Part III

Utensils

Chapter 4

Remote Operations Stub-Generator

The *rosy*(1) program is a compiler for specifications of remote operations.

For those interested in trivia, *rosy* stands for Remote Operations Stub-generator (YACC-based). The *yacc*(1) program, of course, is a compiler-compiler.

4.1 Remote Operations Module

The *rosy* program reads a description of a *remote operations module* and produces the corresponding *C* stubs and definitions for use with the run-time environment provided by the *librosy*(3n) library. The syntax of the input file corresponds almost exactly to the remote operations notation described in any of the references found in Section 3.7 on page 84 of *Volume One*. (But see Section 4.3 on page 36 for the details.) In short, three kinds of objects are defined: *operations*, *errors*, and *types*.

An example of a complete remote operations module can be found in Figure 4.1 starting on page 34. For now, let's briefly consider the three individual objects.

4.1.1 Operation Definitions

Operations are defined by a name which starts with a lowercase character followed by the keyword `OPERATION`. Following this are optional definitions for the operation:

`ARGUMENT:` defines the ASN.1 type which the operation expects as its argument;

`RESULTS:` defines the ASN.1 type which the operation returns on success; and,

`ERRORS:` defines the errors which the operation can return on failure.

These optional definitions must occur in the order presented here. Following any optional definitions, the lexeme "`::=`" follows with an integer which uniquely identifies the operation in the module. By convention, the integer, termed the *operation code*, should be non-negative.

4.1.2 Error Definitions

Errors are defined by a name which starts with a lowercase character followed by the keyword `ERROR`. Following this are optional definitions for the operation:

`PARAMETER:` defines the ASN.1 type which the error returns when it occurs.

Following any optional definitions, the lexeme "`::=`" follows with an integer which uniquely identifies the error in the module. By convention, the integer, termed the *error code*, should be non-negative.

4.1.3 Type Definitions

Types are defined by a name which starts with an uppercase character followed by the lexeme "`::=`" and then an ASN.1 type definition. As the ASN.1 language is rich in its notation, the rules are not presented herein. The reader should consult [ISO87, CCI88c] for the full details, though Section 6.2.1 on page 60 of *Volume One* presents an incomplete treatment of the less substantive aspects of the language.

```
IMAGE DEFINITIONS ::=

BEGIN

-- operations

initialize OPERATION
        ARGUMENT    Initialize-INV
        RESULT      Initialize-RSP
        ERRORS      { congested, versionNotSupported }
        ::=         1

readImage OPERATION
        ARGUMENT    ReadImage-INV
        RESULT      ReadImage-RSP
        ERRORS      { congested, noMatches, formatNotSupported }
        ::=         2


-- errors

congested ERROR
        ::=         1

versionNotSupported ERROR
        PARAMETER   VersionsSupported
        ::=         2

noMatches ERROR
        ::=         3

formatNotSupported ERROR
        PARAMETER   FormatsSupported
        ::=         4


-- types
```

```
Initialize-INV ::=
        SEQUENCE {                                                    40
           proposed[0]
                IMPLICIT Version
                DEFAULT { version-1 }
        }

Initialize-RSP ::=
        SEQUENCE {
           negotiated[0]            -- need not be present if only one
                                    -- version was requested and that was chosen,
                                    -- otherwise exactly one bit is set   50
                IMPLICIT Version
                OPTIONAL
        }

VersionsSupported ::=
        SEQUENCE {
           supported[0]
                IMPLICIT Version
        }
                                                                      60
Version ::=
        BITSTRING { version-1(0) }


ReadImage-INV ::=
        SEQUENCE {
                                    -- in order of preference for searching
           SEQUENCE OF
                CHOICE {
                   file-name[0]                                       70
                        IMPLICIT Filename,

                   mail-address[1]
                        IMPLICIT Mail-Address
                },

                                    -- in order of preference for retrieval
```

```
            SEQUENCE OF
                Format
        }

ReadImage-RSP ::=
        SEQUENCE {
            CHOICE {
                x-windows
                    X-Windows-Bitmap
            }
        }

FormatsSupported ::=
        SEQUENCE {
            supported[0]
                IMPLICIT Format
        }

Format ::=
        BITSTRING { x-windows-bitmap(0) }


-- discriminants

Filename ::=
        [APPLICATION 1]
            IMPLICIT GraphicString

Mail-Address ::=
        [APPLICATION 2]
            IMPLICIT SEQUENCE {
                local[0]
                    IMPLICIT GraphicString,

                domain[1]
                    IMPLICIT GraphicString,

                options[2]
                    IMPLICIT BITSTRING { default-local(0), default-host(1) }
```

```
                DEFAULT { default-local, default-host }
        }

                                                                    120
-- image formats

X-Windows-Bitmap ::=
        [APPLICATION 100]
          IMPLICIT SET {
                width[0]
                  IMPLICIT INTEGER,

                height[1]
                  IMPLICIT INTEGER,                                 130

                hot-spot-x[2]
                  IMPLICIT INTEGER
                  DEFAULT -1,

                hot-spot-y[3]
                  IMPLICIT INTEGER
                  DEFAULT -1,

                data[4]             -- padded to short boundaries   140
                  IMPLICIT OCTETSTRING
        }

END
```

Figure 4.1: Example of a Remote Operations Module

4.2 ROSY Environment

The *rosy* program will produce several files after reading its remote operations module.

4.2.1 Abstract Syntax Module

The abstract syntax module produced by *rosy* is simply a copy of the type definitions found in the remote operations module. In addition, *rosy* may define a few new types based on the definitions used in the operations and errors.

4.2.2 C Language Stubs

Three separate stub files are produced by *rosy*.

The first is a *C* pre-processor file containing definitions of:

- each operation code

  ```
  #define operation_MODULE_OPNAME OPCODE
  ```

  as in:

  ```
  #define operation_IMAGE_initialize 1
  ```

- each error code

  ```
  #define error_MODULE_ERNAME ERRCODE
  ```

  as in:

  ```
  #define error_IMAGE_congested 1
  ```

- stub routines

  ```
  #define op_MODULE_OPNAME(sd,in,out,rsp,roi) \
                                    RyOperation ( ... )
  ```

  as in:

  ```
  #define op_IMAGE_initialize(sd,in,out,rsp,roi) \
                                    RyOperation ( ... )
  ```

The stub routines require a lot more explanation. Rather than introduce the full details, for the moment simply note that the stub routines are actually calls to the `RyOperation` routine discussed in Chapter 8 which "does all the work".

The second file produced by *rosy* defines the *C* language data structures used by the run-time environment which the *librosy*(3n) library provides.

The third file produced by *rosy* is used in conjunction with *lint*(1). It defines as routines, rather than macros, the same stub routines found in the *C* pre-processor file. These routines permit more stringent argument checking.

4.3 Known Deficiences

The *rosy* program is a prototype. It probably won't undergo substantive change, but experience will tell.

In particular, *rosy* uses essentially the same front-end as the *pepy*(1) program, so it has some limitations in the ASN.1 syntax it can accept. Consult Section 6.3.1 for the details.

The `BIND` and `UNBIND` macros are currently unrecognized, along with the newer (`OBJECT IDENTIFIER`) notation for operation codes.

The `LINKED` clause is currently recognized but otherwise ignored for the `OPERATION` macro.

4.4 Running ROSY

Here are a few things to know when running *rosy*.

4.4.1 Options

The *rosy* program has a few options to modify its behavior.

The '`-b`' switch sets the prefix used for the output file. If this option is not given, the ASN.1 module name is used.

The '`-d`' switch directs *rosy* to ignore any *pepy*-style augmentations.

The '`-o`' switch sets the name of the output file, which is normally derived from the name of the input file. The distinguished file "`-`" can be used to force the use of the standard output.