

INVOKING EXTERNAL PROCEDURES

Elizabeth Boss, Boss Consulting Services, Inc.

Abstract

This presentation will discuss the steps necessary to create an external procedure (using C) that is called from a PL/SQL program unit. The attendee will see how Oracle8 allows the developer to create a program in C and call it from a PL/SQL program unit. The presentation will discuss the creation of the PL/SQL wrapper procedure that will be used to map PL/SQL variables to C variables. The database will be queried, and data will be passed from the database to a report created through the 3GL program. The presentation will cover all steps necessary to invoke the 3GL procedure/function.

External Procedures

Oracle8 provides a method for calling a 3GL program from within a PL/SQL program unit. Currently, only C programs can be called from a PL/SQL program unit. Future releases of Oracle will support Java and C++ external programs. An external procedure is simply a program unit that is not written in Oracle's SQL or PL/SQL and resides outside of the database. To execute external procedures, a process called *extproc* is spawned whenever a call to an external procedure is issued. The *extproc* process loads the shared library (or libraries) that contain the C program and execute it. The results are sent back to the shadow process, which in turn sends the results back to the client.

Calling External Procedures

External procedures must be stored as shared libraries at the operating system level. In a Windows NT environment, the procedure would be compiled into a DLL. In a Unix environment, the procedure might be compiled into a shared object (.so) depending on the flavor of Unix.

An extproc process will exist for each session that calls an external procedure. The extproc process is spawned by the SQL*NET or NET8 listener. The extproc process and listener process must reside on the same server as the database.

Creating the External Procedure

The following steps are required when creating an external procedure:

1. Create and compile the C program into a shared library.
2. Ensure that the SQL*Net listener process is configured and running on the database server machine.
3. Create a Library object in the Oracle database to represent the operating system library.
4. Create a PL/SQL procedure to map PL/SQL parameters to C parameters.

Creating the C Procedure

The first step in creating an external procedure is to write and compile the C program. The following C program will be used as an example:

The C Program (.so version):

```
#include <stdio.h>

writefile(char *path_in,
          char *message_in,
          char *message2_in,
          char *message3_in,
          char *message4_in)
{
    file_name = fopen(path_in, "a");
    fprintf(file_name, "\n%s", message_in);
    fprintf(file_name, "%s", message2_in);
}
```

```

        fprintf(file_name, "\n                %s", message3_in);
        fprintf(file_name, "\n                %s", message4_in);
        fclose(file_name);
    }

```

The C Program (DLL version):

```

// eab.cpp : Defines the entry point for the DLL application.
//

#include "stdafx.h"

BOOL APIENTRY DllMain( HANDLE hModule,
                      DWORD  ul_reason_for_call,
                      LPVOID lpReserved
                      );

__declspec(dllexport) int __cdecl writefile(char,
                                           char,
                                           char,
                                           char,
                                           char);

FILE *file_name;

__declspec(dllexport) writefile(char *path_in,
                                char *message_in,
                                char *message2_in,
                                char *message3_in,
                                char *message4_in)
{
    file_name = fopen(path_in, "a");
    fprintf(file_name, "\n%s", message_in);
    fprintf(file_name, "%s", message2_in);
    fprintf(file_name, "\n                %s", message3_in);
    fprintf(file_name, "\n                %s", message4_in);
    fclose(file_name);
    return 1;
}

```

The C program is creating a function named writefile, which accepts five input arguments: path, message_in, message2_in, message3_in, and message4_in. A file will be opened and the string in the message variables will be printed to the file. The file is opened with an append option to allow the file to be written to multiple times.

Compiling the C Program

To compile and create the shared object library issue a command similar to the following:

```
cc -G -o writefile.so writefile.c
```

The cc command compiles and creates the object code for a shared library named writefile.so. The file that is being compiled is named writefile.c. The shared library will be placed in the current directory, as there is no directory specification in the compile command. The full path and file name may be specified for both the library and the c program.

To create the DLL file, the Microsoft Visual C++ “Build DLL” option was used.

Configuring the SQL*Net Listener

The tnsnames.ora and listener.ora files need to be modified to include information specific to the extproc process. The listener only needs to be configured one time for all future use of external procedures. The following excerpt from the listener.ora file is a sample that can be modified to fit your site.

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = inventory.us.company.com)
      (SID_NAME = ORCL)
    )
    (SID_DESC =
      (SID_NAME = extproc)
      (PROGRAM = extproc)
    )
  )
```

The following excerpt from the tnsnames.ora file is a sample that can be modified to fit your site.

```
extproc_connection_data.world =
  (DESCRIPTION =
    (ADDRESS =
      (PROTOCOL = IPC)
      (KEY = EXTPROC0)
    )
    (CONNECT_DATA = (SID - extproc)
  )
)
```

After the tnsnames.ora and listener.ora files have been successfully modified the listener process needs to be started. From a Unix operating system prompt, issue the following command:

```
$ lsnrctl start
```

Creating a Database Library Object

A database library object that maps to the operating system shared library needs to be created. The library acts as a pointer to the operating system storage location of the external procedure. The CREATE LIBRARY command is used to create a library object in the database. The basic syntax for the CREATE LIBRARY command is:

```
CREATE LIBRARY library_name is
'fullpath_file_name';
```

The *library_name* is the user-defined name for the library database object and must be 30 characters or less in length.

The *fullpath_file_name* is the full path and file name of the operating system shared library. The path and file name are case sensitive and must be enclosed in single quotes.

The following example creates a library named writefile in the database.

```
SQL> create or replace library writefile
      is 'e:\eab.dll';
      /
```

Once the library has been created it can be referenced in the same way that a PL/SQL program unit would be referenced. To determine what libraries have been created, query the data dictionary table user_libraries or dba_libraries.

```
SQL> select owner,
      library_name,
      substr(file_spec,1,30)file_spec
      from dba_libraries
      order by owner,
      library_name;
```

The full path and file name for the operating system library is not verified when the CREATE LIBRARY command is issued. When a program is executed that calls the database library, the path specified for the operating system library will be verified.

The CREATE LIBRARY privilege is required to create a database library. The GRANT command can be issued to grant other users access to a database library object. The following example grants EXECUTE privileges on the library writefile previously created.

```
SQL> GRANT EXECUTE ON writefile TO student1;
```

Creating the PL/SQL Wrapper Procedure

The next step is to create a PL/SQL wrapper procedure that will be used to call the external procedure. The PL/SQL procedure will be used to map PL/SQL parameters to the C parameters. The PL/SQL procedure will also be used to identify the name of the external procedure to PL/SQL. The basic syntax for creating a wrapper procedure is:

```
CREATE OR REPLACE PROCEDURE proc_name
[parameter_list]
AS EXTERNAL LIBRARY library_name
[NAME external_name]
[LANGUAGE language_name]
```

```
[CALLING STANDARD {C|PASCAL}]
[WITH CONTEXT]
[PARAMETERS (external_parameter_list)]
```

The *proc_name* is the user-defined name for the procedure.

The *parameter_list* is the list of input/output arguments required by the procedure.

The *library_name* is the database library object name that represents the operating system library. Note: you must own the library or have EXECUTE privileges on the library to reference it. This is the only required clause.

The *external_name* specifies the name of the external procedure and defaults to the name of the PL/SQL wrapper procedure. Remember that the database stores object names as upper case. If the C procedure name is lower case, double quotes are required around the name.

The *language_name* identifies the language that the external procedure is written in. The default is C. C is currently the only language that is supported.

The *calling standard* determines the order that parameters are placed on the stack. The default is C. With the Pascal Calling Standard, parameters are reversed on the stack.

The *with context* parameter is used with the OCI for database callbacks. A context pointer will be passed to the external procedure.

The *external_parameter_list* is used to specify how the PL/SQL parameters should be mapped to the C parameters.

The following example creates a wrapper procedure for the writefile C procedure:

```
SQL> create or replace procedure writeit
      (path_in varchar2,
       message_in varchar2,
       message2_in varchar2,
       message3_in varchar2,
       message4_in varchar2)
      as external library writefile
      name "?writefile@YAHPAD0000@Z"
      parameters(path_in string,
                 message_in string,
                 message2_in string,
                 message3_in string,
                 message4_in string);
/
```

The PL/SQL Procedure:

The writeit procedure can be called from any PL/SQL program unit. In this case, the procedure will be called from another PL/SQL procedure named printinv. The printinv procedure in turn will be called from an Oracle Form that allows the user to enter the criteria for printing the customer invoice. The printinv procedure is being stored in the database.

```
create or replace procedure printinv
(path_in varchar2,
 id varchar2) is
cursor customer_cursor(id char) is
  select customer_id,
         first_name||' '||last_name name
  from customer
  where customer_id = id;
cursor invoice_cursor(id char) is
  select to_char(depart_date,'DD MON')depart,
         substr(to_char(depart_date,'DAY'),1,3)day,
         rpad(airline_name,20)airline_name,
         rpad(invoice.flight_no,5) flt,
```

```

        substr(fare_basis,1,1)fare_basis,
        rpad(depart_city,19)depart_city,
        to_char(depart_time,'HH:MI AM')depart_time,
        rpad(arrive_city,19)arrive_city,
        to_char(arrive_time,'HH:MI AM')arrive_time,
        ticket_price
from invoice,
    flight_schedule,
    airline
where invoice.flight_no = flight_schedule.flight_no
and invoice.airline_id = airline.airline_id
and cust_id = id
order by depart_date;
begin
for customer_rec in customer_cursor(id) loop
writeit(path_in,customer_rec.name,',','');
for invoice_rec in invoice_cursor(customer_rec.customer_id) loop
writeit(path_in,' ',invoice_rec.depart||'-'||
    invoice_rec.day||' '||
    invoice_rec.airline_name||' FLIGHT: '||
    invoice_rec.flt||' CLASS: '||
    invoice_rec.fare_basis,
    invoice_rec.depart_city||' '||
    ' DEPART: '||invoice_rec.depart_time,
    invoice_rec.arrive_city||' '||
    ' ARRIVE: '||invoice_rec.arrive_time);

end loop;
end loop;
end printinv;
/

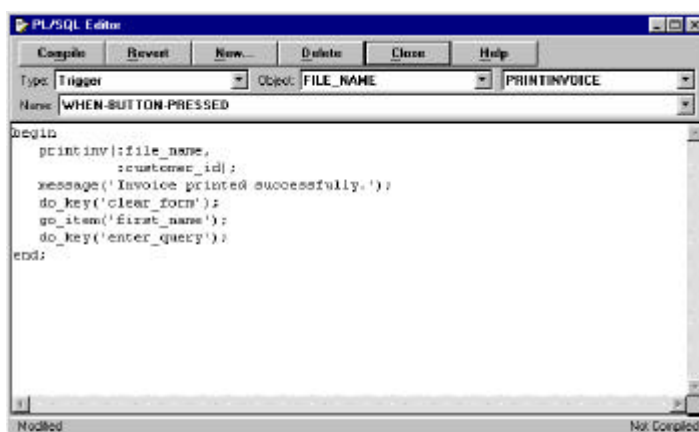
```

In this example, the file name will be passed in to the printinv procedure from the end-user Oracle Forms application. The form will also be used to provide input for the id input argument.

The Oracle Form

An Oracle Form has been created that will call the PL/SQL procedure printinv. The purpose of the form is to allow the end-user a method of selecting a customer and then printing the invoice information for the customer. The invoice information is being written to a flat text file in the location specified by the user. The invoice can then be printed to the desired printer destination.

The Oracle Form consists of two blocks a customer block and a non-table based block for the file name. The customer block will be used to select the customer id from the customer table. The file name block holds the file and directory entered by the user. A button named printinvoice is created that will call the printinv PL/SQL procedure.



Example Form

Following is an example of the form during runtime. The user performs a query in the customer block to display the customer number for the desired customer. Next, the full path and file name for the output file is entered in the File Name item on the form. To execute the procedure printinv (which calls the external procedure writeit), the user single-clicks on the Print Invoice button.

The screenshot shows a Windows 95/NT application window titled "CUSTOMER_WINDOW". The window has a menu bar with "Action", "Edit", "Query", "Block", "Record", "Field", "Window", and "Help". Below the menu bar is a toolbar with various icons. The main area of the window contains a form with the following elements:

- Two text input fields: "First Name" containing "Melinda" and "Last Name" containing "Johnson".
- A text input field: "Customer Number" containing "33333".
- A section titled "Enter the directory and file name for the printed invoice:" containing a "File Name" input field with the text "E:\Johnson.txt".
- A "Print Invoice" button.

At the bottom of the window, there is a status bar that reads "Record 1/1".

Sample Output File

The sample output file is a "txt" file, which can be opened in a variety of word processing applications or could be sent directly to a printer for printing of the invoice. The following example is the text file produced by the external procedure writeit.

Melinda Johnson

```
21 MAR-SUN Continental Airlines FLIGHT: 516 CLASS: Q
ATL DEPART: 07:20 PM
DEN ARRIVE: 10:58 PM
21 MAR-SUN Transworld Airlines FLIGHT: 908 CLASS: Q
STL DEPART: 04:45 PM
DEN ARRIVE: 05:13 PM
04 JUL-SUN Continental Airlines FLIGHT: 517 CLASS: K
ATL DEPART: 06:20 PM
DEN ARRIVE: 08:48 PM
12 JUL-MON Continental Airlines FLIGHT: 617 CLASS: K
DEN DEPART: 01:00 PM
ATL ARRIVE: 06:34 PM
```

Conclusion

An external procedure is simply a program unit that is written in a 3GL and resides outside of the database. Currently, the only 3GL supported by Oracle 8 is the C programming language. After the C program has been compiled a database library object needs to be created. The Database library object is used to map the C shared library to a database object. A wrapper procedure is used to map the PL/SQL arguments to the C arguments. The wrapper procedure is written in PL/SQL. The PL/SQL procedure can be called from any other PL/SQL program unit including procedures and functions.

About the Author

Elizabeth Boss is president of Boss Consulting Services, Inc. She has more than 15 years of experience in application development and database administration as a database administrator, consultant, instructor, and curriculum developer. As a senior consultant/instructor, Elizabeth has worked directly with customers in all phases and aspects of the design, development, and administration of Oracle systems. She is a frequent presenter at international and local user groups, and, in 1997, was listed among the top 25 speakers at the IOUG-A Conference.

Elizabeth can be reached at:

Boss Consulting Services, Inc.

710 Kings Deer Point

Monument, CO 80132

Phone: (877) 489-7745 Fax: (719)481-5820

E-mail: eboss@boss-consulting-inc.com

Web: www.boss-consulting-inc.com