5. Seeking

Seeking in Gstreamer is done with the seek() and seek_simple() methods. To be able to seek you will also need to tell Gstreamer what kind of seek it should do. In the following example we will use a gst.FORMAT_TIME format constant which will as you may guess do a time seek. :D We will also use the query_duration() and query_position() methods to get the file length and how long the file has currently played. Gstreamer uses nanoseconds by default so you have to adjust to that.

In this next example we take the Vorbis-Player from example 4.1 and update it with some more stuff so it's able to seek and show duration and position.

Example 5.1

```
#!/usr/bin/env python

import sys, os, thread, time
import pygtk, gtk, gobject
import pygst
pygst.require("0.10")
import gst

class GTK_Main:

        def __init__(self):
                window = gtk.Window(gtk.WINDOW_TOPLEVEL)
                window.set_title("Vorbis-Player")
                window.set_default_size(500, -1)
                window.connect("destroy", gtk.main_quit, "WM destroy")
                vbox = gtk.VBox()
                window.add(vbox)
                self.entry = gtk.Entry()
                vbox.pack_start(self.entry, False)
                hbox = gtk.HBox()
                vbox.add(hbox)
                buttonbox = gtk.HButtonBox()
                hbox.pack_start(buttonbox, False)
                rewind_button = gtk.Button("Rewind")
                rewind_button.connect("clicked", self.rewind_callback)
                buttonbox.add(rewind_button)
                self.button = gtk.Button("Start")
                self.button.connect("clicked", self.start_stop)
                buttonbox.add(self.button)
                forward_button = gtk.Button("Forward")
                forward_button.connect("clicked", self.forward_callback)
                buttonbox.add(forward_button)
                self.time_label = gtk.Label()
                self.time_label.set_text("00:00 / 00:00")
                hbox.add(self.time_label)
                window.show_all()

                self.player = gst.Pipeline("player")
                source = gst.element_factory_make("filesrc", "file-source")
                demuxer = gst.element_factory_make("oggdemux", "demuxer")
                demuxer.connect("pad-added", self.demuxer_callback)
                self.audio_decoder = gst.element_factory_make("vorbisdec", "vorbis-decoder")
                audioconv = gst.element_factory_make("audioconvert", "converter")
                audiosink = gst.element_factory_make("autoaudiosink", "audio-output")

                self.player.add(source, demuxer, self.audio_decoder, audioconv, audiosink)
                gst.element_link_many(source, demuxer)
                gst.element_link_many(self.audio_decoder, audioconv, audiosink)

                bus = self.player.get_bus()
                bus.add_signal_watch()
                bus.connect("message", self.on_message)

        def start_stop(self, w):
                if self.button.get_label() == "Start":
                        filepath = self.entry.get_text()
```

```
			if os.path.isfile(filepath):
				self.button.set_label("Stop")
				self.player.get_by_name("file-source").set_property("location", filepath)
				self.player.set_state(gst.STATE_PLAYING)
				self.play_thread_id = thread.start_new_thread(self.play_thread, ())
		else:
			self.play_thread_id = None
			self.player.set_state(gst.STATE_NULL)
			self.button.set_label("Start")
			self.time_label.set_text("00:00 / 00:00")

	def play_thread(self):
		play_thread_id = self.play_thread_id
		gtk.gdk.threads_enter()
		self.time_label.set_text("00:00 / 00:00")
		gtk.gdk.threads_leave()

		while play_thread_id == self.play_thread_id:
			try:
				time.sleep(0.2)
				dur_int = self.player.query_duration(gst.FORMAT_TIME, None)[0]
				if dur_int == -1:
					continue
				dur_str = self.convert_ns(dur_int)
				gtk.gdk.threads_enter()
				self.time_label.set_text("00:00 / " + dur_str)
				gtk.gdk.threads_leave()
				break
			except:
				pass

		time.sleep(0.2)
		while play_thread_id == self.play_thread_id:
			pos_int = self.player.query_position(gst.FORMAT_TIME, None)[0]
			pos_str = self.convert_ns(pos_int)
			if play_thread_id == self.play_thread_id:
				gtk.gdk.threads_enter()
				self.time_label.set_text(pos_str + " / " + dur_str)
				gtk.gdk.threads_leave()
			time.sleep(1)

	def on_message(self, bus, message):
		t = message.type
		if t == gst.MESSAGE_EOS:
			self.play_thread_id = None
			self.player.set_state(gst.STATE_NULL)
			self.button.set_label("Start")
			self.time_label.set_text("00:00 / 00:00")
		elif t == gst.MESSAGE_ERROR:
			err, debug = message.parse_error()
			print "Error: %s" % err, debug
			self.play_thread_id = None
			self.player.set_state(gst.STATE_NULL)
			self.button.set_label("Start")
			self.time_label.set_text("00:00 / 00:00")

	def demuxer_callback(self, demuxer, pad):
		adec_pad = self.audio_decoder.get_pad("sink")
		pad.link(adec_pad)

	def rewind_callback(self, w):
		pos_int = self.player.query_position(gst.FORMAT_TIME, None)[0]
		seek_ns = pos_int - (10 * 1000000000)
		self.player.seek_simple(gst.FORMAT_TIME, gst.SEEK_FLAG_FLUSH, seek_ns)

	def forward_callback(self, w):
		pos_int = self.player.query_position(gst.FORMAT_TIME, None)[0]
		seek_ns = pos_int + (10 * 1000000000)
		self.player.seek_simple(gst.FORMAT_TIME, gst.SEEK_FLAG_FLUSH, seek_ns)

	def convert_ns(self, t):
		# This method was submitted by Sam Mason.
		# It's much shorter than the original one.
		s,ns = divmod(t, 1000000000)
		m,s = divmod(s, 60)

		if m < 60:
			return "%02i:%02i" %(m,s)
		else:
			h,m = divmod(m, 60)
			return "%i:%02i:%02i" %(h,m,s)

GTK_Main()
gtk.gdk.threads_init()
gtk.main()
```

[Prev](#) [Next](#)
4. Src, sink, pad ... oh my! [Home](#) 6. Capabilities