MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

GARCHING BEI MÜNCHEN

Development of FORTRAN Programs Using an AMOS Terminal

Friedrich Hertweck

IPP R-29 September 1978

IPP R/29 Sep 14, 1978

Development of FORTRAN Programs Using an AMOS Terminal

Friedrich Hertweck

Development of FORTRAN Programs Using an AMOS Terminal

0. Introduction

The IPP Computer Center is equipped with two coupled processors IBM 360/91 and AMDAHL 470V/6. On the 91, an OS/360 MVT system is used (Release 21.6), on the 470 the VM/370 system is used to run several virtual machines, in particular the AMOS and an MVT machine. In addition, there are other virtual machines, like the CMS machines, Reduce, etc. However, there is no general CMS service available and in this note we are only concerned with the AMOS/MVT system.

AMOS is a File Management and File Editor System. Users may create FORTRAN jobs as AMOS files and submit them to either MVT system (MVT91 or MVT470) where they are processed as batch jobs. Apart from normal OS/360 data sets and tapes a FORTRAN job may also use AMOS files for input or output. The AMOS system prevents more than one job modifying an AMOS file, though several jobs may read a file.

With the catalogued JCL procedures FOG1CA and FOHXCA it is possible to compile a FORTRAN program (or individual subroutines or batches of subroutines) and placing the object program(s) into an AMOS file segment. Should the compilation fail then instead of the object program the user will find the compiler error messages in that file segment.

A FORTRAN program consisting of a batch of object modules may be executed using the catalogued procedure FOX. A HASP command will direct the output of the job to a remote batch terminal line printer or it may be placed into an AMOS file segment for inspection from the terminal.

The catalogued procedures AFORT, AFORTH, and AJOB may be used to minimize the job output (no source program lists or loading maps) and to make the directing of job output to AMOS file segments as easy as possible.

The last section contains a few remarks that may prove useful for the novice computer and AMOS user.

1. General Remarks on Using the AMOS System

The user must have an AMOS identification (usually a three-letter code showing his initials) and an account number assigned to him. If he wants to obtain job output at a terminal he must generate a file called AMOSOUT and admit via the USER subcommand of the FILEORG command the user AMOS to write into this file:

```
GENERATE AMOSOUT,SIZE=n
FILEORG
USER *AMOS
```

Apart from this file the user may generate any number of files he wants, subject to his space restrictions. (For details of how to use the AMOS system, cf. the Programmer's Handbook, section F.)

2. Compiling FORTRAN Programs

The newer versions of the IBM FORTRAN compilers have the facility to produce error messages suited for display at the terminal. When used with the appropriate catalogued procedures, either the object output or the error messages may be directly routed into an AMOS file segment.

2.1 Using Catalogued Procedures FOG1CA or FOHXCA

The user should create a file segment containing the following commands:

```
// EXEC FOG1CA,USERID=uid,FILE=filename,NAME=segname
{   FORTRAN source code    }
```

The following general rules apply: Capital letters denote those parts of the command that have to be typed exactly as shown, lower case letters denote quantities to be specified by the user. The FORTRAN source code may be given either explicitly or by a sequence of $$ substitute commands.

If the above file segment is submitted as an express job (using the command XSUBMIT, or XS for short), it will be compiled by the FORTRAN IV G1 compiler.

The following default parameters are used:

```
NOLIST, SOURCE, NODECK, NOMAP
```

The object module(s) will be placed into the AMOS file segment

```
uid:filename.segname
```

if the compilation was successful, i.e. the condition code was 0 or 4.

If the compilation failed (i.e. condition code >4) then the AMOS file segment will contain the terminal error messages generated by the compiler.

There are two ways for the user to find out whether the compilation was successful or not:

(1) If he follows the progress of his job by using occasionally the AMOS command JOBS, he may see whether the compile step C is followed by the step OBJ (copy object module(s) into AMOS file) or by step TERM (copy compilation error messages into AMOS file)

(2) After the job has terminated the user may list the last line of the segment by

```
LIST filename.segname<*>
```

It contains either the last card of the object module in a format similar to

```
END            15734-F02  020078219
```

or the last line of the error messages in a format like

```
* 005 DIAGNOSTICS GENERATED, HIGHEST SEVERITY CODE IS 8
```

If error messages were generated, the user should list the complete segment, obtaining something like this:

```
 100 G1 COMPILER ENTERED
 200      510         POL:=POLY*X+C(1)
 300                   $
 400 01)  IGI013I SYNTAX
 500      600       DO 3 JJ=1,JJ1
 600
 700        $
 800 01)  IGI013I SYNTAX
 900      780         I2=I1+99    *(
1000
1100        $
1200 01)  IGI013I SYNTAX
1300      910         B=C(K)
1400        $
1500 01)  IGI002I LABEL
1600 IGI022I    UNDEFINED LABEL
1700   1234
1800 SOURCE ANALYZED
1900 PROGRAM NAME = MAIN
2000 * 005 DIAGNOSTICS GENERATED, HIGHEST SEVERITY CODE IS 8
```

The first column of numbers are the line numbers of the AMOS segment. The line numbers of the original source code appear as the second column of numbers (in the example above, 510, 600, 780, and 910). With the help of the program name appearing at the bottom of the sublist for each compilation, the user may locate the erroneous lines in his source segment.

If the FORTRAN H-extended compiler is to be used, the catalogued procedure FOHXCA must be used.

2.2 Using Catalogued Procedures AFORT or AFORTH

The method described in the previous section has two drawbacks:

(1) An output list is generated at the central site

(2) The source program is reproduced in the output list

Normally, the output list printed centrally is of no use to the user at the terminal, because the source code is available to the user at the terminal anyway.

The alternative JCL procedures AFORT and AFORTH may be used to suppress the source listing and with an additional HASP control command direct the output of the job into the user's file AMOSOUT. The job will then look as follows:

```
/*ROUTE  PRINT REMOTE13
// EXEC AFORT,OBJ='progfile.objseg'
{  FORTRAN source program  }
```

The object module (or list of error messages if the compilation failed) is placed into the user's file progfile as segment objseg. The file AMOSOUT.jobname contains the HASP and MVT log messages and possibly error messages. The source code listing is suppressed.

3. Execution of FORTRAN Object Programs

If no special requirements have to be taken into account, the linking loader should be used to execute FORTRAN programs (it is simpler to use and more efficient than the Linkage Editor). We will discuss several versions of job setups.

3.1 Normal Batch Execution

In normal batch execution the job is submitted by either the XSUBMIT or the SUBMIT statement. The XSUBMIT (abbreviation XS) will use the following default parameters:

- execution time limit = 10 sec
- region size = 120 K

The job looks as follows (it is contained in an AMOS file segment of the user):

```
//  EXEC FOX
//G.SYSLIN DD *
{ $$ command(s) pointing to AMOS file segment(s) }
{ containing object modules, in any order        }
//G.SYSIN DD *
{ explicit input data or                         }
{ $$ command(s) pointing to segment(s) with data }
```

The loader will use the FORTRAN subroutine libraries

(1) SYS1.FORTLIB
containing the standard subroutines of FORTRAN

(2) SYS1.SSPLIB
containing further subroutines from the Scientific Subroutine Package (they are listed in the Programmer's Handbook in chapter C3.2)

to supply any missing subroutines, including the I/O interface package. The loader will use the following default options:

```
PRINT, MAP, LET, CALL, RES, NOTERM
```

Any printer output by the job, produced by WRITE(6,format) statements, will be spooled and by default printed at the central computer site at Garching on one of the line printers. In fact, four data sets are concatenated and printed as one "output list". They are:

(1) HASP log messages

(2) MVT log messages

(3) Loader output

(4) Output of user's program

3.2 Alternate Routing of Printer Output

There are two additional ways to handle the printer output of the job:

(1) to have it printed on a remote line printer

(2) to have it moved into a segment of the user's AMOS file AMOSOUT for inspection at a terminal

In both cases the method to obtain the desired result is the same: the first line of the job should be

```
/*ROUTE  PRINT REMOTEn
```

where n denotes the Remote Job Station (RJE Terminal). The user should use the appropriate number for the terminal he wants to use. However, for certain CRT terminals the ROUTE command is automatically added and hence the job is printed by default at the user's nearest RJE Terminal.

If n=13, the printer output is moved into a segment of the user's file AMOSOUT (cf. section 1). The segment is automatically generated by the system. Its name is identical to the jobname of the user, as indicated by AMOS after a SUBMIT or XSUBMIT command, for example:

```
JOB FRH789 SUBMITTED FOR EXECUTION
$*16.55.12        MVT-470
JOB 117 FRH789  AWAITING EXEC F  3 PRIO 12
OK
```

After the job has completed (reply to Job command is "JOB NOT FOUND") the user would type in the AMOS command

```
LIST AMOSOUT.FRH789
```

in the above example. Due to currently existing limitations of the AMOS system, the printer output must be converted into card images. A line with not more than 72 characters will be converted into one card image, with the label being a multiple of 10. If a print line is longer than 72 characters, the remaining up to 61 characters are converted into a second card image with the label incremented by one. All printer control characters are discarded (cf. B&B no. 85).

The user should keep in mind that gradually his file AMOSOUT will overflow. He therefore must not forget to purge the segments not longer needed. Alternatively, from time to time, he may purge the whole file and generate it again.

3.3 Job Output written into AMOS File

The user may choose to write some of his formatted output into an AMOS file. The output produced must be compatible with the card image format, i.e. a line must not contain more than 72 characters.

In order to achieve the desired result the user must

(1) include the AMOS I/O procedures into his program

(2) supply an appropriate DD statement for the file reference number used

Let us assume the user wants to put his standard output file (FT06F001) into an AMOS segment. Then his job would for instance look as follows:

```
/*ROUTE  PRINT REMOTE13
//  EXEC FOX
//G.SYSLIN DD DSN=SYS1.FORTLIB(AMOSIO),DISP=SHR
//          DD *
$$ ABC:PROGFILE.OBJECT
//G.PRINT DD UNIT=DISK,VOL=SER=AMOS,DISP=SHR,
//             DSN='W: ABC:WORKFILE.OUT<1.1>;'
//G.SYSIN DD *
$$ ABC:DATAFILE.DATA
```

Here we have assumed that the user ABC had his program compiled previously and had the object module(s) placed into file segment PROGFILE.OBJECT. The DD-statement with ddname G.PRINT directs the standard printer output (file reference number = 6) to the user's AMOS segment WORKFILE.OUT and indicates that the output lines (in the format of card images!) are to be labelled 1, 2, 3,... .

3.4 Using the catalogued Procedure AJOB

If the procedure explained in the previous section 3.3 is to be used as a standard, the user may use the catalogued procedure AJOB (="AMOS Job") instead which contains all these modifications by default.

His job will then look as follows:

```
/*ROUTE  PRINT REMOTE13
//  EXEC AJOB,LIST='listfile.listseg'
//OBJ DD *
$$ userid:progfile.objseg
//DATA DD *
$$ userid:datafile.dataseg
```

All HASP, MVT, and other system messages will go into AMOSOUT.jobname. The output from the FORTRAN job itself will go into the AMOS file specified in the LIST parameter. The fourth line specifies the object program to be executed, the last line specifies the input to the job.

With this procedure the loader will not produce any output.

4. Further Remarks on Using AMOS

In this section we add a few remarks that may help the user to use the system more efficiently.

4.1 Printing a Time Stamp on the Output File

It is recommended that a time stamp is being placed in the job output file. This will prevent the user from misinterpreting his LIST output (which always goes into the same file segment) should his job have failed.

A typical error that may occur is the attempt to execute a job after a failed compilation. The loader would reject the job, telling so in the AMOSOUT.jobname segment produced by the job. However, the user print file 6 would be left unchanged and will show the results of the previous run.

In order to avoid this problem, the user should let his FORTRAN program begin with:

```
      REAL D(6)
      CALL DATE(D)
      WRITE(6,100) D
  100 FORMAT(3(2A4,4X)/)
```

This will produce as the first line of the list segment (for example):

```
24.08.78    10.37:42    ABC707
```

4.2 Some Remarks on Program Organization

For longer programs it is recommended that the user exploits the capabilities of FORTRAN and compiles smaller parts of the program separately. He may create a set of N segments P(n) of the form

```
    /*ROUTE  PRINT REMOTE13
    //  EXEC AFORT,OBJ='WORK.OBJ(n)'
    {   part n of FORTRAN source code    }
```

for n = 1,2,...,N. Whenever a subroutine in part P(k), say, has been modified the editing is terminated with AMOS command

```
         XSUBMIT
```

which will produce a new object module OBJ(k).

The new version of the program can then be executed by submitting the segment

```
    /*ROUTE  PRINT REMOTE13
    // EXEC AJOB,LIST='WORK.L'
    //OBJ DD *
    $$ ABC:WORK.OBJ(1)
             .
             .
             .
    $$ ABC:WORK.OBJ(N)
    $$ XYZ:OBJLIB1.POISSON
    //DATA DD *
    $$ ABC:WORK.PARM
```

In this example user ABC includes the POISSON solver POISSON of user XYZ who keeps the object module in his file OBJLIB1.

4.3 Handling COMMON Statements

The user should also use the $$ commands to include COMMON and other globally used declarations in his subroutines. Then changes of these declarations will immediately be propagated into the whole program if a compile job is submitted:

```
    /*ROUTE  PRINT REMOTE13
    $$ ABC:WORK.P(1)<20:*>
             .
             .
             .
    $$ ABC:WORK.P(N)<20:*>
```

Here we have assumed that the source segments P(n) are labelled 10,20,30,... . Then the /*ROUTE commands are omitted from the input.

4.4 Printing AMOS Job Output on a Remote Station

Due to present limitations of AMOS, it is not possible to print an AMOS file segment on an arbitrary line printer. Per default the PRINT command will print the segment on the printer associated with the terminal on which PRINT was issued (i.e. mostly the segment will be printed centrally).

If the user has completed development of his program and if he has decided that he would like to keep the last output list, he may issue a job

```
    /*ROUTE  PRINT REMOTEn
    // EXEC APRINT
    $$ ABC:WORK.LIST
```

which will print the contents of his segment WORK.LIST on remote printer n.

An even more convenient way of handling the output is by including the JCL statements in the output. The time stamp format statement (cf. section 4.1) may be modified to

```
    100 FORMAT ('/*ROUTE  PRINT REMOTEn'/
       *        '// EXEC APRINT'/
       *        3(2A4,4X)/)
```

producing an AMOS segment

```
    1 /*ROUTE  PRINT REMOTE8
    2 // EXEC APRINT
    3 24.08.78     15.11:37      ABC837
    4
    5 {        output from job        }
    .
    .
    .
```

which after inspection at the terminal may be printed at remote printer n simply be typing in the command XSUBMIT. If the segment is to be printed centrally, line 1 with the ROUTE command must be deleted.