

This is Google's cache of <http://pygstdocs.berlios.de/pygst-tutorial/seeking.html>. It is a snapshot of the page as it appeared on 15 May 2014 15:42:33 GMT. The [current page](#) could have changed in the meantime. [Learn more](#)
 Tip: To quickly find your search term on this page, press **Ctrl+F** or **⌘-F** (Mac) and use the find bar.

[Text-only version](#)

5. Seeking

[Prev](#)

[Next](#)

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



```

1 #!/usr/bin/env python
2
3 import sys, os, thread, time
4 import pygtk, gtk, gobject
5 import pygst
6 pygst.require("0.10")
7 import gst
8
9 class GTK_Main:
10
11     def __init__(self):
12         window = gtk.Window(gtk.WINDOW_TOPLEVEL)
13         window.set_title("Vorbis-Player")
14         window.set_default_size(500, -1)
15         window.connect("destroy", gtk.main_quit, "WM destroy")
16         vbox = gtk.VBox()
17         window.add(vbox)
18         self.entry = gtk.Entry()
19         vbox.pack_start(self.entry, False)
20         hbox = gtk.HBox()
21         vbox.add(hbox)
22         buttonbox = gtk.HButtonBox()
23         hbox.pack_start(buttonbox, False)
24         rewind_button = gtk.Button("Rewind")
25         rewind_button.connect("clicked", self.rewind_callback)
26         buttonbox.add(rewind_button)
27         self.button = gtk.Button("Start")
28         self.button.connect("clicked", self.start_stop)
29         buttonbox.add(self.button)
30         forward_button = gtk.Button("Forward")
31         forward_button.connect("clicked", self.forward_callback)
32         buttonbox.add(forward_button)
33         self.time_label = gtk.Label()
34         self.time_label.set_text("00:00 / 00:00")
35         hbox.add(self.time_label)
36         window.show_all()
37
38         self.player = gst.Pipeline("player")
39         source = gst.element_factory_make("filesrc", "file-source")
40         demuxer = gst.element_factory_make("oggdemux", "demuxer")
41         demuxer.connect("pad-added", self.demuxer_callback)
42         self.audio_decoder = gst.element_factory_make("vorbisdec", "vorbis-decoder")
43         audioconv = gst.element_factory_make("audioconvert", "converter")
44         audiosink = gst.element_factory_make("autoaudiosink", "audio-output")
45
46         self.player.add(source, demuxer, self.audio_decoder, audioconv, audiosink)
47         gst.element_link_many(source, demuxer)
48         gst.element_link_many(self.audio_decoder, audioconv, audiosink)
49
50         bus = self.player.get_bus()
51         bus.add_signal_watch()
52         bus.connect("message", self.on_message)
53
54     def start_stop(self, w):
55         if self.button.get_label() == "Start":
56             filepath = self.entry.get_text()

```

```

57         if os.path.isfile(filepath):
58             self.button.set_label("Stop")
59             self.player.get_by_name("file-source").set_property("location", filepath)
60             self.player.set_state(gst.STATE_PLAYING)
61             self.play_thread_id = thread.start_new_thread(self.play_thread, ())
62         else:
63             self.play_thread_id = None
64             self.player.set_state(gst.STATE_NULL)
65             self.button.set_label("Start")
66             self.time_label.set_text("00:00 / 00:00")
67
68     def play_thread(self):
69         play_thread_id = self.play_thread_id
70         gtk.gdk.threads_enter()
71         self.time_label.set_text("00:00 / 00:00")
72         gtk.gdk.threads_leave()
73
74         while play_thread_id == self.play_thread_id:
75             try:
76                 time.sleep(0.2)
77                 dur_int = self.player.query_duration(gst.FORMAT_TIME, None)[0]
78                 if dur_int == -1:
79                     continue
80                 dur_str = self.convert_ns(dur_int)
81                 gtk.gdk.threads_enter()
82                 self.time_label.set_text("00:00 / " + dur_str)
83                 gtk.gdk.threads_leave()
84                 break
85             except:
86                 pass
87
88             time.sleep(0.2)
89             while play_thread_id == self.play_thread_id:
90                 pos_int = self.player.query_position(gst.FORMAT_TIME, None)[0]
91                 pos_str = self.convert_ns(pos_int)
92                 if play_thread_id == self.play_thread_id:
93                     gtk.gdk.threads_enter()
94                     self.time_label.set_text(pos_str + " / " + dur_str)
95                     gtk.gdk.threads_leave()
96                 time.sleep(1)
97
98     def on_message(self, bus, message):
99         t = message.type
100         if t == gst.MESSAGE_EOS:
101             self.play_thread_id = None
102             self.player.set_state(gst.STATE_NULL)
103             self.button.set_label("Start")
104             self.time_label.set_text("00:00 / 00:00")
105         elif t == gst.MESSAGE_ERROR:
106             err, debug = message.parse_error()
107             print "Error: %s" % err, debug
108             self.play_thread_id = None
109             self.player.set_state(gst.STATE_NULL)
110             self.button.set_label("Start")
111             self.time_label.set_text("00:00 / 00:00")
112
113     def demuxer_callback(self, demuxer, pad):
114         adec_pad = self.audio_decoder.get_pad("sink")
115         pad.link(adec_pad)
116
117     def rewind_callback(self, w):
118         pos_int = self.player.query_position(gst.FORMAT_TIME, None)[0]
119         seek_ns = pos_int - (10 * 1000000000)
120         self.player.seek_simple(gst.FORMAT_TIME, gst.SEEK_FLAG_FLUSH, seek_ns)
121
122     def forward_callback(self, w):
123         pos_int = self.player.query_position(gst.FORMAT_TIME, None)[0]
124         seek_ns = pos_int + (10 * 1000000000)
125         self.player.seek_simple(gst.FORMAT_TIME, gst.SEEK_FLAG_FLUSH, seek_ns)
126
127     def convert_ns(self, t):
128         # This method was submitted by Sam Mason.
129         # It's much shorter than the original one.
130         s, ns = divmod(t, 1000000000)
131         m, s = divmod(s, 60)
132
133         if m < 60:
134             return "%02i:%02i" % (m, s)
135         else:
136             h, m = divmod(m, 60)
137             return "%i:%02i:%02i" % (h, m, s)
138
139     def __init__(self):
140         gtk.gtk.gtk_threads_init()
141         gtk.main()

```

[Prev](#)

4. Src, sink, pad ... oh my!

[Home](#)

[Next](#)

6. Capabilities