler
http://ladish.org

LASH’s successor, incorporates both LASH and JACK Session.

Frontend:
  ▶ patchage

Non-compliant applications can be added as “Level 0-clients”.

Non-compliant applications can be added as “Level 0-clients”.
Applications not supporting the protocol can be added as “NSM Proxy-clients”.

http://non.tuxfamily.org/wiki/Non%20Session%20Manager
## Chino vs. the others

<table>
<thead>
<tr>
<th></th>
<th>others</th>
<th>Chino</th>
</tr>
</thead>
<tbody>
<tr>
<td>any connection graph</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>any application</td>
<td>no(^1)</td>
<td>yes(^2)</td>
</tr>
<tr>
<td>central saving point</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>preparation required</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>GUI</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

\(^1\) LADISH, Qjackctl and NSM allow to include non-compliant applications.

\(^2\) If the application is sufficiently scriptable.
Concepts
Presets and sessions

Chino does nothing,
the preset does it all,
the preset is just another session.

the meta-application
"patch files" of the meta-application

session A  session B  session C
Templates, inheritance of files
Applications

A preset defines a number of *applications*.

An application—in Chino—consists of

- a program
- the way it’s used, defined by
  - *application files* – patch files, configuration files, dirs, etc.
  - an *application library* – a textfile (Bash) defining how the program gets started and interconnected.
Methods

Categories of similar applications, defined via method libraries.

Method types:

**unique methods**
- jnoise
- seq24
- qmidiarp
- ardour

**channel methods**
- ch 9
- ch 8 yoshimi
- ch 7 ams
- ch 6
- ch 5 ams
- ch 4 yoshimi
- ch 3 yoshimi
- ch 2
- ch 1
Method types, methods and applications

hard-coded (in Chino)  user-defined (via preset)

channel  instr  yosh
          effect  ams
          ...  ...

unique  misc  yosh
          seq24

method types  methods  applications
Method IDs and application IDs

hard-coded (in Chino) | user-defined (via preset)

channel

ch_instrument

ch_effect

chef_reverb

unique

method types | methods | applications

hard-coded
user-defined

channel

ch_instrument

ch_effect

chef_reverb

unique

method types | methods | applications

hard-coded
user-defined
Application files

are kept in `<application_ID>` directories below the session’s base directory.

Presets must contain all application files.

Sessions will contain the ones they (ever) require(d).
Libraries

are kept in a *libs* directory below the session’s base directory.

Presets must contain all libraries.

Sessions may optionally hold libraries.

A "root-library", called `<session_name>-listlib`, holds a list of all allowed methods and applications.
Inheritance of libraries
Dependencies

program + use case $\Rightarrow$ dependencies
Dependencies
Dependencies
Dependencies – port-groups
Dependencies – port-groups
Dependencies – ambiguous provides
Dependencies – ambiguous provides
Dependencies – methods
Dependencies – nodes and anchors

<audio provides> <Midi provides>

<node ID>

<audio depends> <Midi depends>

STEREO
uqhw_stereo

KBD
uqhw_uf6

CC
uqhw_nano
STEREO -
uqhw_stereo
- -
MIXER REC EFFBUS -
uqms_ardour2
STEREO :CC
- KBD
uqhw_uf6
- -
- SEQ
uqms_seq24
- KBD
- CC
uqhw_nano
- -
SYNTH SYNTH
ch_synth
MIXER :REC :SEQ :MDR
- -
002-chsy_aeolus
- -
- -
003-chsy_synthv1
- -
Implementation
Steps

- assign
- check
- list
- copy
- start
- acn
- mcn
- the “tweak-and-save loop” (not a step)
- unassign or kill
A *Task* is a series of steps accomplishing something useful... 

<table>
<thead>
<tr>
<th>step</th>
<th>application</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>assign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>check</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>list</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>copy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>acn</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mcn</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unassign</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kill</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tasks

...like restarting an application...

<table>
<thead>
<tr>
<th>step</th>
<th>application</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign</td>
<td></td>
<td></td>
<td></td>
<td>s2</td>
<td></td>
</tr>
<tr>
<td>check</td>
<td></td>
<td></td>
<td></td>
<td>s3</td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
<td>s4</td>
<td></td>
</tr>
<tr>
<td>copy</td>
<td></td>
<td></td>
<td></td>
<td>s5</td>
<td></td>
</tr>
<tr>
<td>start</td>
<td></td>
<td></td>
<td></td>
<td>s6</td>
<td></td>
</tr>
<tr>
<td>acn</td>
<td></td>
<td></td>
<td></td>
<td>s7</td>
<td></td>
</tr>
<tr>
<td>mcn</td>
<td></td>
<td></td>
<td></td>
<td>s8</td>
<td></td>
</tr>
<tr>
<td>unassign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>s1</td>
</tr>
<tr>
<td>kill</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tasks

...re-establishing all audio connections...

<table>
<thead>
<tr>
<th>step</th>
<th>application</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>check</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>copy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acn</td>
<td>s1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mcn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unassign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kill</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tasks

...or starting an entire session.

<table>
<thead>
<tr>
<th>step</th>
<th>application</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
</tr>
</thead>
<tbody>
<tr>
<td>assign</td>
<td></td>
<td>s1</td>
<td>s2</td>
<td>s3</td>
<td>s4</td>
</tr>
<tr>
<td>check</td>
<td></td>
<td>s5</td>
<td>s6</td>
<td>s7</td>
<td>s8</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>s9</td>
<td>s10</td>
<td>s11</td>
<td>s12</td>
</tr>
<tr>
<td>copy</td>
<td></td>
<td>s13</td>
<td>s14</td>
<td>s15</td>
<td>s16</td>
</tr>
<tr>
<td>start</td>
<td></td>
<td>s17</td>
<td>s18</td>
<td>s19</td>
<td>s20</td>
</tr>
<tr>
<td>acn</td>
<td></td>
<td>s21</td>
<td>s22</td>
<td>s23</td>
<td>s24</td>
</tr>
<tr>
<td>mcn</td>
<td></td>
<td>s25</td>
<td>s26</td>
<td>s27</td>
<td>s28</td>
</tr>
<tr>
<td>unassign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kill</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>method type</td>
<td>unique methods</td>
<td>channel methods</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>method ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>uq_aa</td>
<td>uq_bb</td>
<td>ch_aa</td>
<td>ch_bb</td>
<td></td>
</tr>
<tr>
<td>appl. step</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>assign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>check</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>list</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>copy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>start</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mcn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unassign</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kill</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table shows the method types and their corresponding unique and channel methods. The 'appl. step' column lists the steps 'a1', 'f', 'a2', 'a3', 'a4', 'f', 'a5', 'a6', 'f', 'a7', 'a8', 'a9', and 'f'.
<table>
<thead>
<tr>
<th>method type</th>
<th>unique methods</th>
<th>channel methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>method ID</td>
<td>uq_aa</td>
<td>uq_bb</td>
</tr>
<tr>
<td>appl. step</td>
<td>a1 f a2 a3 a4</td>
<td>f a5 a6</td>
</tr>
</tbody>
</table>

assign

check

list

copy

start

acn s1 s2 s3 s4 | s5 | s6 | s7 | s8 | s9 |
| f | f | f | f | f |

mcn

unassign

kill
<table>
<thead>
<tr>
<th>method type</th>
<th>unique methods</th>
<th>channel methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>method ID</td>
<td>uq_aa</td>
<td>uq_bb</td>
</tr>
<tr>
<td>appl. step</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a1 f</td>
<td>a2 a3 a4 f</td>
</tr>
<tr>
<td>assign</td>
<td>s2</td>
<td></td>
</tr>
<tr>
<td>check</td>
<td>s3</td>
<td>f</td>
</tr>
<tr>
<td>list</td>
<td>s4</td>
<td>f</td>
</tr>
<tr>
<td>copy</td>
<td>s5</td>
<td>f</td>
</tr>
<tr>
<td>start</td>
<td>s6</td>
<td>f</td>
</tr>
<tr>
<td>acn</td>
<td>s7</td>
<td>f</td>
</tr>
<tr>
<td>mcn</td>
<td>s8</td>
<td>f</td>
</tr>
<tr>
<td>unassign</td>
<td>s1</td>
<td>f</td>
</tr>
<tr>
<td>kill</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Step functions

Chino executes a step by calling a corresponding function ("step function") from the method library.
Step functions

# Step-functions in a method library.
#
# Naming:
# s_<method ID>_<step name>

# the assign step
s_ch_synth_assign()
{
    # some stuff
}

# the check step
s_ch_synth_check()
{
    # some stuff
}
Steps – helper functions

Inside a method’s step function, usually a *helper function* is called that will accomplish the step in a standardised way.
Steps – usage of helper functions

# Helper-functions, prefixed h_, called by
# the corresponding step-functions.
#
# Naming:
# h_<step name>

s_uq_msc_check()
{
    declare -ri i=$1
    COPY_uq_msc[i]="$(h_check uq_msc $i)"
    echo "== COPY_uq_msc[$i]  ${COPY_uq_msc[i]}"
}

s_uq_msc_list()
{
    declare -ri i=$1
    h_list uq_msc $i
}
h_assign() and h_copy()

h_assign() sources the application library, thereby implementing inheritance rules for libraries.

h_copy() copies the application files, thereby implementing inheritance rules for application files.

If the application requires it, h_copy() will call <application_ID>_move().
h_start() calls:

- a function `<application_ID>_start()` from the application library to start the program;
- *Assign functions* in which we must assign ports to variables, in accordance with the method’s and application’s depends and provides.


h_start()

method library

s_start_<meth_ID>()

calls

h_start()

reads variables, takes ALSA/JACK port snapshots accordingly

starts application

waits for ports

another snapshot, delivers new ports to assign-functions

reads

<app_ID>_start()

calls

calls

calls

calls

<app_ID>_am_assign()

<app_ID>_ja_assign()

ADEP_<meth_ID>='FOO BAR'
APRO_<meth_ID>=
MDEP_<meth_ID>='FOO'
MPRO_<meth_ID>=

reads

ADEP_<app_ID>=
APRO_<app_ID>=
MDEP_<app_ID>='BAZ'
MPRO_<app_ID>=

reads

MIDI_<app_ID>='ALSA'

reads
h_start() – assign-functions

APRO_CH_effect=’EFFECT’
APRO_CH_effect=’EFFBUS’

# The method ch_effect has audio depends/Provides,
# so an assign-function for audio is required.
ch_effect_ja_assign()
{
    declare -ri chan=$1
    declare -r inports=$2
    declare -r outports=$3

    EFFECT_AIN_L[chan]=$(echo "$inports" | sed -n 1p)
    EFFECT_AIN_R[chan]=$(echo "$inports" | sed -n 2p)
    EFFECT_AOUT_L[chan]=$(echo "$outports" | sed -n 1p)
    EFFECT_AOUT_R[chan]=$(echo "$outports" | sed -n 2p)

    echo "== EFFECT_AIN_L[chan] ${EFFECT_AIN_L[chan]}"
    echo "== EFFECT_AIN_R[chan] ${EFFECT_AIN_R[chan]}"
    echo "== EFFECT_AOUT_L[chan] ${EFFECT_AOUT_L[chan]}"
    echo "== EFFECT_AOUT_R[chan] ${EFFECT_AOUT_R[chan]}"
}

h_acn() and h_mcn() call connect-functions in which we must establish connections, one for each of the method’s and application’s depends.
h_acn()

method library
s_mcn_<meth_ID>()

ADEP_<meth_ID>='FOO BAR'

<meth_ID>_acn_FOO()
reads
depends
calls
connect-functions accordingly

ADEP_<app_ID>=
# The method ch_effect depends on EFFBUS, so a corresponding audio connect-function required.

```bash
ch_effect_acn_EFFBUS()
{
    declare -ri chan=$1
    # inputs from send ports
    declare -i i=0
    while [ $i -lt $MIXER_NCHAN ]; do
        msaudioconnect EFFBUS_SEND "$chan,$i" EFFECT_AIN $chan
        (( i++ ))
    done
    # outputs to return ports
    msaudioconnect EFFECT_AOUT $chan EFFBUS_RETURN $chan
}
```

ADEF_ch_effect='EFFBUS'

h_acn() – connect-functions
msaudioconnect HW_AOUT 0 MX_AIN 14

will, depending on existence of port-variables, connect:

\[
\begin{align*}
\{\text{HW}_A\text{OUT}_L\} & \quad \{\text{MX}_A\text{IN}_R[14]\} \\
\{\text{HW}_A\text{OUT}_R\} & \quad \{\text{MX}_A\text{IN}_L[14]\}
\end{align*}
\]

\[
\begin{align*}
\{\text{HW}_A\text{OUT}\} & \quad \{\text{MX}_A\text{IN}_R[14]\} \\
& \quad \{\text{MX}_A\text{IN}_L[14]\}
\end{align*}
\]

\[
\begin{align*}
\{\text{HW}_A\text{OUT}_L\} & \quad \{\text{MX}_A\text{IN}_R[14]\} \\
\{\text{MX}_A\text{IN}_R\} & \quad \{\text{MX}_A\text{IN}[14]\}
\end{align*}
\]

\[
\begin{align*}
\{\text{HW}_A\text{OUT}\} & \quad \{\text{MX}_A\text{IN}[14]\}
\end{align*}
\]

\[
\begin{align*}
\{\text{HW}_A\text{OUT}_L\} & \quad \{\text{MX}_A\text{IN}_R[14]\}
\end{align*}
\]
h_mcn()
h_mcn() – connect-functions

MDEP_chef_jackrack='CC'

# The application chef_jackrack depends on CC, so a
# corresponding Midi connect-function is required.
chef_jackrack_mcn_CC()
{
    declare -ri chan=$1
    ajmidiconnect CC_MOUT 0 chef_jackrack_MIN $chan
}
ajmidiconnect SEQ_MOUT 4 SYNTH_MIN 4

will, depending on existence of port-variables, connect:

$\{SEQ\_MOUT\_J[4]\} \cdot \{SEQ\_MOUT\_A[4]\} \cdot \{SYNTH\_MIN\_J[4]\} \cdot \{SYNTH\_MIN\_A[4]\}$

$\{SEQ\_MOUT\_J[4]\} \cdot \{SEQ\_MOUT\_A[4]\} \cdot \{SEQ\_MOUT\_J[4]\} \cdot \{SYNTH\_MIN\_A[4]\}$

$\{SEQ\_MOUT\_J[4]\} \cdot \{SEQ\_MOUT\_A[4]\} \cdot \{SYNTH\_MIN\_J[4]\} \cdot \{SYNTH\_MIN\_A[4]\}$

$\{SEQ\_MOUT\_J[4]\} \cdot \{SEQ\_MOUT\_A[4]\} \cdot \{SYNTH\_MIN\_J[4]\} \cdot \{SYNTH\_MIN\_A[4]\}$

a2jmidid $\{SEQ\_MOUT\_J[4]\} \cdot \{SEQ\_MOUT\_A[4]\} \cdot \{SYNTH\_MIN\_J[4]\} \cdot \{SYNTH\_MIN\_A[4]\}$
Keeping the graph intact

Connect-functions aren’t exclusively called by `h_acn()` and `h_mcn()`.

E.g. the task for starting an application will call connect-functions of nodes depending on any newly provided port-group.
User interface

- command line options and arguments
- runtime user interface
- the session definition file
User interface – command line

[foo@bar ~]$ chino -h

chino -h
chino -v
chino -w [-p <ppath>:{pname} -t <tpath>:{tname}]
chino -m [-p <ppath>:{pname} -t <tpath>:{tname}]
chino -a [-p <ppath>:{pname} -t <tpath>:{tname}]
chino -n name [-p <ppath>:{pname} -t <tpath>:{tname}]
chino -o sdef

-h display this help text
-v print version number
-w write session definition file prototype
-m create new method library
-a create new application library
-n <name> start new session with name <name>
-o <sdef> open session using session definition file <sdef>
-p <ppath>:{pname} Use preset <pname> in directory <ppath> as preset, overriding the default from ~/.chinorc.
-t <tpath>:{tname} Use session <tname> in directory <tpath> as template. If not specified, the preset will serve as template.
User interface - at runtime

M - add a method
A - add an application
X - remove an application
R - restart an application

. - localise application/method library
: - localise all used libraries
f - force sourcing of application/method library

l - list current session
d - check dependency tree
g - toggle session graph display
w - write changes to session definition file

a - redo audio connections
m - redo midi connections

s - store connection snapshot
r - restore connection snapshot
u - un-store connection snapshot

ctrl+c - quit
User interface - session definition file

NAME=opus2
PRESET=/usr/share/chino/presets/default:default
TMPL=

# methods and applications
UQMETHS=hw msc meter
uq_hw=stereo
uq_msc=ardour2 vkeybd nonseq
uq_meter=jkmeter japa

CHMETHS=synth
ch_synth-CH-001=yoshimi
ch_synth-CH-002=
ch_synth-CH-003=
ch_synth-CH-004=
ch_synth-CH-005=
ch_synth-CH-006=
ch_synth-CH-007=
ch_synth-CH-008=
Miscellaneous
Over the network? ssh!

Portability

- self-contained or shared preset
- program versions
- hardware requirements
- sampling rates
- local configuration files
- external (audio) files

The Future

- application groups
- guided fixing of unresolved depends
- have a preset determine window placement for applications—via scriptable WM
- text-only sessions in git
Thanks to

- LAD, LAU and the wider Linux audio community, for all that software and the reading matter they provide.
Demo
Q&A

http://chino.tuxfamily.org

david.jo.adler@gmail.com