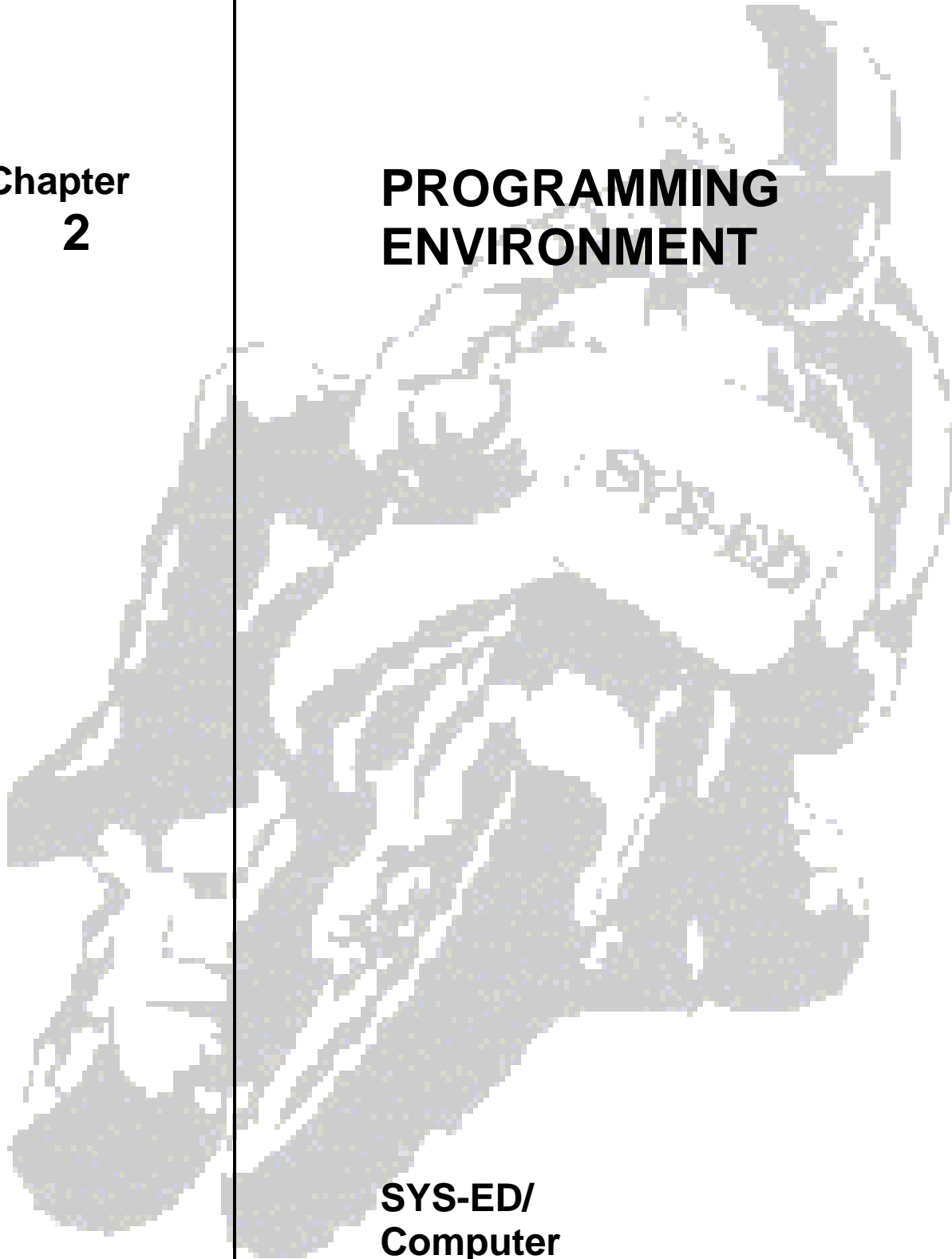


**Chapter
2**

**PROGRAMMING
ENVIRONMENT**



**SYS-ED/
Computer
Education
Techniques, Inc.**

Objectives

You will learn:

- How to use the language environment.
- Setting language environment run-time options.
- MSGFILE.
- RPTOPTS.
- Viewing RUNOPTS settings.
- Implementing security and authorization.



1 Language Environment - LE

LE - Language Environment is a component of the OS/390 and z/OS operating systems.

LE:

- establishes a common run-time environment for different programming languages.
- combines essential run-time services, such as routines for run-time message handling, condition handling, and storage management.

All of these services are available through a set of interfaces that are consistent across programming languages. These interfaces can be called directly or through the language-specific services that call the interfaces.

The Language Environment, can use one run-time environment for applications, regardless as to the application's programming language or system resource needs. DB2 uses LE to provide a run-time environment for stored procedures written in high-level languages: COBOL, PL/I, and C. Stored procedures written in each of these languages can execute in the same stored procedure address space; they can be separated for reasons relating to resource management.

LE performs provides the following functions for DB2:

- Hides the differences between programming languages.
- Provides the ability to make a stored procedure resident in the stored procedure address spaces.
- Supports a large number of run-time options, including those options required for using tools to debug stored procedures.

2 Language Environment Run-time Options

DB2 stored procedures can use existing values for LE run-time options, either the defaults provided with z/OS, or values which have been overridden when setting up the Language Environment.

- Some options can be overridden for the purpose of improved debugging capabilities and better management of storage below the 16 MB line.
- Default values for LE run-time options can be overridden by specifying new values for the options on the RUN OPTIONS parameter of the CREATE PROCEDURE or ALTER PROCEDURE statement.

2.1 MSGFILE Run-time Option

The MSGFILE run-time option specifies the ddname of the file that contains run-time diagnostics for the stored procedure.

The format of the RUN OPTIONS parameter to specify a MSGFILE is:

```
RUN OPTIONS `MSGFILE(ddname , , , ENQ | NOENQ) `
```

The purpose of the MSGFILE option is to provide a destination file for diagnostic messages.

When a stored procedure also contains host language statements that display informational messages, then those messages will also be written to the MSGFILE data set.

When managing diagnostics messages, situations should be avoided where many stored procedures write to the same MSGFILE data set. This could lead to a JES spool serialization error, resulting in an S02A abend with reason C.

2.2 RPTOPTS Run-time Option

The RPTOPTS run-time option generates a report of the run-time options in effect while the application was running. The report is directed to the ddname specified in the MSGFILE run-time option.

This information can be useful when debugging a stored procedure. The resulting report will display the default options specified in the LE environment, as well as any options that have been overridden at the stored procedure level.

The format of the RUN OPTIONS parameter to specify RPTOPTS is:

```
RUN OPTIONS `RPTOPTS(ON)'
```

The RPTOPTS option should only be utilized when there is a requirement to only understand what run-time options are in effect while testing a stored procedure.

The RPTOPTS option should be set to OFF when running in production; the report generation process will increase the time it takes to run the stored procedure.

3 Options to Limit Storage Required by LE at Execution Time

The following LE run-time options can be specified to minimize storage usage below the 16 MB line.

LE Run-time Option	Purpose
HEAP(,ANY	Allocate program heap storage above the 16 MB line.
STACK(,ANY,	Allocate program stack storage above the 16 MB line.
STORAGE(,,4K	Reduce reserve storage area below the line to 4 KB.
BELOWHEAP(4K,,	Reduce the heap storage below the line to 4 KB.
LIBSTACK(4K,,	Reduce the library stack below the line to 4 KB.
ALL31(ON	Indicate all programs contained in the stored procedure run with AMODE(31) and RMODE(ANY).

4 Viewing RUNOPTS Settings

The RUN OPTIONS for a DB2 stored procedure are set when a CREATE PROCEDURE or ALTER PROCEDURE statement is executed.

The options specified in the DDL are stored in the system catalog in table SYSIBM.SYSROUTINES in the column RUNOPTS.

An empty string in the RUNOPTS column means that the z/OS default values or installation supplied values for that language are used for the LE run-time options for the procedure.



5 Security and Authorization

Stored procedures are objects that are maintained by DB2. They are objects just like other objects such as tables, views, packages, and plans.

Access to stored procedures is controlled by the privileges that have been granted to the authorization IDs that are requesting access.

5.1 Workload Manager Security Requirements

There are two external levels of security that have to be set up to for WLM stored procedures:

- Each authid that attempts to create a stored procedure needs the authority for creating stored procedures in the WLM environment where the procedure will be run.
- Application developers or DBAs that need to refresh WLM application environments must be permitted access to the DB2-supplied stored procedure for refreshing the address spaces.

5.2 Privileges Required to Create Stored Procedures

Stored procedures are typically created by a DBA or an application programmer, based upon how roles and responsibilities are defined for an organization.

The DBA or application programmer requires certain DB2 privileges in order to create stored procedures.

5.3 CREATEIN Privilege on the Schema

A schema is a collection of named objects such as stored procedures, triggers, and user-defined functions. When a stored procedure is created, it is implicitly or explicitly qualified by a schema.

When a stored procedure is created, it is given a three-part name:

- The first part is the location, or DB2 subsystem, where the stored procedure is defined. The location can be implicitly or explicitly specified.

If the location is left blank it defaults to the subsystem on which the CREATE PROCEDURE statement is issued.

- The second part of the name is the schema name; it can be implicitly or explicitly specified.

If the schema is left blank, it defaults to the current authid in effect for the person issuing the CREATE PROCEDURE statement.

The following SQL statement allows authid PAOLORW to create stored procedures in the development environment:

```
GRANT CREATEIN ON SCHEMA DEVL7083 TO PAOLORW
```

There may be many application developers or DBAs who create stored procedures into the same schema. Accordingly, it may be a good practice to grant the CREATEIN privilege on the schema to a secondary authid which represents a group of users who create stored procedures.

5.4 BINDADD Privilege for Stored Procedures that Contain SQL

When the stored procedure being created contains SQL statements, then a package will be created and stored in the DB2 catalog. The BINDADD system privilege is required to create new packages in a DB2 subsystem.

The SQL for granting BINDADD privilege to authid PAOLORW is:

```
GRANT BINDADD TO PAOLORW
```

Since the BINDADD privilege is a system level privilege, the GRANT statement only needs to be issued once per authid for a subsystem. Rather than grant BINDADD to every individual who can create stored procedures, the privilege can be granted to a secondary authid, which represents a group of users who create stored procedures.

5.5 Privileges Required to Execute Stored Procedures

Once a stored procedure has been created, it will most likely be executed by a number of DB2 users.

Two types of authorizations are required:

Authorization to execute the CALL statement.

The privileges required to execute the CALL depend on several factors including the way the CALL is invoked. The CALL can be dynamic or static.

- The dynamic invocation of a stored procedure is whenever the CALL is executed with a host variable:

```
CALL :host-variable
```

- The static invocation of a stored procedure is whenever the CALL is executed with a qualified procedure name:

```
CALL procedure-name
```

Authorization is required for executing the stored procedure package and any dependent package.

The privileges required to EXECUTE the package is independent of the type of CALL.

5.6 Privileges to Execute a Stored Procedure Called Dynamically

For static SQL programs that use the syntax CALL host variable, the authorization ID of the plan or package that contains the CALL statement must have one of the following:

- The EXECUTE privilege on the stored procedure.
- Ownership of the stored procedure.
- SYSADM authority.

The following SQL statement is then executed to grant the EXECUTE privilege on the stored procedure to the client authid:

```
GRANT EXECUTE ON PROCEDURE DEVL7083.EMPDTLSC TO PAOLORW
```

5.7 Privileges to Execute a Stored Procedure Called Staticly

For static SQL programs that use the syntax CALL procedure, the owner of the plan or package that contains the CALL statement must have one of the following:

- The EXECUTE privilege on the stored procedure.
- Ownership of the stored procedure.
- SYSADM authority.

It does not matter whether the current authid at execution time has the EXECUTE privilege on the stored procedure. As long as the authid has EXECUTE authority on the plan or package of the calling application it will be able to execute any CALL statements within the calling application.

Unless VALIDATE(RUN) is used, this privilege is checked at the time the plan or package for the calling application is bound.

The following SQL statement is then executed to grant the EXECUTE privilege on the package for the calling application to authid PAOLORW:

```
GRANT EXECUTE ON PACKAGE DEVL7083.CALDTLSC TO PAOLORW
```

6 Limiting the Types of SQL that can be Executed

One of the options on the CREATE PROCEDURE statement is MODIFIES SQL DATA.

The valid values are:

- MODIFIES SQL DATA
- READS SQL DATA
- CONTAINS SQL DATA
- NO SQL

This option can be used to control the types of SQL statements that may be executed within the stored procedure.

A stored procedure created with the option READS SQL DATA cannot include an INSERT, UPDATE, or DELETE statement.