Identification

The FORTRAN Command
L. L. Garthe, K. C. Shih

Purpose

This section describes the FORTRAN command for Prototype MULTICS. A user invokes this command to compile standard text and linkage segments (one of each, symbol seg deferred) from a segment containing a single FORTRAN symbolic source program.

For details on the language see BZ.4.01. For an overview of the compiler itself see BZ.4.02.

Usage

When a user types a command line of the following form:

```
          fortran alpha
```

the FORTRAN command is invoked by the Shell (MSPM BX.2). In the format above, "alpha" represents a user selected segment name.

The symbol "alpha" may be either a path name or an entry name and specifies that segment <alpha.fortran> is the source program to be compiled. If alpha is an absolute path name it is assumed to be relative to the root directory. If alpha is a relative path name it is assumed to be relative to the current working directory.

Optimization Option

The use of the option "optim" will cause the compiler to attempt extensive code optimization on a statement by a statement basis. See BZ.4.02 for more detail; see BX.12 on options in general.

System Option Commands

FORTRAN will initially observe the "list" option only. Implementation of "brief" is deferred. These options may be "interjected" and are interpreted as per BX.0.00 and BX.12.00.

General Description of Operation

The FORTRAN command as described here is essentially an EPL control program which is called by the Shell and which in turn sets up and calls the actual compiler (see figure 1). The control program analyzes the command arguments, sets up all segments necessary to run the compiler, queries appropriate options, sets corresponding internal switches, and calls the compiler.

The compiler is written in an interpretive language called POPS. Therefore, the compiler essentially consists of 1) a segment containing the compiler logic, which is sometimes called the "upstairs", 2) a segment containing the 645 POPS interpreter, sometimes called the "downstairs", and 3) a data segment used for storage of compiler variables, most of which are contained in "rolls".

When control returns to the command from the compiler, various output segments are assigned names and registered in the file system, checks being made to insure uniqueness of symbolic segment names in the file system hierarchy.

Any working segments are disposed of, the error segment, if present, is typed out on the terminal and control is returned to the Shell to await the next user command.

Method

Preparation Prior to Assembler Call

1. Upon being called by the Shell, the first action of the FORTRAN command is to analyze the elements in the array of strings passed (in effect the command arguments) by the Shell to the command. The resulting information controls the handling of the optional optimizing argument.

2. The FORTRAN command then invokes the unique_chars procedure (MSPM BY.15.01) to construct the following symbolic segment names to be used during assembly:

unique_name.text	(output)
unique_name.link	(output)
unique_name.symbol	(output, deferred)
unique_name.error	(output)
unique_name.fortran_data	(used during assembly)
unique_name.list	(output, optional)

3. The FORTRAN command then calls the smm$set_name_status procedure (MSPM BD.3.02) to create an empty segment in the process directory named unique_name.fortran_data. The compiler data segment, containing initial values, is located, copied into this new segment and released. The new segment will serve as working storage for the compiler, containing all variables used during compilation, as well as serving as a communication medium between the command and the compiler. After compilation this segment will be discarded.

4. Using the symbol "alpha" as the input argument, the command next calls the entryarg procedure (MSPM BY.2.04) to determine the path name and entry name for <alpha.fortran> which contains the FORTRAN symbolic source program to be compiled. A pointer to <alpha.fortran> is then established by calling the smm$initiate procedure (MSPM BD.3.02), and stored in the data segment (established in 3. above).

   If a pointer to the segment <alpha.fortran> cannot be created, error type 1 (see discussion on errors below) is typed out on the terminal, the compiler data segment is discarded and control is returned to the Shell for corrective user action.

5. Using procedures outlined above, the next step is to get a pointer to the FORTRAN upstairs segment and store it in the data segment.

6. Successive calls to the smm$set_name_status procedure (MSPM BD.3.02) are then executed to create the following empty data segments in the process directory:

<unique_name.text>	to contain the equivalent object program text for the source program <alpha.fortran>
<unique_name.link>	to contain the linkage information for alpha
<unique_name.error>	to contain the errors found during compilation

   Pointers to each the above are stored in appropriate positions in the compiler data segment.

   If any of the above segments cannot be created, error type 2 is reported.

7. The command now obtains the setting of the standard system list option by a call to the read_opt procedure.

   If the list option is on, a new segment is created and assigned the name unique_name.list. A pointer to this segment is established and placed in the compiler data segment. This position is set to null if the list option is off. Again, error type 2 may be reported.

The Compiler Call

The compiler is now ready to be called. Since it utilizes the 645 POPS interpreter, the call (in EPL) is:

```
        call pops_645 (data_ptr)
```

where "data_ptr" is a pointer to the compiler data segment, which contains all information, segment pointers, etc. necessary to properly drive the compilation.

After Return From Compiler To Command

1. Although not yet clearly specified, there will be a switch available in the compiler data segment to allow the compiler to notify the command program of disastrous termination of compilation. This is the first thing checked by the command upon return from the compiler. If on, the command will dispose of the various unique_name segments, provide error output as outlined in more detail below and return to the Shell.

2. The entry_status$type procedure (MSPM BY.2.10) is then invoked to determine the status of potential segments with the following segment names, in the file system:

<alpha>	
<alpha.link>	
<alpha.symbol>	(deferred)
<alpha.error>	
<alpha.list>	(if list option on)

Note that segments with the above names may have been available in the file system prior to the request for the compilation of the source program in <alpha.fortran>.

If any of the names identify "old" segments which are links or directory branches in the file system, error type 3 is reported.

If an "old" segment is not present, the change_name procedure is invoked to change the name of the uniquely named segment just created to the corresponding "alpha" name from the above list. For example, if the compiler just created <unique_name.error>, it will be changed to <alpha.error>, to be made available to the user.

If an "old" segment exists as a branch (non directory), the following steps are executed to return the new segment to the user:

(a) The truncate_seg procedure (MSPM BY.2.01) is invoked to discard the contents of the "old" segment.

(b) The move_file procedure is invoked to move the contents of the "new" (e.g. unique_name) segment to the empty "old" segment.

(c) The delete_entry procedure (MSPM BY.2.01) is invoked to delete the uniquely named segment.

(Note: the above referenced move_file procedure does not actually involve a physical copy process and is therefore not as inefficient as it appears.)

If there is difficulty in moving or renaming segments in the above, error type 4 is reported.

3. The contents of <alpha.error> are then typed by a call to write_out (MSPM BY.4.01).

   If for some reason the contents of <alpha.error> cannot be typed, error type 5 is reported.

4. The delete_entry procedure (MSPM BY.2.01) is invoked to delete unique_name.fortran_data.

5. Control is then returned to the Shell to await the next user command.

Errors

Command Errors

In the above discussion various error conditions that may arise while control is in the FORTRAN command have been briefly discussed. In general these errors have to do with improper usage of segments or names, as specified by the command typed in by the user, and usually result in rejection of the command. Consequently, when such errors occur, any segments that may have been established are destroyed prior to the invocation of the standard error handling mechanism (MSPM BY.11) for reporting such abnormalities.

The condition "fortran_err" is reported for the following errors:

Error No.	Meaning
1	error in attempt to obtain a pointer to the following segment: <"segment_name_filled_in">
2	error in trying to create the following segment: <"segment_name_filled_in">
3	error in trying to delete the following segment: <"segment_name_filled_in">
4	error in trying to move or rename the following segment: <"segment_name_filled_in">
5	error in trying to type the contents of the following segment: <"segment_name_filled_in">

Compiled Errors

Errors encountered by the compiler in the processing of the FORTRAN source text normally do not result in termination of compilation. Such errors are placed in the error segment which is typed out after the compiler returns control to the command.

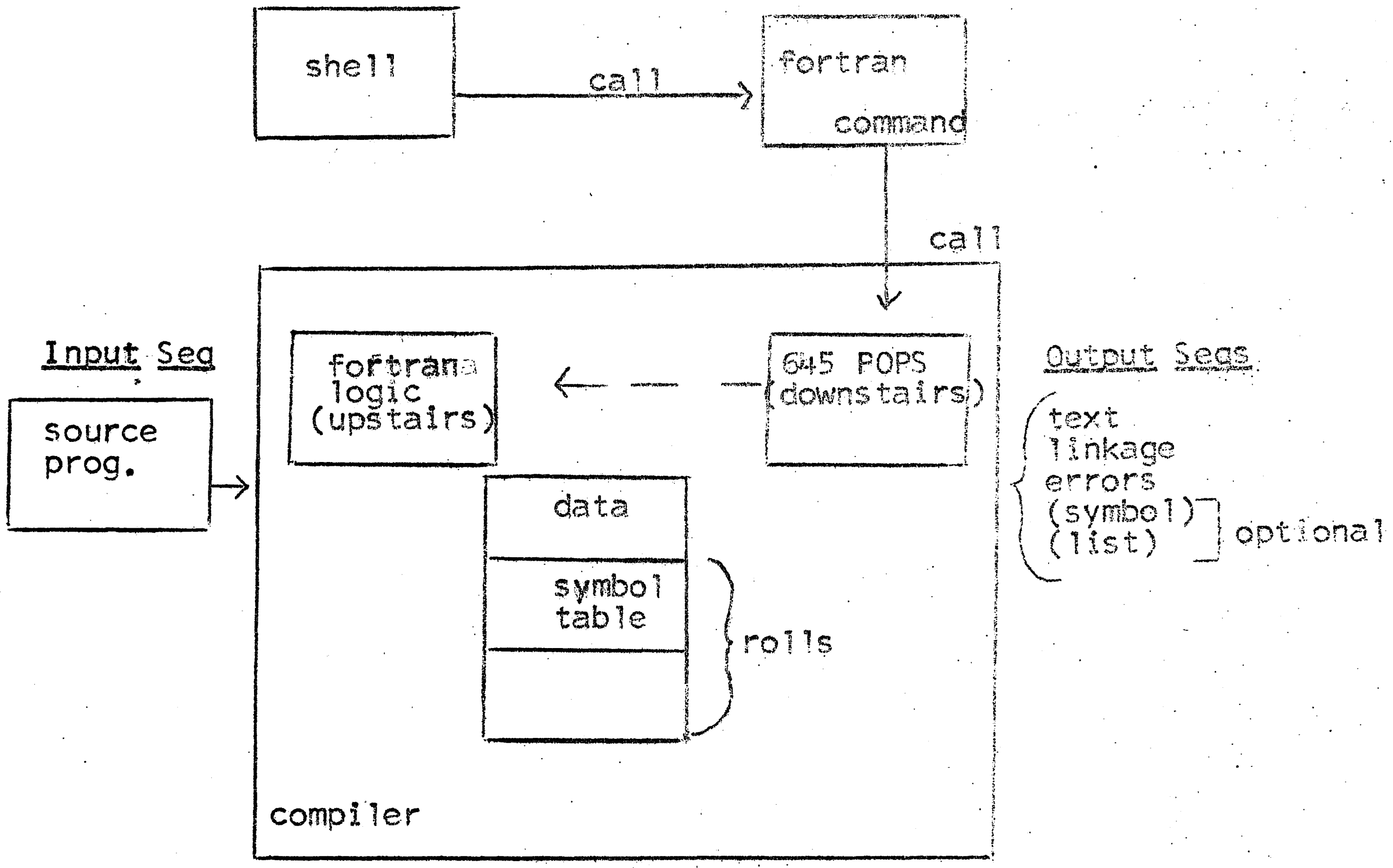
shell
call
fortran command
call
Input Seg
source prog.
fortran logic (upstairs)
645 POPS (downstairs)
Output Segs
text
linkage
errors
(symbol)
(list)
optional
data
symbol table
rolls
compiler

FIGURE 1