050.128
Interdisziplinäres Didaktikpraktikum WS 2005/2006
Portierung der MIDI-Befehle zwischen zwei Logo-Implementationen
Fächer: Informatik, Musikerziehung

Alexander Ölzant
9301547

24. Januar 2006

Inhaltsverzeichnis

Kapitel 1

Einleitung

Für die Lehrveranstaltung „Spezielle Kapitel der Schulinformatik“ bei Prof. Neuwirth ist die Verwendung des an und für sich freien Logo-Dialekts MSW-Logo vorgesehen, die jedoch nur für eine spezifische Plattform verfügbar ist. Die Didaktischen Konzepte der Lehrveranstaltung bestehen in Verstehen und Begreifen einer funktionalen Programmiersprache und erweiterter Datenstrukturen mittels der Programmierung einer Melodie- bzw Geräuschstruktur und Abspielen dieser über die MIDI-Funktionen der Programmiersprache, um diese nun auch auf freien Betriebssystemen nutzen zu können, wurde eine Möglichkeit gesucht, die MIDI-Ausgabe auch unter GNU/Linux zu ermöglichen.

Eine Möglichkeit wäre hier gewesen, über den Emulator Wine die Midischnittstellen des Host-Systems mit MSWLogo zu nutzen, was jedoch umständlich und unelegant gewesen wäre und auch nicht auf allen CPU-Plattformen funktioniert.

Die Implementation erfolgte hier mittels Ansteuerung über ALSA-Sequencer, tatsächlich ist die gewählte Möglichkeit eine von mehreren Varianten, timidity asynchron anzusprechen; der Vorteil ist, dass ein beliebiges Midi-Device (Software-Synthesizer, Keyboard, ...) angeschlossen werden kann, der Nachteil liegt auf der Hand: durch die Beschränkung auf die ALSA-Architektur ist die Verwendungsmöglichkeit derzeit auf GNU/Linux eingeengt, eine Verwendung auf anderen Plattformen bedingt eine Portierung auf z. B. timidity-eigene Interfaces.

Der patch wurde auf x86 und ppc getestet, sollte also eigentlich auf allen Linux-Plattformen (Big Endian/Little Endian) funktionieren, allerdings wurde keine 64-bit-Plattform getestet.

Da der Rest der Dokumentation inhaltlich bereits als README vorlag, wurde dieses per cut and paste importiert.

Kapitel 2

Zeitplan

Die Implementierung erfolgte innerhalb von drei Wochen, wobei wie unten beschrieben der Grossteil der Zeit dafür verwendet wurde, die ALSA-Sequencer-Schnittstelle in ausreichendem Masse zu verstehen.

Kapitel 3

Hauptteil (README-Übersetzung)

3.1 Copyright, Autor

Dieser Patch steht wie die Ursprungsprogramme unter der GNU GPL Version 2 oder einer späteren Version, siehe `http://www.gnu.org/copyleft/gpl.html` für weitere Informationen. Im Verweisteil sind auch die originalen Versionen von UCB Logo und MSWLogo zu finden.

Alexander Oelzant <alexander@oelzant.priv.at>

3.2 Installation

Die Verteilung erfolgt als Debian GNU/Linux Sarge Pakete, der Patch sollte mit jeder Version von UCBLogo funktionieren. Wenn Du das Paket selbst kompilieren möchtest, achte auf die Verfügbarkeit von libasound2 und den asound2 EntwicklerInnenpaketen (diese stellen unter anderem das Verzeichnis `/usr/include/alsa` und die dort befindlichen Header-Dateien zur Verfügung).

Es besteht sowohl die Möglichkeit, das Debian-Paket selbst zu kompilieren mittels einer Kommandozeile ähnlich `chmod a+x debian/rules; dpkg-buildpackage -rfakeroot or dpkg-source -x ucblogo_5.5-1.dsc; cd ucblogo-5.5; dpkg-buildpackage`), oder es in der üblichen Art „von Hand“ zu installieren:

```
> curl http://www.cs.berkeley.edu/~bh/downloads/ucblogo.tar.gz | tar -xzv
or
> curl http://tigerente.htu.tuwien.ac.at/~aoe/mystuff/ucblogo_midi/ucblogo_5.5.orig
> mv ucblogo ucblogo-5.5
> cd ucblogo-5.5
> curl http://tigerente.htu.tuwien.ac.at/~aoe/mystuff/ucblogo_midi/ucblogo_5.5-1.di
> ./configure
```

```
> make
# make install
```

Laut Voreinstellungen werden die Midi-Fähigkeiten eingebaut, beim `configure` sollte also die folgende Zeile in der Ausgabe aufscheinen:

```
checking for snd_seq_event_output in -lasound... yes
```

Wenn dieser Test negativ ausfällt, siehe oben - vermutlich fehlen die ALSA (Advanced Linux Sound Architecture) dynamischen Laufzeitbibliotheken oder die zum Neubauen notwendigen EntwicklerInnenbibliotheken nd Header-Dateien aus dem entsprechenden `-dev`-Paket.

Eine andere Möglichkeit wäre eine alte ALSA-Installation, die es notwendig macht <sys/asoundlib.h> anstatt von <alsa/asoundlib.h> einzubinden, wofür ich leider noch keinen `configure`-Test habe, es wäre in diesem Falle also notwendig, in der Datei `mmwind.c` '1||' am Anfang zu entfernen.

Für die Dokumentation ist auch LaTeX notwendig, um jedoch nur die ausführbare Datei zu erhalten, kann darauf verzichtet und der auftretende Fehler bei der Ertstellung der Dokumentation ignoriert werden, in diesem Falle kann direkt zur Installationsphase weitergegangen werden.

Wenn alle anderen Mittel versagen, kann noch `configure` mit `autoconf` neu gebaut werden, indem `autoreconf` aufgerufen wird.

Wenn Du die gepatchte Version ausprobieren, aber nicht Deine gesamte Installation dafür riskieren willst, ist es hierbei allerdings notwendig, `./logo` mitzuteilen, wo es seine Logo-Bibliotheken findet, in denen einige der Standardbefehle wie `foreach`, `, `while` etc definiert sind. Das kann entweder geschehen, indem make eine `LIBLOC` übergeben wird (siehe auch die `debian/rules` Datei)

```
> make LIBLOC=/usr/share/ucblogo
```

oder indem `./configure` das bereits beim Start mitgeteilt wird:

```
> ./configure   --libdir=/usr/share/ucblogo
```

Andernfalls können Beschwerden über das Fehlen von Standardbefehlen in der folgenden Art auftreten:

```
I don't know how  to while
I don't know how  to foreach
...
```

3.3 Verwendung

Wenn eine Distributionseigene Variante von UCBLogo am System installiert ist, kann, wie im vorhergehenden Kapitel beschrieben, die neugebaute Version sofort getestet werden, ohne erst installiert werden zu müssen, wenn der `LIBLOC`-Pfad korrekt gesetzt ist.

Da UCBLogo von sich aus keine `readline`-Unterstützung zur komfortableren Bedienung der zeilenorientierten Schnittstelle bietet, empfiehlt die offizielle Dokumentation (`http://www.cs.berkeley.edu/~bh/announce`) `rlwrap` als Ersatz; dieser Empfehlung kann ich mich nur anschliessen, da es das Arbeiten mit dem Interpreter sehr viel flüssiger und einfacher gestaltet.

Das modifizierte Programm bietet drei neue Befehle: `midiopen`, `midimessage` und `midiclose`.

`midiopen` nutzt 0 oder 2 Eingabeparameter. Als Voreinstellung wird `client:port` als `128:0` angenommen, was dem ersten Client-Sequencer beim ALSA-System entspricht, auf den normalerweise ein Software-Midisynthesizer wie TiMidiTy+ gebunden wird. wenn `(midiopen :client :port)` aufgerufen wird, also etwa `(midiopen 64 0)`, dann wird dieser Synthesizer-Client anstatt dessen verwendet. Das wäre die korrekte Anwendung, wenn etwa ein Hardware-Midi-Synthesizer an einen MPU401 Adapter angeschlossen wäre oder ein „Wavetable"-Gerät verwendet werden soll, wie es bei manchen Audiokarten eingerichtet ist. Eine Liste der gültigen `client:port` Konfigurationen kann mittels des Befehls `pmidi -l` angezeigt werden.

Der empfohlene Weg, TiMidiTy+ als ALSA Sequencer Client aufzurufen, ist folgender:

```
> timidity -iAD -Os -q0/0 -k0 --reverb=d --chorus=d
```

oder

```
> timidity -iA -Ol -Os -q0/0 -k0 --reverb=d --chorus=d --verbose=3 | \
  grep -ivE \lq ME_NONE|output data .2048\rq.
```

Die erste Version verschwindet nach dem Start als Daemon im Hintergrund, die letztere Version zeigt als Fehlersuch-Ausgabe die Midi-Nachrichten an, die der Sequencer empfängt und bleibt im Vordergrund des aktiven Terminals.

Da alle beschriebenen Vorkehrungen unternommen wurden, um mit Prof. Neuwirths `mididefs.lgo` verwendet zu werden, ist hier ein Warnhinweis angebracht: da nur die Midi-Befehle in UCBLogo eingebaut wurden, funktioniert der Rest *nicht*, im speziellen schlagen `dllopen/dllcall` natürlich fehl. Es scheint auch, als ob diese Implementation eine leichte Diskrepanz in der Aufrufsemantik zu jener von MSWLogo aufweist, sodass einige kleine Änderungen an `mididefs.lgo` notwendig sind, bevor es mit der modfizierten Version von UCBLogo korrekt funkioniert:

1. Die Datei muss von DOS- auf UNIX-Zeilenumbruchkonvention umgestellt werden (cr/lf auf nur lf), andernfalls bricht UCBLogo mit folgender Fehlermeldung ab:

   ```
   too many inputs to midi.rescale  in noteon
   ]midimessage (list 208 + :chan - 1 (midi.rescale :press 0 1) 0)
   ```

2. Alle `midimessage`-Befehle müssen ein vorangestelltes `run` oder `ignore` erhalten, sonst erwartet UCBLogo einen Rückgabewert ungleich []:

   ```
   ? noteon 1 60 1
   You don't say what to do with []  in noteon
   ```

3. die `dllopen/dllcall`-Befehle müssen durch funktional äquivalente Konstrukte ersetzt werden. Da nur `waitmilli` diese verwendet, reicht es hier behelfsmässig, anstatt der `dllcall.* wait int :millisec / (1000/60)` einzusetzen. (Natürlich wäre es auch möglich, `dlopen/dlsym` als Midibefehle zur Verfügung zu stellen oder eine `waitmilli` Logofunktion zu implementieren; eine hässlichere (und durch den `fork/exec`-overhead wohl nicht sinnvolle) Möglichkeit wäre es auch, ein `(shell [usleep (:millisec * 1000)]` dafür zu verwenden.

Diese Kommandozeile wandelt die originale `mididefs.lgo` in eine UCBLogotaugliche Variante um:

```
> < mididefs.lgo.orig  sed -e 's/\r//' -e 's/midimessage/run &/g' \
  -e 's/dllload [^ ]*//' -e 's/.*dllcall.*/wait int :millisec \/ (1000\/60)/' \
  > mididefs.lgo
```

(Mensch beachte, dass das erste '>' den Shellprompt zymbolisieren soll, während die anderen grösser/kleiner-Zeichen unverändert zu kopieren sind)

Als weiteres *caveat* wäre hier anzuführen, dass TiMidiTy+ natürlich patches (Sample-Dateien, Audiofiles für die Instrumente) installiert haben muss, damit es auch tatsächlich Töne ausgeben kann. Wenn nur sporadische oder keine Töne ausgegeben werden, kann es am Fehlen oder an der Unvollständigkeit der Patchsets liegen; eine Distributionseigene Variante wird üblicherweise ein dazupassendes Paket mit Patch-Dateien aufweisen, ansonsten können Installationsanleitungen etwa auf der TiMidiTy+-Homapage gefunden werden (s. u.) oder auf jener der Software-Drummachine tk707, die ebenfalls gut mit einem korrekt eingerichteten Software-Synthesizer funktioniert.

Schlussendlich müssen die Programme `logo` und `timidity` tatsächlich in der Lage sein, auf die ALSA-Gerätedateien, im speziellen `/dev/snd/seq`, zuzugreifen. Wenn diese nicht existiert, was bei der kürzlichen Auflassung von devfs nicht so unwahrscheinlich erscheint, kann sie leicht mittels folgender Befehle neu eingerichtet werden:

```
# mkdir /dev/snd
# mknod /dev/snd/seq -m 666 c 116 1
```

Falls Lese- und Schreibrechte nicht bereits den (etwas riskanten) freizügigen Bedingungen der letzten Zeile entsprechen, können diese mit `chmod` gesetzt werden:

```
# chmod 666 /dev/snd/seq
```

Die etwas sicherere Variante ist die Verwendung einer eigenen `audio`-Gruppe, wie sie auf vielen Systemen bereits vorhanden sein wird:

```
# chmod 660 /dev/snd/seq
# chgrp audio /dev/snd/seq
```

```
# adduser <your_user> audio
```

(<**your_user**> ist hier durch den Namen des/der BenutzerIn zu erstetzen, welchEr `logo` bzw `timidity` ausführen können soll)

3.4 Implementation

Der Logo betreffende Teil war relativ leicht zu bewältigen: ausser den notwendigen Zeilen in `init.c` und globals.h, um die Schnittstelle zu definieren, waren noch Teile von `mmwind.c` aus dem MSWLogo-Quellcode zu portieren bzw. dür ALSA neu zu implementieren.

Hilfedateien für die neuen Befehle sind im entsprechenden Verzeichnis abgelegt.

Die Programmierung des ALSA-Sequencer-Teils erwies sich als etwas schwieriger: da ich keine Tutorials zur Erstellung von Sqeuencer Clients finden konnte, musste ich auf den Quellcode vorhandener Projekte zurückgreifen, nämlich `pmidi`, `tk707` sowie `ltrace`s der Aufruffolgen entsprechender Programme. `pmidi` scheint allerdings techisch gesehen ein ofline-Player zu sein: es ordnet den Ereignissen Zeitstempel zu, anstatt sie direkt abzuspielen, `tk707` ist dafür wieder sehr vielseitig und nicht allzu leicht zu analysieren.

Letzten Endes brachte ich den Code zum Laufen, nachdem ich das Versäumnis nachholte, die Queue zu starten, was ALSA davon angehalten hatte, die Nachrichten zu senden, bzw TiMidiTy+, sie zu empfange. Besonders erfreut war ich über die `snd_midi_event_encode`-Schnittstelle, welche eine Midi-Nachricht direkt in eine für die Sequencer-Sendefunktionen taugliche Struktur packte.

Die Reihenfolge der Midi-relevanten Funktionsaufrufe ist die folgende:

```
lmidiopen
 snd_seq_open
```

```
 snd_seq_alloc_queue
 snd_seq_client_id
 snd_seq_create_simple_port (wrapper for snd_seq_create_port)
 snd_seq_connect_to
 snd_seq_start_queue

midimessage
 snd_seq_ev_clear (macro)
 snd_midi_event_new (macro)
 snd_midi_event_init (macro)
 snd_midi_event_no_status
 snd_midi_event_reset_encode
 snd_seq_ev_set_dest
 snd_seq_ev_set_source
 snd_midi_event_encode
 snd_seq_ev_schedule_real
 snd_seq_event_output
 snd_seq_drain_output
```

„`long midi messages`“für `sysex` Ereignisse (0xf0, system exclusive, also für bestimmte Systeme spezifische Metadaten wie Sampledateien oder spezielle Kontrollnachrichten) wurden bisher nicht implementiert, da kein Bedarf bestand und ich keine Hardwaregeräte besitze, welche ungweöhnliche Ansteuerung benötigen könnten. Das wird sich wohl in nächster Zukunft nicht ändern, die Funktionalität sollte aber bei Bedarf einfach hinzuzufügen sein.

3.5 Bekannte Unbilden

- Der Rückgabewert von `midimessage` verwirrt ein nicht modifiziertes `mididefs.lgo`. Modifikationen siehe ‘usage’.
- `midiopen` gibt keine Beschwerde aus, wenn es zu einem nicht existenten oder empfangsbereiten Sequencer verbunden wird.

3.6 Noch zu erledigen

- `midiopen` gibt keine Beschwerde aus, wenn es zu einem nicht existenten oder empfangsbereiten Sequencer verbunden wird.
- `waitmilli` sollte in korrekter, eleganter Manier angelegt werden
- `autoconf`-Test auf das Vorhandensein von <sys/asoundlib.h> anstatt von <alsa/asoundlib.h>

- „`long midi messages`“für `sysex` Ereignisse (0xf0, system exclusive)

3.7 Verweise, Danksagungen usw.

- `http://homepage.univie.ac.at/erich.neuwirth/midilogo/midilogo.zip`

  Prof. Erich Neuwirths Vortrag beim Eurologo 2005-Kongress beschreibt erstmalig die `noteon/noteoff` Functionen als didaktisches Konstrukt, um die Informatik mit der Musikerziehung zu kombinieren und dabei spielerisch komplexe Datenstrukturen audzubauen und musikalische Grundbegriffe zu lernen und zu implementieren. Siehe auch sein PDF unter

  `http://eurologo2005.oeiizk.waw.pl/PDF/E2005Neuwirth.pdf`

- `http://www.alsa-project.org/`

  Das ALSA-Projekt (Advanced Linux Sound Architecture) stellt die Grundlage für alle modernen Midi-Applikationen unter GNU/Linux dar. Auf der Homepage findet sich umfangreiche Dokumentation zu den Schnittstellen.

  `http://www.alsa-project.org/alsa-doc/alsa-lib/`

- `http://www.cs.berkeley.edu/~bh/`

  UCB Logo kann direkt von Brian Harveys homepage bezogen werden, wo auch sein Werk ‘Computer Science Logo Style’ kostenlos in den Formaten PDF and HTML zur Verfügung steht.

- `http://www.softronix.com/`

  Die Firma, welche MSWLogo wartet und vertreibt, ein C++-Derivat von UCBLogo, hat einigen Aufwand betrieben, um alle zu dem Zeitpunkt attraktiv erscheinenden Multimedia-Erweiterungen auch für Logo zur Verfügung zu stellen. Da ihre Software unter der GPL Version 2 oder später steht, hatte ich natürlich auch keinerlei Bedenken, die Interfaces und Funktionsdefinitionen direkt zu kopieren. Danke dür diesen Beitrag an die Gemeinschaft!

- `http://www.timidity.jp/`

  TiMidity++ basiert auf TiMidity von Tuukka Toivonen. Es wird von Masanao Izumo <iz@onicos.co.jp> aktiv weiterentwickelt und umfasst heute neben der ALSA-Sequencer-Schnittstelle noch andere Möglichkeiten, es üeber das Netzwerk anzusprechen. Die remote-Methode erschien allerdings bei einem anderen Projekt etwas instabil bzw ungenau, ausserdem ist mir der Wartungsstatus dieser Schnittstelle nicht ganz klar.

Bezüglich der Konfiguration von TiMidiTy+, insbesondere das Setup der Patch-Dteien/Samples, verweise ich auf die Homepage bzw. die Dokumentation zur Software-Drum-Machine *tk707*.

- `http://sourceforge.net/projects/pmidi`

  Diese einfache Midi-Applikation wurde von Steve Ratcliffe <steve@parabola.demon.co.uk> geschrieben, um Midi-Dateien über dan ALSA-Sequencer abzuspielen. Einige der Codezeilen dieses Projekts (Sequencer-Device-Initialisierung, Schleife zur Ausgabe des Namens des Midi-Endpunkts) flossen ebenfalls in den Midi-Patch ein.

- `http://www-lmc.imag.fr/lmc-edp/Pierre.Saramito/tk707`

  TK707 ist eine Software-drum machine für das ALSA midi interface, teilweise trug es zu meinem Verständnis für die ALSA-Programmierung bei. Auf der Webseite ist auch Dokumentation über das Bauen und Einrichten des Software-Synthesizer TiMidiTy+ zu finden, die sehr hilfreich sein kann, wenn hierzu keine fertigen Pakete verwendet werden sollen.

  `http://www-lmc.imag.fr/lmc-edp/Pierre.Saramito/tk707/tk707.html#SEC8`

- `http://www.midiox.com/app.htm`

  `ftp://ftp.cs.ruu.nl/pub/MIDI/PROGRAMS/mf2tsrc.zip`

  Quellcode für t2mf und mf2t ist von Jamie O'Connell verfügbar. Die C-Syntax ist ein wenig altmodisch, kann jedoch mit GCC 3.x unter Verwendung folgender Konstrukte erfolgen:

```
> curl -O ftp://ftp.cs.ruu.nl/pub/MIDI/PROGRAMS/mf2tsrc.zip
> mkdir mf2tsrc
> cd mf2tsrc
> unzip -x ../mf2tsrc.zip
> sed -ie '/extern.*sys_errlist/d' mf2t.c t2mf.c
> make -f makefile.unx
> cp t2mf mf2t ~/bin
```

  Dies setzt voraus, dass ~/bin sich im $PATH befindet, ansonsten können die Programme in ein Verzeichnis kopiert werden, für welches diese Bedingung erfüllt ist.

```
# cp t2mf mf2t /usr/local/bin
```