一、编写cn.com.itep.powermanager.xml文件

```
<node name="/cn/com/itep/PowerManager">
    <interface name="cn.com.itep.PowerManager.Display">
        <method name="get_timeout">
            <arg direction="out" type="i" name="timeout"/>
        </method>
        <method name="set_timeout">
            <arg direction="in" type="i" name="timeout"/>
        </method>
    </interface>
</node>
```

二、在Makefile.am文件中加入生成generated.c generated.h的语句, 并添加到编译模块中。

```
AM_CPPFLAGS = -DLOCALEDIR=\"$(localedir)\"
autostartdir = $(sysconfdir)/xdg/autostart
gladedir = $(pkgdatadir)
confdir = $(pkgdatadir)

introspection = cn.com.itep.powermanager.xml

bin_PROGRAMS = itep-power-manager
glade_DATA = itep-power-manager.glade
conf_DATA = itep-power-manager.conf
autostart_DATA = itep-power-manager.desktop
itep_power_manager_SOURCES = generated.c generated.h main.c \
    itep-power-manager.c \
    itep-power-manager.h \
    ipm-display.c \
    ipm-dispaly.h \
    egg-idletime.c \
    egg-idletime.h \
    ipm-idle-shutdown.c \
    ipm-idle-shutdown.h \
    ipm-timed-shutdown.c \
    ipm-timed-shutdown.h \
    misc.c \
    misc.h \
    ipm-systemd.c \
    ipm-systemd.h
itep_power_manager_CFLAGS = $(GTK_CFLAGS)
itep_power_manager_CFLAGS += -DGLADE_FILE=\"$(gladedir)\/itep-power-manager.glade\"
itep_power_manager_CFLAGS += -DCONF_FILE=\"$(confdir)\/itep-power-manager.conf\"
itep_power_manager_LDADD = $(GTK_LIBS) -lXext -lX11

LDADD = $(LIBINTL)
```

```
generated.c generated.h: $(introspection)
 gdbus-codegen --interface-prefix cn.com.itep. --generate-c-code generated --c-namespace Itep $^

EXTRA_DIST = $(introspection) $(glade_DATA) $(autostart_DATA) $(conf_DATA)

clean-local:
 -rm -rf generated.c generated.h
```

三、编写服务器端程序

```
#include "generated.h"

static gboolean display_get_timeout(ItepPowerManagerDisplay *display,
                                    GDBusMethodInvocation *invocation,
                                    ItepPowerManager *ipm) //接收传人参数
{
        itep_power_manager_display_complete_get_timeout(display,
                                    invocation,
                                    itep_power_manager_display_get_timeout(ipm));
        return TRUE;
}

static gboolean display_set_timeout(ItepPowerManagerDisplay *display,
                                    GDBusMethodInvocation *invocation,
                                    gint timeout,
                                    ItepPowerManager *ipm) //接收传人参数

{
        itep_power_manager_display_set_timeout(ipm, timeout);
        itep_power_manager_display_complete_set_timeout(display, invocation);
        return TRUE;
}

static void on_bus_acquired(GDBusConnection *connection, const gchar *name, ItepPowerManager *ipm) //接收传人参数
{
    ItepPowerManagerDisplay *display = NULL;
    display = itep_power_manager_display_skeleton_new();

    g_signal_connect(display, "handle-get-timeout",
                                G_CALLBACK(display_get_timeout), ipm);//将ipm传给回调函数
    g_signal_connect(display, "handle-set-timeout",
                                G_CALLBACK(display_set_timeout), ipm);//将ipm传给回调函数
    g_dbus_interface_skeleton_export(G_DBUS_INTERFACE_SKELETON(display),
                                connection,
                                "/cn/com/itep/powermanager/display",
                                NULL);
}

static void on_name_acquired(GDBusConnection *connection, const gchar *name, ItepPowerManager *ipm) //接收传人参
数
{
```

```
    ;
}

static void on_name_lost(GDBusConnection  *connection, const gchar *name, ItepPowerManager *ipm) //接收传人参数
{
    ;
}


int main(int argc, char *argv[])
{
    ItepPowerManager *ipm; //ItepPowerManager为自定义类型。
    ipm = itep_power_manager_new();  //创建ipm用来传递参数，参数可以是任意类型

    gtk_init(&argc, &argv);
    g_bus_own_name(G_BUS_TYPE_SESSION,
                   "cn.com.itep.powermanager",
                   G_BUS_NAME_OWNER_FLAGS_NONE,
                   on_bus_acquired,
                   on_name_acquired,
                   on_name_lost,
                   ipm, //传递参数给on_bus_acquired, on_name_acquired, on_name_lost;
                   NULL);
    gtk_main();
}
```

四、编写客户端程序

复制xml文件，添加生成generated.c 与 generated.h文件的代码到 客户端的Makefile.am中，与服务器端添加规则一样。

```
#include "generated.h"

int main(int argc, char *argv[])
{
    ItepPowerManagerDisplay *display;
    GError *error = NULL;
    int get_display_time = 0;
    int set_display_time = 0;

    display = itep_power_manager_display_proxy_new_for_bus_sync(
                                    G_BUS_TYPE_SESSION,
                                    G_DBUS_PROXY_FLAGS_NONE,
                                    "cn.com.itep.powermanager",
                                    "/cn/com/itep/powermanager/display",
                                    NULL
                                    &error);
    if (display == NULL)
    {
        g_printf("Error getting proxy:%s\n", error->message);
        g_error_free(error);
        return -1;
    }
```

```
    gtk_init(&argc, &argv);

    itep_power_manager_display_call_get_timeout_sync( //调用服务器端get函数，获取返回值，保存到get_display_time中
                                    display,
                                    &get_display_time,
                                    NULL,
                                    NULL);

    itep_power_manager_display_call_set_timeout_sync( //调用服务器端set函数， 将set_display_time, 传人服务器set函数。
                                    display,
                                    set_display_time,
                                    NULL,
                                    NULL);
    gtk_main();
}
```