Jukebox Extension

```
#include "ruby.h"
#include "cdjukebox.h"
static VALUE cCDPlayer;

// Helper function to free a vendor CDJukebox

static void cd_free(void *p) {
free_jukebox(p);
}

// Allocate a new CDPlayer object, wrapping
// the vendor's CDJukebox structure
static VALUE cd_alloc(VALUE klass) {
CDJukebox *jukebox;
VALUE obj;
// Vendor library creates the Jukebox
jukebox = new_jukebox();
// then we wrap it inside a Ruby CDPlayer object
obj = Data_Wrap_Struct(klass, 0, cd_free, jukebox);
return obj;
}

// Assign the newly created CDPLayer to a particular unit
static VALUE cd_initialize(VALUE self, VALUE unit) {
int unit_id;
CDJukebox *jb;
Data_Get_Struct(self, CDJukebox, jb);
unit_id = NUM2INT(unit);
assign_jukebox(jb, unit_id);
return self;
}

// Copy across state (used by clone and dup). For jukeboxes, we
// actually create a new vendor object and set its unit number from the old
static VALUE cd_init_copy(VALUE copy, VALUE orig) {
CDJukebox *orig_jb;
CDJukebox *copy_jb;
if (copy == orig)
return copy;
// we can initialize the copy from other CDPlayers or their
// subclasses only
if (TYPE(orig) != T_DATA ||
RDATA(orig)>
dfree != (RUBY_DATA_FUNC)cd_free) {
rb_raise(rb_eTypeError, "wrong argument type");
}
```

```
// copy all the fields from the original object's CDJukebox structure to the new object
Data_Get_Struct(orig, CDJukebox, orig_jb);
Data_Get_Struct(copy, CDJukebox, copy_jb);
MEMCPY(copy_jb, orig_jb, CDJukebox, 1);
return copy;
}

// The progress callback yields to the caller the percent complete
static void progress(CDJukebox *rec, int percent) {
if (rb_block_given_p()) {
if (percent > 100) percent = 100;
if (percent < 0) percent = 0;
rb_yield(INT2FIX(percent));
}
}

// Seek to a given part of the track, invoking the progress callback as we go
static VALUE cd_seek(VALUE self, VALUE disc, VALUE track) {
CDJukebox *jb;
Data_Get_Struct(self, CDJukebox, jb);
jukebox_seek(jb,
NUM2INT(disc),
NUM2INT(track),
progress);
return Qnil;
}

// Return the average seek time for this unit
static VALUE
cd_seek_time(VALUE self)
{
double tm;
CDJukebox *jb;
Data_Get_Struct(self, CDJukebox, jb);
tm = get_avg_seek_time(jb);
return rb_float_new(tm);
}

// Return this player's unit number
static VALUE cd_unit(VALUE self) {
CDJukebox *jb;
Data_Get_Struct(self, CDJukebox, jb);
return INT2NUM(jb>
unit_id);
}
```

```
void Init_CDPlayer() {
cCDPlayer = rb_define_class("CDPlayer", rb_cObject);
rb_define_alloc_func(cCDPlayer, cd_alloc);
rb_define_method(cCDPlayer, "initialize", cd_initialize, 1);
rb_define_method(cCDPlayer, "initialize_copy", cd_init_copy, 1);
rb_define_method(cCDPlayer, "seek", cd_seek, 2);
rb_define_method(cCDPlayer, "seek_time", cd_seek_time, 0);
rb_define_method(cCDPlayer, "unit", cd_unit, 0);
}
```

Now we can control our jukebox from Ruby in a nice, object-oriented way.

```
require 'CDPlayer'
p = CDPlayer.new(13)
puts "Unit is #{p.unit}"
p.seek(3, 16) {|x| puts "#{x}% done" }
puts "Avg. time was #{p.seek_time} seconds"
p1 = p.dup
puts "Cloned unit = #{p1.unit}"
```

produces:

```
Unit is 13
26% done
79% done
100% done
Avg. time was 1.2 seconds
Cloned unit = 13
```