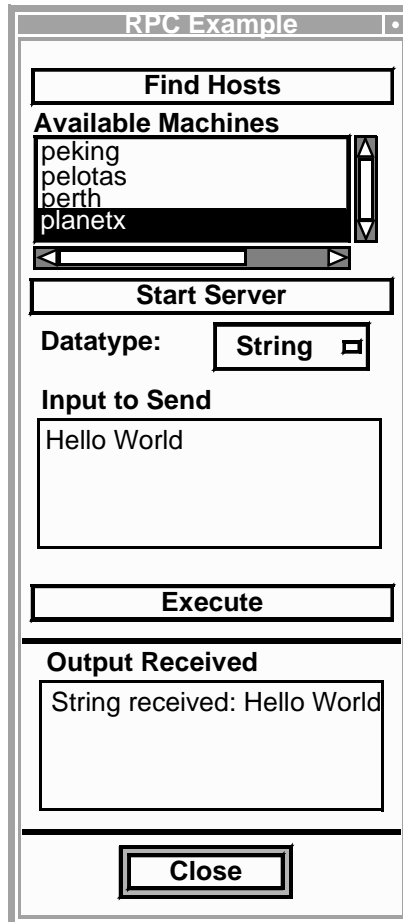

EXERCISE 12

Remote Procedural Calls



Objectives:

- Understand general Remote Procedure Calls.
- Become familiar with Patran's interface of the Remote Procedure Calls.
- Write a simple function to receive a Remote Procedure Call and act on a server machine.
- Write a simple PCL interface for the Remote Procedure Call to act as a Client machine.



Exercise Description:

Write a program that will act as a server. Compile this program and place the executable in your path. Write a PCL function that will interface with that function. Start Patran and spawn this function on a remote machine. Communicate with that machine, via the Remote Procedure Calls.

Exercise Procedure:

1. Change your directory to `ex12`. Edit the C function in the file `exercise_12a.template`. Replace the blanks with the appropriate C expressions. Rename the file to `rpc_process.c` when you are done.
2. Compile the function, linking it with the necessary library `rpc.a`.

SGI:

```
cc rpc_process.c <install directory>/rpc.a -lsun -o rpc_process
```

Sun Solaris:

```
cc rpc_process.c <install directory>/rpc.a -lnsl -lsocket -o rpc_process
```

All Others:

```
cc rpc_process.c <install directory>/rpc.a -o rpc_process
```

3. Make a `bin` directory in your home directory. Copy the `rpc_process` executable to that directory, and add that directory to your path.

```
% cd
```

```
% mkdir bin
```

```
% cp <work_dir> /rpc_process bin
```

```
% vi .cshrc
```

```
set path = ($path ~/bin )
```

4. Edit the PCL function in file `exercise_12b.template`. Replace the blanks with the appropriate PCL expressions. Rename the file to `rpc_example.pcl` when you are done.
5. Test your work by running Patran and playing the session file `test12.ses`.

Exercise Template:

RPC_PROCESS.C

```
#define _PI_ 3.14159
static char rcsid[]=
"$Header: /kola/users/pflib/testfiles/rpc/RCS/test1.c,v 1.2 93/10/04 16:20:02
    malamut Exp $";
RpcHandler(handle)
int handle;
{
int type, count, rand_int;
float rand_flt;
char string[256];
char buf[256];
    *****1***** (handle, &type, &count);
    printf("type = %d, count = %d\n", type, count);
/* added lines begin here */
    switch ( type ) {
        case 3:
/* integer */
            *****2*****;
            strcpy(buf, "Integer recieved : ");
            RpcPutStatus(handle, 13);
            RpcPutString(handle, buf);
            RpcPutInteger(handle, rand_int);
            break;
        case 4:
/* boolean */
            RpcGetBoolean(handle, &rand_int);
            if (rand_int) { strcpy(buf, "Logical recieved : True"); }
            else { strcpy(buf, "Logical recieved : False"); };
            *****3***** (handle, 14);
            RpcPutString(handle, buf);
            break;
        case 5:
/* real */
            RpcGetReal(handle, &rand_flt);
            strcpy(buf, "Diameter recieved, Circumference is : ");
            RpcPutStatus(handle, 15);
            RpcPutString(handle, buf);
            *****4***** (handle, rand_flt * _PI_);
            break;
        case 6:
/* string */
            RpcGetString(handle, &string[0], sizeof(string));
            strcpy(buf, "String received : ");
            strcat(buf, string);
            RpcPutStatus(handle, 16);
            *****5*****;
            break;
        default:
            printf("I am confused by your strange and wonderous ways. (%d)", type);
    }
/* RpcServerEnd(); */ /* Server will die after current Rpc call completes */
}

/*****
 *
 * The all PDA RPC servers must make a call to RpcInitServer. This establishes
 * the server's run-time environment as follows. All clients must make a call
 * to RpcAddServer and RpcInitClient.
 *****/
```

```
*
*  RpcInitServer(  int prognum          - program number for client/server
*                  same value as in RpcInitClient
*                  int progver         - program version
*                  void (*handler)(int) - RPC service handler
*                  logical inetd       - server startup via inetd.
*                  int server_timeout  - # of seconds of inactivity
*                                      before server dies
*                  logical rpc_debug   - display rpc data on stdout
*
*****/
main(argc, argv)
int argc;
char **argv;
{
int prognum;
  if((argc != 2) || (sscanf(argv[1], "%d", &prognum) != 1)){
    printf("usage : %s <prognum>\n", argv[0]);
    exit(2);
  }
  RpcInitServer(*****6*****);
}
#include "appforms.p"
#define _PROG_NAME "rpc_exec"
```

RPC_EXAMPLE.PCL

```
CLASS rpc_example

CLASSWISE WIDGET start_button,send_button,find_button
CLASSWISE WIDGET send_text_id, receive_text_id
CLASSWISE WIDGET host_lb, type_om
CLASSWISE STRING hostname[256], send_type[4]
CLASSWISE INTEGER handle, status, prog_num

FUNCTION init()
  REAL y_loc

  WIDGET form_id

  form_id = ui_form_create( @
    /* callback (always blank for forms) */ "" , @
    /* form x loc (standard) */ FORM_X_LOC_SML , @
    /* form y loc (standard) */ FORM_Y_LOC , @
    /* xy loc is Upper Left (standand) */ "UL" , @
    /* form width (standard) */ FORM_WID_SML , @
    /* form height (standard max) */ FORM_HGT_TALL , @
    /* Title in Banner */ "RPC Example" , @
    /* unused argument */ "" )

  y_loc = FORM_T_MARGIN + INTER_WIDGET_SPACE

  /*
   * Create the "start server" button
   */

  find_button = ui_button_create( @
    /* parent form */ form_id , @
    /* callback */ "find_hosts" , @
    /* x loc */ BUTTON_HALF_X_LOC1 , @
    /* y loc */ y_loc , @
    /* button width (half size) */ BUTTON_WID_FULL , @
    /* button height (default size ) */ BUTTON_HGT , @
    /* button label */ "Find Hosts" , @
    /* unused argument */ TRUE , @
    /* default button */ TRUE )

  y_loc += BUTTON_HGT + INTER_WIDGET_SPACE

  host_lb = ui_listbox_create( @
    /* parent */ form_id , @
    /* callback */ "" , @
    /* x pos */ FORM_L_MARGIN , @
    /* y pos */ y_loc , @
    /* width */ LBOX_WID_SINGLE , @
    /* num_rows */ 4 , @
    /* label */ "Available Machines" , @
    /* selection */ "SINGLE" , @
    /* sort */ TRUE )

  y_loc += LBOX_4L_HGT_LABOVE + INTER_WIDGET_SPACE

  start_button = ui_button_create( @
    /* parent form */ form_id , @
    /* callback */ "start_cb" , @
    /* x loc */ BUTTON_HALF_X_LOC1 , @
```

```

        /* y_loc                                */ y_loc                                , @
        /* button width (half size)            */ BUTTON_WID_FULL                          , @
        /* button height (default size )      */ BUTTON_HGT                               , @
        /* button label                        */ "Start Server"                           , @
        /* unused argument                    */ TRUE                                       , @
        /* default button                     */ FALSE                                      )

y_loc += BUTTON_HGT + INTER_WIDGET_SPACE

type_om = ui_optionmenu_create(                                     @
        /* parent                                */ form_id                            , @
        /* callback                              */ "om_cb"                          , @
        /* x pos                                 */ FORM_L_MARGIN                       , @
        /* y pos                                 */ y_loc                                , @
        /* lab_len                              */ 1.0                                , @
        /* label                                */ "Datatype:"                          , @
        /* lab_above                            */ FALSE )

y_loc += OPT_MENU_HGT_NO_LABOVE + INTER_WIDGET_SPACE

ui_item_create( type_om, "STR", "String", FALSE)
ui_item_create( type_om, "INT", "Integer", FALSE)
ui_item_create( type_om, "REA", "Real", FALSE)
ui_item_create( type_om, "LOG", "Logical", FALSE)

send_text_id = ui_text_create( @
        /* parent form                          */ form_id                            , @
        /* callback                              */ ""                                , @
        /* x loc                                 */ UNFRAMED_L_MARGIN                   , @
        /* y loc                                 */ y_loc                                , @
        /* width                                 */ TBOX_WID_SINGLE                       , @
        /* number of rows                       */ 5                                    , @
        /* label                                */ "Input to Send"                       , @
        /* default value                        */ "Hello World"                       , @
        /* editable?                            */ TRUE                                       )

y_loc += TBOX_5L_HGT_LABOVE + INTER_WIDGET_SPACE

send_button = ui_button_create( @
        /* parent form                          */ form_id                            , @
        /* callback                              */ "send_cb"                          , @
        /* x loc                                 */ BUTTON_HALF_X_LOC1                       , @
        /* y loc                                 */ y_loc                                , @
        /* button width (half size)            */ BUTTON_WID_FULL                          , @
        /* button height (default size )      */ BUTTON_HGT                               , @
        /* button label                        */ "Execute"                               , @
        /* unused argument                    */ TRUE                                       , @
        /* default button                     */ FALSE                                      )

y_loc += BUTTON_HGT + INTER_WIDGET_SPACE

ui_separator_create ( @
        /* parent form                          */ form_id                            , @
        /* callback                              */ ""                                , @
        /* x loc                                 */ 0.0                                , @
        /* y loc                                 */ y_loc                                , @
        /* button width (half size)            */ FORM_WID_SML                            , @
        /* default setting                     */ TRUE                                       )

y_loc += LINE_THICKNESS + INTER_WIDGET_SPACE

```

```

receive_text_id = ui_text_create( @
    /* parent form                */ form_id          , @
    /* callback                    */ ""              , @
    /* x loc                       */ UNFRAMED_L_MARGIN , @
    /* y loc                       */ y_loc           , @
    /* width                       */ TBOX_WID_SINGLE  , @
    /* number of rows              */ 5                , @
    /* label                       */ "Output Received" , @
    /* default value               */ ""              , @
    /* editable?                  */ TRUE             )

y_loc += TBOX_5L_HGT_LABOVE + INTER_WIDGET_SPACE

ui_separator_create ( @
    /* parent form                */ form_id          , @
    /* callback                    */ ""              , @
    /* x loc                       */ 0.0             , @
    /* y loc                       */ y_loc           , @
    /* button width (half size)    */ FORM_WID_SML    , @
    /* default setting            */ TRUE             )

y_loc += LINE_THICKNESS + 2.0 * INTER_WIDGET_SPACE

ui_button_create( @
    /* parent form                */ form_id          , @
    /* callback to close form     */ "close_cb"       , @
    /* x loc for second of two buttons */ BUTTON_THIRD_X_LOC2 , @
    /* y loc                       */ y_loc           , @
    /* button width (half size)    */ BUTTON_WID_THIRD , @
    /* button height (default size ) */ ZERO         , @
    /* button label                */ "Close"         , @
    /* unused argument            */ TRUE            , @
    /* not default button         */ FALSE           )

y_loc += BUTTON_HGT * .6 + 2.0 * FORM_B_MARGIN

ui_wid_set( form_id , "HEIGHT" , y_loc )

    ui_wid_set(send_button,"ENABLE", false )
$    ui_wid_set(host_lb,"ENABLE", false )
    ui_wid_set(start_button,"ENABLE", false )

END FUNCTION /* init */

FUNCTION set_unique_num()

    REAL dummy

    dummy = sys_clock()

    prog_num = mth_nint(dummy / 10)

END FUNCTION /* set_unique_num */

FUNCTION start_cb()

    LOGICAL wonder
    static INTEGER version = 0
    STRING mach_type[10], rem_shell_com[10]

    version += 1
    ui_wid_get(host_lb, "VALUE", hostname)

```

```

IF ( hostname == "" ) THEN
  user_message("ACK", 6735331, "RPC", "I need a host!\n Aborting!")
RETURN 1
END IF

rem_shell_com = "rsh "
sys_get_info ( 1, mach_type)
SWITCH ( mach_type )
CASE("SUN4")
  status = utl_process_spawn("/usr/etc/ping " // hostname, TRUE)
  wonder = utl_process_error(status)
  IF ( wonder ) THEN
    user_message("ACK",6735331,"RPC","Host " // hostname // @
      " not responding")

    RETURN 1
  END IF
CASE("SGI4D")
  status = utl_process_spawn("/usr/etc/ping -c 1 " // hostname , TRUE)
  wonder = utl_process_error(status)
  IF ( wonder ) THEN
    user_message("ACK",6735331,"RPC","Host " // hostname // @
      " not responding")

    RETURN 1
  END IF
CASE("DECS")
CASE("RS6K")
  status = utl_process_spawn("/etc/ping " // hostname // " -c 1", TRUE)
  wonder = utl_process_error(status)
  IF ( wonder ) THEN
    user_message("ACK",6735331,"RPC","Host " // hostname // @
      " not responding")

    RETURN 1
  END IF
CASE("HP700")
/* HP rsh command is remsh */
  rem_shell_com = "remsh "
  status = utl_process_spawn("/etc/ping " // hostname // " 8 1", TRUE)
  wonder = utl_process_error(status)
  IF ( wonder ) THEN
    user_message("ACK",6735331,"RPC","Host " // hostname // @
      " not responding")

    RETURN 1
  END IF
END SWITCH

rpc_example.set_unique_num()

status = utl_process_spawn(rem_shell_com // hostname // " " // @
  _PROG_NAME // " " // str_from_integer(prog_num), FALSE )
IF( utl_process_error( status ) ) THEN
  utl_display_process_error( status, 4 )
  RETURN status
END IF
utl_process_spawn("sleep 5", TRUE)

*****1*****
*****2*****
ui_wid_set(start_button,"ENABLE", false )
ui_wid_set(send_button,"ENABLE", true )
ui_wid_set(type_om, "ENABLE", TRUE)

```

```

END FUNCTION /* start_cb */

FUNCTION send_cb()

STRING  instring[256]
INTEGER temp
STRING  outstring[256]
INTEGER out_int
REAL    out_real

    ui_wid_get(type_om, "VALUE", send_type)
    ui_wid_set(type_om, "ENABLE", FALSE)
    ui_wid_get(send_text_id, "VALUE", instring)

    SWITCH (send_type)
        CASE("STR")
            *****3*****
        CASE("INT")
            temp = str_to_integer(instring)
            rpc_put_integer(handle, temp)
        CASE("REA")
            rpc_put_real(handle, str_to_real(instring))
        CASE("LOG")
            rpc_put_boolean(handle, str_to_logical(instring))
    END SWITCH
    *****4*****

    rpc_get_status(handle, status)
    SWITCH (send_type)
        CASE("STR")
            rpc_get_string(handle, outstring)
        CASE("INT")
            rpc_get_string(handle, outstring)
            rpc_get_integer(handle, out_int)
            outstring = outstring // str_from_integer(out_int)
        CASE("REA")
            rpc_get_string(handle, outstring)
            *****5*****
            outstring = outstring // str_from_real(out_real)
        CASE("LOG")
            rpc_get_string(handle, outstring)
    END SWITCH
    ui_wid_set(receive_text_id, "VALUE", outstring)
    ui_wid_set(start_button,"ENABLE", TRUE )
    ui_wid_set(type_om, "ENABLE", TRUE)

END FUNCTION /* send_cb */

FUNCTION display

    ui_form_display( "rpc_example" )

END FUNCTION /* display */

FUNCTION close_cb

    ui_form_hide( "rpc_example" )

END FUNCTION /* close_cb */

```

```

FUNCTION find_hosts()

    STRING filespec[64], temp_host[128], output[64]
    STRING host_array[32](VIRTUAL)
    INTEGER chan, answer, linelen
    INTEGER hosts = 30, lines = 0

    file_build_fname("/etc", "hosts", "equiv", "O", filespec)

    IF ( ! ( file_exists(filespec, "") ) ) THEN
        user_message("Ack", 8569665, "RPC", "Unable to find " // filespec )
        RETURN 1
    END IF

    status = text_open( filespec, "OR", 0, 0, chan)
    IF ( status != 0 ) THEN
        user_message("Ack", 8569665, "RPC", "Unable to open " // filespec )
        RETURN 1
    END IF

    status = sys_allocate_array(host_array, 1, hosts)
    IF ( status != 0 ) THEN
        user_message("Ack", 8569665, "RPC", "Unable to allocate array")
        RETURN 1
    END IF

    WHILE (text_read_string(chan,temp_host,linelen) == 0 )
        lines += 1
        IF ( lines > hosts ) THEN
            hosts += 10
            status = sys_reallocate_array(host_array, 1, hosts)
            IF ( status != 0 ) THEN
                user_message("Ack", 8569665, "RPC", "Unable to reallocate array")
                RETURN 1
            END IF
        END IF
        host_array(lines) = temp_host
    END WHILE

    status = text_close(chan, "")
    IF ( status != 0 ) THEN
        msg_to_form(status,2,0,0,0.0,"")
        return status
    ENDIF

    status = ui_listbox_items_create(host_lb, host_array, host_array, lines, @
        WIDGET_NULL)
    IF ( status != 0 ) THEN
        user_message("Ack", 8569665, "RPC", "Unable to allocate array")
        RETURN 1
    END IF

    ui_wid_set(find_button, "ENABLE", FALSE)
    ui_wid_set(start_button, "ENABLE", TRUE)

END FUNCTION /* find_hosts */

FUNCTION om_cb(name)

```

```

STRING NAME[]

SWITCH (name)
  CASE("INT")
    ui_wid_set(send_text_id, "VALUE", "123")
  CASE("REA")
    ui_wid_set(send_text_id, "VALUE", "123.4")
  CASE("STR")
    ui_wid_set(send_text_id, "VALUE", "Hello Dolly")
  CASE("LOG")
    ui_wid_set(send_text_id, "VALUE", "TRUE")
END SWITCH

END FUNCTION /* om_cb */

END CLASS /* rpc_example */

```

```

*1* RpcInquireItem
*2* RpcGetInteger(handle, &rand_int)
*3* RpcPutStatus
*4* RpcPutReal
*5* RpcPutString(handle, buf)
*6* program, 1, RpcHandler, 0, 90, 0

RPC_EXAMPLE.PCL

*1* rpc_add_server(_PROG_NAME, prog_num, version, hostname)
*2* rpc_init_client(_PROG_NAME, handle)
*3* rpc_put_string(handle, instrng)
*4* rpc_call(handle)
*5* rpc_get_real(handle, out_real)

```