DPST--89-234

DE89 010412

NRTSC
NUCLEAR REACTOR TECHNOLOGY
AND SCIENTIFIC COMPUTATIONS

KEYWORDS: Fortran
Computer
Source Code

RETENTION: Lifetime

The GRIPS Program: User Manual

July 26, 1988

By

Henry C. Honeck
Computer Application Technology, Inc.

Issued: February 1, 1989

SRL SAVANNAH RIVER LABORATORY, AIKEN, SC 29808
E. I. du Pont de Nemours & Company, Inc.
Prepared for the U. S. Department of Energy under Contract
DE-AC09-76SR00001

Project: Computer Utility Programs

Document: DPST-89-234

Title: The GRIPS Program: User Manual
July 26, 1988

Contract: AX-811431
GRIPS and FLASH Program Development

Approvals

J. E. Halverson, Research Supervisor — Date: 1-19-89

R. J. Pryor, Research Manager — Date: 1/19/89

B. W. Westmoreland, Technical Reviewer — Date: 1/19/89

Table of Contents

List of Figures

Introduction

We must come to grips with the problem of maintaining source code for a variety of computers. Ideally, we would like to maintain ONE set of source code on the VAX in a scattered form (one subroutine per file). We could then use the GRIPS program to produce other sets of source code for other computers such as the SCS-40, Crays, IBMs, and PCs. The name GRIPS is an acronym for

Gather and scatter,
Remove specified card images,
Include common files, change
Parameters, and
Select computer-dependent code.

The operation of GRIPS is illustrated in Figure 1. Let us start with computer B in the middle of Figure 1. On the left are the Fortran source files in a scattered form. There may also be non-Fortran text or data files. GRIPS can gather up these files into a single composite file for shipment to another computer. Shipment is very awkward when there are many files to ship. Being able to ship a single composite file is a great convenience. GRIPS is controlled by a set of directives in a directives file. These directives tell GRIPS the names of the source and text files, the name of the composite file, the templates of files to be included, and how to prepare the composite file for the target computer. The composite file can then be used in two different ways.

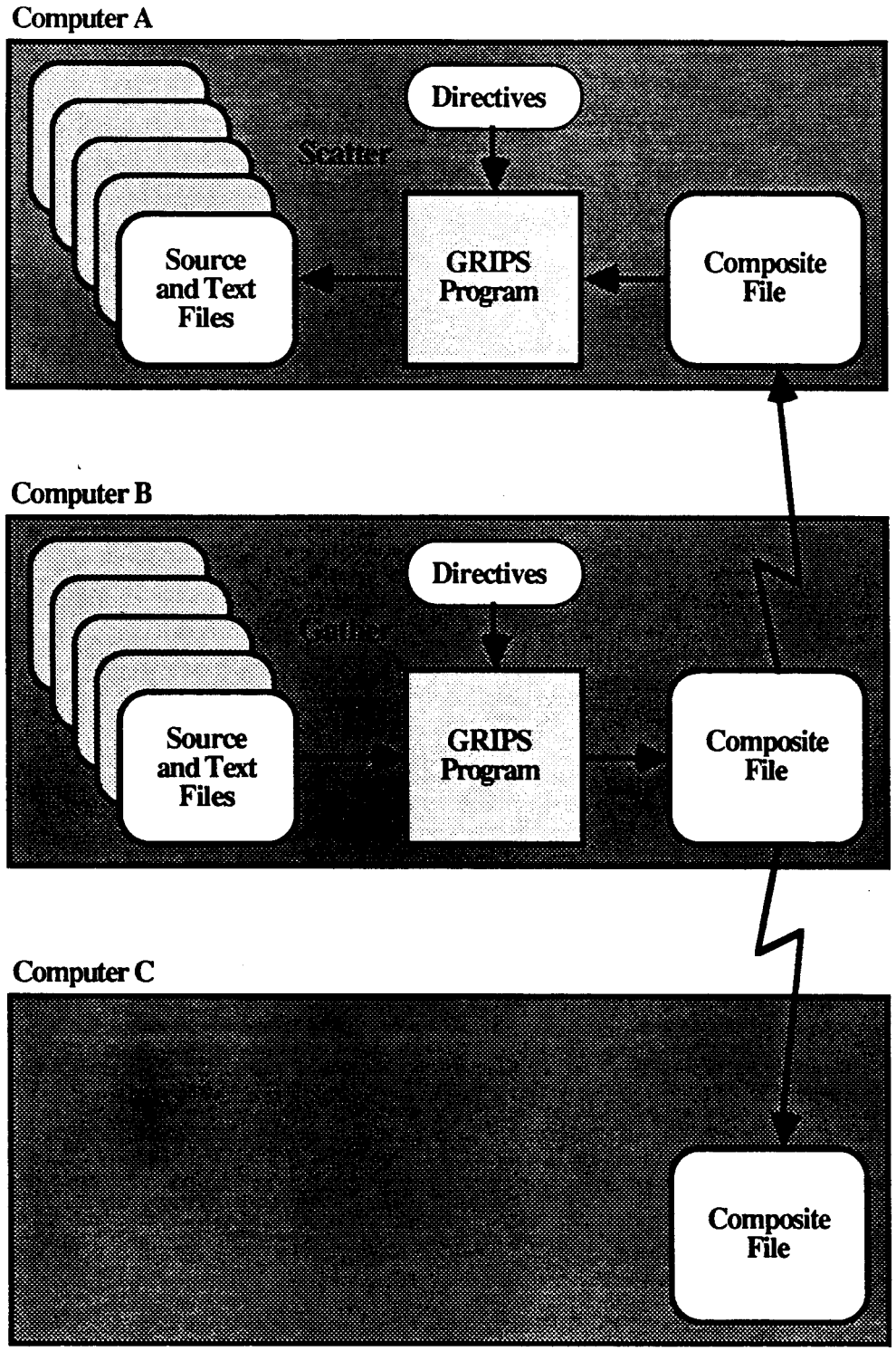
If further development of the source code is to be done, then the composite file is shipped to computer A and we again want the source code in a scattered form. GRIPS on computer A scatters files from the composite file to a series of source and text files on computer A. When the development is completed, the operation can be reversed. GRIPS gathers the source and text files on computer A into a composite file which is shipped to computer B where GRIPS scatters it into source and text files on computer B. Thus, the scatter and gather operations are reversible provided that certain directives which remove card images were not used.

An example of this type of operation was the Joshua Precompiler which was developed on an IBM PC/AT. There are over 120 subroutines, 12 include files, and 30 document files. The directives file on the PC/AT which gathers the source and text files is as follows:

```
GATHER
SELECT (SYSTEM='VAX/VMS')
SOURCE (C:\JPC\SRC\*.FOR,FORTRAN)
SOURCE (C:\JPC\SRC\*,INCLUDE)
TEXT (C:\JPC\DOC\*.DOC,DOCUMENT)
EXCLUDE (C:\JPC\SRC\TEST.FOR)
COMPOSITE (C:\JPC.CMP)
```

The syntax will be explained in detail later, but we can easily see that the operation is to gather the source files whose names match the template C:\JPC\SRC*.FOR which are of type FORTRAN, add the source files whose names match the template C:\JPC\SRC* which are of type INCLUDE, add the text files whose names match the template C:\JPC\DOC*.DOC which are of type DOCUMENT, exclude the single file named C:\JPC\SRC\TEST.FOR, and put all of them in the composite file named C:JPC.CMP.

Figure 1 GRIPS Programs and Files



The type names FORTRAN, INCLUDE, and DOCUMENT will be explained later. The composite file is shipped to the VAX-8550 where it is named [S8999]JPC.CMP. It is then scattered using the following directives:

```
SCATTER
SOURCE ([S8999.JPC.SRC]*.FOR,FORTRAN)
SOURCE ([S8999.JPC.INC]*.INC,INCLUDE)
TEXT ([S8999.JPC.DOC]*.DOC,DOCUMENT)
COMPOSITE ([S8999]JPC.CMP)
```

The procedure can be reversed (gather on the VAX, ship to the PC, scatter on the PC) by interchanging the SCATTER and GATHER directives.

If minimal further changes are to be made in the source, then the situation is like that shown at the bottom of Figure 1. The composite file on computer C is to be compiled, linked, and executed. Further changes in the source files will be made on computer B and the process repeated. Since shipment may be done often, the composite file should be as small as possible. This appears to be a very convenient way of working when computer C is the SCS-40 or a Cray. For example, the directives to gather the source for a code named DIF might be:

```
GATHER
UPDATE
NO COMMENTS
SELECT (SYSTEM='SCS/CTSS')
PRIORITY ([S8999.DIF.SRC]*.SCS]
SOURCE ([S8999.DIF.SRC]*.FOR)
INCLUDE ([S8999.DIF.INC]*.INC)
COMPOSITE ([S8999]DIF.SRC
```

Only those files which have been updated since the last GRIPS run are included. All comment lines are to be removed. Include statements are to be expanded using the template [S8999.DIF.INC]*.INC. If a source file with an extent SCS is found, it takes priority over the file with the extent FOR. Statements for the SCS-40 computer are to be selected, and the composite file [S8999]DIF.SRC is prepared for shipment to the SCS-40. The same procedure which executes GRIPS can also ship, synchronize, compile, link, and run the DIF program on the SCS-40, all via one user command from the VAX.

The remaining sections describe the execution of GRIPS and the format of the directives.

Executing GRIPS

The execution of GRIPS is controlled by a series of directives contained in a file named *DirectivesFileName* which will be referenced by GRIPS as standard input unit 5. There is a short printed output which is written to the file named *PrintFileName* which will be referenced by GRIPS as standard output unit 6. The connection between file name and unit number must be made outside of GRIPS. GRIPS is executed on the VAX using:

```
$ DEFINE FOR005 DirectivesFileName
$ DEFINE FOR006 PrintFileName
$ GRIPS
```

The Directives File

A directive has the form of a keyword followed (optionally) by one or two arguments enclosed in parentheses. The keyword will be converted to upper case, blanks ignored, and truncated to 4 characters. Thus all of the following represent the keyword SOURCE:

S o U r C e
SOURCES of INFORMATION
SOUR

Each directive is placed on a separate 72-character line and must be complete on the line. Directives may be entered in any order and will be processed in the order entered. Literal strings may be used in the arguments. Literal strings may be enclosed in either single or double quotes. A single quote is treated as an ordinary character in a literal string delimited by double quotes. A double quote is treated as an ordinary character in a literal string delimited by single quotes. For example:

'He called "Help"'	defines the string	He called "Help"
"I can't go"	defines the string	I can't go
'I can't go'	is not valid	
"abc"def"	is not valid	
'string"	is not valid	

Blanks not in literal strings are ignored. Lower case letters not in literal strings are converted to upper case letters. Non-ASCII characters are invalid.

Since file naming conventions vary from computer to computer, the following general definitions are used here:

"file name" means the complete name for a file on a specific computer.
VAX example: SRLUSER1:[S8999.SOURCE]MAIN.FOR

"file shorthand" means the short (1-8) character name which is computer independent.
VAX example: MAIN in the above file name.

"file template" means a file name with the file shorthand represented by a '*'
VAX example: SRLUSER1:[S8999.SOURCE]*.FOR

Note that SRLUSER1:[S8999.SOURCE]*.* is NOT a valid file template.

Gathering Source Files

There are several directives that are used to specify the gathering of several source or text files into a single composite output file. The formats for these directives are as follows:

GATHER
UPDATE
PRIORITY (*SourceFileSpecs* [, *FileTag*])
SOURCE (*SourceFileSpecs* [, *FileTag*])
TEXT (*TextFileSpecs* [, *FileTag*])
EXCLUDE (*ExcludeFileName*)
COMPOSITE (*CompositeFileName*)

The above directives may be entered in the directives file in any order. There must be one and only one GATHER and COMPOSITE directive. The UPDATE directive is optional but only one such directive should be used. There may be multiple PRIORITY, SOURCE, TEXT, and EXCLUDE directives.

The PRIORITY and SOURCE directives specify the names of the Fortran source files to be gathered, and the TEXT directive specifies the names of the non-Fortran files to be gathered. Multiple PRIORITY, SOURCE, and TEXT directives may be used. They will be processed in the sequence entered. The *SourceFileSpecs* or *TextFileSpecs* may be a file name which specifies a single file to be gathered, or they may be a file template which specifies a set of files to be gathered. The *FileTag* is an optional user-supplied word which describes the type of file. It is used only in conjunction with a subsequent GRIPS SCATTER run. The distinction between the PRIORITY and SOURCE directives is that if a file shorthand satisfies the *SourceFileSpecs* in both a PRIORITY and SOURCE directive, then the file specified by the PRIORITY directive has priority and is gathered, while the file specified by the SOURCE directive is excluded. No combination of SOURCE and PRIORITY directives will cause the same file to be gathered twice.

The EXCLUDE directive specifies the name of an input file to be excluded (not read). Multiple EXCLUDE directives may be used and may be entered in any order. Each dire ve applies to all of the PRIORITY, SOURCE, and TEXT directives regardless of ord n the directives file.

The UPDATE directive specifies that only updated files are to be read. A Reference time is defined as follows. If the COMPOSITE file (described below) exists, and if the UPDATE directive is given, then the Reference time is that of the COMPOSITE file. Otherwise, the Reference time is zero. PRIORITY, SOURCE, and TEXT files are processed only if they (or something included in them) were created or updated after the Reference time, or if they contain a Parameter that is to be changed. However, GRIPS does not know which files were processed in prior runs. If a previously unprocessed file is to be processed in the current run, then it must be given a time later than that of the COMPOSITE file.

The COMPOSITE directive specifies the name of the single output file. Only one COMPOSITE directive may be entered and the *CompositeFileName* must be a file name. All files specified by the PRIORITY, SOURCE. and TEXT directives and not excluded by the EXCLUDE directives are written in sequence to the composite file. If a *FileTag* was specified, then each file is headed by a card image with one of the two following forms:

* SOURCE (*SourceFileShorthand* , *FileTag*)
* TEXT (*TextFileShorthand* , *FileTag*)

where *SourceFileShorthand* or *TextFileShorthand* is the shorthand name of the source or text file. Note that if the NO COMMENTS directives (see below) is used, then these card images will be removed and the composite file cannot be scattered at a later time.

Scattering Source Files

There are several directives that are used to specify the scattering of source and text files contained in a single composite file. The formats for these directives are as follows:

SCATTER
COMPOSITE (*CompositeFileName*)
SOURCE (*SourceFileTemplate* , *FileTag*)
TEXT (*TextFileTemplate* , *FileTag*)

These directives may be entered in any order in the directives file. There must be one and only one SCATTER and COMPOSITE directive. There may be several SOURCE and TEXT directives.

The COMPOSITE directive specifies the name of the single file whose component files are to be scattered. The *CompositeFileName* must be a file name. The file must have been produced by a GRIPS GATHER run and have one of the two following card images at the head of each source or text file in the composite file.

* SOURCE (*SourceFileShorthand* , *FileTag*)
* TEXT (*TextFileShorthand* , *FileTag*)

where *SourceFileShorthand* and *TextFileShorthand* are the shorthand names of the source and text file, and *FileTag* is the user supplied tag for the file. Note that if the NO COMMENTS directive had been used when the composite file was gathered, then these card images would have been removed and the composite file cannot be scattered.

The SOURCE directive specifies the templates of the Fortran files, and the TEXT directive specifies the templates of the non-Fortran files to be scattered. Multiple SOURCE and TEXT directives may be used. Each file read from the composite file must have either a *SOURCE or *TEXT as the first card image. Assume that a *SOURCE card image is encountered. The *SourceFileShorthand* and the *FileTag* are obtained. The SOURCE directives are scanned in entry sequence for the one with a matching *FileTag*. An error stop occurs if no matching *FileTag* is found. The *SourceFileShorthand* is then inserted in the *SourceFileTemplate* from the SOURCE directive to obtain the *SourceFileName* where the source file will be written. Note that if there are several SOURCE directives with the same *FileTag*, only the first one will be found and used. The same procedure is used for the *TEXT files.

Changing Parameter Values

The value assigned to a parameter in a Fortran PARAMETER statement in the source can be changed to another value using a PARAMETER directive of the form:

PARAMETER (*ParameterName* = *ParameterValue*)

This directive may be used in either the GATHER or SCATTER mode. The delimiter between the arguments may be either an equal sign or a comma. The *ParameterName* is the name of the parameter as used in the PARAMETER statement in the source and should not be a literal string. The *ParameterValue* is the new value to be assigned to this parameter and may be a literal string. For example, if the source contains:

```
PARAMETER (NXX=200,IXX=150,ABC='123',XYZ=ABC)
```

then the directives

```
PARAMETER (NXX=500)
PARAMETER (ABC=" '45' ")
PARAMETER (XYZ='78')
```

would produce the new output statement

```
PARAMETER (NXX=500,IXX=150,ABC= '45' ,XYZ=78)
```

The PARAMETER directive is reversible in the sense that, even though the original value was removed, it can be restored using a PARAMETER directive with the original value. The directive can be used in either the GATHER or SCATTER mode.

The current implementation of the PARAMETER directive requires that the Fortran PARAMETER statement have no continuation statements. Further, the substitution of the new *ParameterValue* must not cause the line to overflow past column 72. The safe method is to place each parameter on a separate line.

Removing Selected Source Statements

There are two directives that can be used to remove source statements in both the GATHER and SCATTER mode:

NO COMMENTS
REMOVE (*RemoveString*)

The NOCOMMENTS directive removes all comment statements from the source plus any generated by other directives (e.g. INCLUDE, SELECT) in this GRIPS run. Comment statements are defined to be those with a 'c', 'C', '*', or '!' in column 1. The REMOVE directives matches the *RemoveString* with the statement starting in column 1. If a match is obtained, the statement is removed. For example, the following directive would remove all Cray CFT compiler directives:

```
REMOVE ('CDIR$')
```

The REMOVE directives leave the output in a non-reversible condition since the statements which were removed can not be put back into the source. Their main use is to produce a compact output which will require minimum transmission time to another computer where it will only be compiled and run with little or no editing.

Trimming the Output Card Images

In most instances the card images in the output file can be trimmed, that is, trailing blanks removed. Trimming Fortran source cards can reduce the size (and thus transmission time) of the output file by a factor of two or more. Trimming is the default. There is one known circumstance where trimming can cause a problem. This occurs because VAX Fortran does not restore the trimmed blanks and hence may lose some blanks in a literal continued on a second card image. The following directive can be used in both the GATHER and SCATTER mode to prevent trimming:

NO TRIMMING

Including Groups of Source Statements

We frequently include the same group of statements into several source routines. The definition of COMMON blocks which appear in several subroutines is a typical example. The statements to be included are placed in a separate named file. The INCLUDE statement is then used in the source routine at the place where the statements are to be included. The INCLUDE statement has one of the two forms:

INCLUDE *'IncludeFileShorthand'* or INCLUDE (*IncludeFileShorthand*)

where *IncludeFileShorthand* is the shorthand name for the file. The single quotes in the first form are delimiters and not the start and end of a literal string. *IncludeFileShorthand* from either form is converted to upper case and blanks are ignored.

INCLUDE statements in the source are intended to be expanded by either GRIPS or the compiler of the target computer. In the latter case no action is required of GRIPS and the INCLUDE statements are ignored. If GRIPS is to perform the include (usually in the GATHER mode), then there must be one or more INCLUDE directives of the form:

INCLUDE (*IncludeFileTemplate*)

where *IncludeFileTemplate* is the template for the files to be included. Assume that an INCLUDE statement has been encountered in the source. The *IncludeFileShorthand* is extracted from the INCLUDE statement. The directives are inspected in the order entered. When an INCLUDE directive is found, the *IncludeFileTemplate* is extracted and the *IncludeFileShorthand* replaces the '*' in the *IncludeFileTemplate* to form a file name. An attempt is made to OPEN the file so named. If the attempt is unsuccessful, then the next INCLUDE directive is tried. If they all fail, then the INCLUDE statement is left as is to be processed by the compiler of the target computer. INCLUDE statements may be nested, however, only the outermost one will be expanded by GRIPS.

If the OPEN is successful, then the file is read into the source following the INCLUDE statement and the INCLUDE statement is marked with a '* BGN' in columns 1-5. The file

is also saved in GRIPS main memory for possible repeated use. The following line is inserted in the source following the included statements to mark the end of the inclusion:

```
* END INCLUDE
```

Note that these statements will be removed if the NO COMMENTS directive is used. For example, assume the file to be include is named [S8999.DIF.INC]GENCOM.INC and contains the following statements:

```
INTEGER I1,I2,I3
REAL R4,R5
COMMON /GENCOM/ I1,I2,I3,R4,R5
```

The following INCLUDE directive would be used:

```
INCLUDE ( [S8999.DIF.INC]*.INC )
```

and the source containing the above INCLUDE statement would become:

```
* BGN  INCLUDE 'GENCOM'
        INTEGER I1,I2,I3
        REAL R4,R5
        COMMON /GENCOM/ I1,I2,I3,R4,R5
* END  INCLUDE
```

The include process is reversible. If the source contains included statements as above with the * INCLUDE and * END INCLUDE lines, then the UNINCLUDE directive can be used to reverse the process. The UNINCLUDE directive has the form:

```
UNINCLUDE
```

INCLUDE and UNINCLUDE directives may not be used in the same GRIPS run. An INCLUDE file may contain SELECT conditionals (described later).

Selecting Groups of Source Statements

We may have to use different groups of source statements for different computers. To do this we would include in the source the following types of statements (the '*' must appear in column 1):

```
* SELECT CASE ( SelectName )
* CASE ( SelectValue [ , SelectValue [ ... ] ] )
* CASE DEFAULT
* END SELECT
```

Except for the '*', these are like the new CASE statements proposed for Fortran 8x. The SELECT CASE statement specifies the *SelectName* of the variable used to make a selection. The CASE statement specifies one or more *SelectValues* of the *SelectName* for which the case is true. If the same *SelectValue* appears in a second CASE statement, it is ignored since only one CASE statement is allowed to be true. Cases not selected by the CASE statements are selected by the CASE DEFAULT statement which should appear after the CASE statements. Finally, the END SELECT statement signals that all cases have been specified. Consider the example where *SelectName* is SYSTEM and there are three *SelectValues* of 'VAX/VMS', 'IBM/MVS', and 'SCS/CTSS'. The following statements might appear in the source code:

```
* SELECT CASE ( SYSTEM )
* CASE ( 'VAX/VMS', 'IBM/MVS' )
    first statement used for the VAX or IBM computer
    second statement used for the VAX or IBM computer
    ...
* CASE ( 'SCS/CTSS' )
*   first statement used for the SCS computer
*   second statement used for the SCS computer
*    ...
* CASE DEFAULT
*   first statement used when the computer is not the VAX nor the IBM nor the SCS
*   second statement used when the computer is not the VAX nor the IBM nor the SCS
*    ...
* END SELECT
```

This source code is "selected" for the VAX or IBM computer since these statements do not have a '*' in column 1, and "deselected" for all other cases by the '*' in column 1. The selection can be changed in a GRIPS run by using a SELECT or EXTRACT directive of the form:

```
SELECT ( SelectName [ = SelectValue ] )
EXTRACT ( SelectName [ = SelectValue ] )
```

The EXTRACT directive will be described later. If the *SelectName* and *SelectValue* in source statements and in directives are not in literal strings, blanks will be ignored and lower case will be converted to upper case. If the *SelectValue* is omitted, the CASE DEFAULT (if any) is selected. If no directive with a matching *SelectName* is found, the source code is left as is. If several SELECT directives with the same *SelectName* are used, only the first will be recognized because only one CASE statement in a SELECT CASE is allowed to be true. If we used the following directive:

```
SELECT ( SYSTEM = 'SCS/CTSS' )
```

in a GRIPS run that read the above source code, then we would get the following output:

```
* SELECT CASE ( SYSTEM)
* CASE ( 'VAX/VMS', 'IBM/MVS' )
*   first statement used for the VAX or IBM computer
*   second statement used for the VAX or IBM computer
*   ...
* CASE ( 'SCS/CTSS' )
    first statement used for the SCS computer
    second statement used for the SCS computer
    ...
* CASE DEFAULT
*   first statement used when the computer is not the VAX nor the IBM nor the SCS
*   second statement used when the computer is not the VAX nor the IBM nor the SCS
*   ...
* END SELECT
```

Note that whatever was previously selected is now deselected, and the statements for the SCS computer are selected. We assume that both the input and output source code are ready for compilation on some computer. This implies that some selection has always been made. We could allow for source code to containing CASE statements with none selected and provide an UNSELECT directive to restore the input source to the unselected state. However, the ability to switch the source code from one case to another using the SELECT directive should be adequate.

Selection involves removing the '*' from statements while deselection involves placing the '*' on statements. If the statements contained a comment line marked with a '*', it would become "uncommented" during selection. Therefore, only a 'C' should be used to mark comment lines in source to be selected. Similarly, there can be problems with removing the '*' from *INCLUDE and *END INCLUDE lines. Therefore, if an INCLUDE statement is used in a SELECT construct, and if it is expanded using an INCLUDE directive, then the '*' lines should be removed with a NO COMMENTS or REMOVE('*') directive.

If the NO COMMENTS directive is not used, then the selection process is reversible and amounts to switching the '*'s between selections. If the NO COMMENTS directive is used, the '*' is treated as a comment and removed. The output contains only the selected statements and the selection cannot be reversed. Another option is to use the EXTRACT directive which has the same format as the SELECT directive. The difference is that the EXTRACT directive removes all deselected cards including those marked with 'C'.

SELECT statements can be nested in the source code to a depth of 9. Figure 2 illustrates nesting. The left-most column represents the source code with nothing selected. The right two columns show the directives and the output source code for two cases. In the middle column patches 11, 22, and 31 have been selected. In the right-most column nothing is selected for key3 since it is in the deselected case key2='val22'. The bottom left column shows what happens when we also use a REMOVE (*) directive. The cleaner situation in the bottom right column results when the EXTRACT directive is used instead of the SELECT directive.

Figure 2 Examples of Nested Select Statements

```
Source Code
C=====================
  patch 00
C=====================
* SELECT CASE (key1)
* patch 10
* CASE ('val11')
* patch 11
C---------------------
*  SELECT CASE (key2)
*  CASE ('val21')
*  patch 21
*  CASE ('val22')
*  patch 22
C - - - - - - - - - - -
*   SELECT CASE (key3)
*   CASE ('val31')
*   patch 31
*   CASE DEFAULT
*   patch 3D
*   END SELECT
C - - - - - - - - - - -
*  END SELECT
C---------------------
* CASE ('val12')
* patch 12
* CASE DEFAULT
* patch 1D
* END SELECT
C=====================
  end
```

```
Directives
SELECT (key1='val11')
SELECT (key2='val22')
SELECT (key3='val31')

Composite Code
C=====================
  patch 00
C=====================
* SELECT CASE (key1)
* patch 10
* CASE ('val11')
  patch 11
C---------------------
*  SELECT CASE (key2)
*  CASE ('val21')
*  patch 21
*  CASE ('val22')
   patch 22
C - - - - - - - - - - -
*   SELECT CASE (key3)
*   CASE ('val31')
    patch 31
*   CASE DEFAULT
*   patch 3D
*   END SELECT
C - - - - - - - - - - -
*  END SELECT
C---------------------
* CASE ('val12')
* patch 12
* CASE DEFAULT
* patch 1D
* END SELECT
C=====================
  end
```

```
Directives
SELECT (key1='val11')
SELECT (key2='val21')

Composite Code
C=====================
  patch 00
C=====================
* SELECT CASE (key1)
* patch 10
* CASE ('val11')
  patch 11
C---------------------
*  SELECT CASE (key2)
*  CASE ('val21')
   patch 21
*  CASE ('val22')
*  patch 22
C - - - - - - - - - - -
*   SELECT CASE (key3)
*   CASE ('val31'
*   patch 31
*   CASE DEFAULT
*   patch 3D
*   END SELECT
C - - - - - - - - - - -
*  END SELECT
C---------------------
* CASE ('val12')
* patch 12
* CASE DEFAULT
* patch 1D
* END SELECT
C=====================
  end
```

```
Directives
SELECT (key1='val11')
SELECT (key2='val21')
REMOVE ('*')

Composite Code
C=====================
  patch 00
C=====================
   patch 11
C---------------------
    patch 21
C - - - - - - - - - - -
C - - - - - - - - - - -
C---------------------
C=====================
  end
```

```
Directives
EXTRACT (key1='val11')
EXTRACT (key2='val21')

Composite Code
C=====================
  patch 00
C=====================
   patch 11
   patch 21
 C====================
  end
```

Printed Output

The output printed to unit 6 by GRIPS is illustrated in Figure 3. This was from the actual GRIPS run on the IBM PC/AT to prepare the file to be shipped to the VAX. The directives are shown at the top of the figure and the output at the bottom fo the figure.

The output is a record of the files read or skipped. The first SOURCE directive requests all of the include files of which there are 2 (GRPCOM and INCCOM). They are given the *FileTag* of INC for use in a SCATTER run on the VAX. The next SOURCE directive requests all the Fortran source files of which there are 20. However, DFNEXT and DFNTAD are skipped by priority (Skip/Prio) because VAX-specific versions will be read later when the PRIORITY directive is executed. Also, TEST is skipped by exclusion (Skip/Excl) because the file is named in an EXCLUDE directive. The remaining 17 files are read. In other situations files might be skipped being duplicates (Skip/Dupl), or might be skipped by update (Skip/Updt). The TEXT directive read 2 documentation files, and the PRIORITY directive read the 2 VAX-specific source files. If INCLUDE directives had been given, then the first time that a file was actually included, a message is written indicating that the file was Saved. This means that the included file was saved for future use in a main memory buffer.

The directives are listed at the bottom of the printed output in the Directive Use Statistics. The number after Used is the number of times the directive was used. If a directive was not used, the message '(not used)' is given. Here the SELECT (SYSTEM='VAX/VMS') was not used. This is not an error in this case but it might be. The user should always check these statistics to see that things are the way he thinks they should be.

Finally, there is a count of cards read and written. In this case there were 21 more cards written than read. These are the 20 *SOURCE cards and 1 *TEXT card written at the head of each subroutine.

Figure 3 Sample GRIPS Output

```
Directives
GATHER
SOURCE (\GRP\SRC\*,INC)
SOURCE (\GRP\SRC.*.FOR,FOR)
TEXT    (\GRP\DOC\*,DOC,DOC)
PRIORITY (\GRP\SRC\*.VAX,FOR)
EXCLUDE  (\GRP\SRC\TEST.FOR)
COMPOSITE (\GRP\DAT\GRIPS.SRC)
SELECT    (SYSTEM='VAX/VMS')


Output
=============> Begin GRIPS Execution
Processing ==> Directives Input File
=============> Gather Operation
Composite  ==> \GRP\DAT\GRIPS.SRC
Read       ==> \GRP\SRC\GRPCOM
Read       ==> \GRP\SRC\CRDGET.FOR
Read       ==> \GRP\SRC\CRDPUT.FOR
Read       ==> \GRP\SRC\CRDTYP.FOR
Skip/Prio  ==> \GRP\SRC\DFNEXT.FOR
Read       ==> \GRP\SRC\DFNGAB.FOR
Skip/Prio  ==> \GRP\SRC\DFNTAD.FOR
Read       ==> \GRP\SRC\DIRGET.FOR
Read       ==> \GRP\SRC\ERROR.FOR
Read       ==> \GRP\SRC\GATHER.FOR
Read       ==> \GRP\SRC\GRIPS.FOR
Read       ==> \GRP\SRC\INCLUD.FOR
Read       ==> \GRP\SRC\PARAM.FOR
Read       ==> \GRP\SRC\RWFILE.FOR
Read       ==> \GRP\SRC\SCATTR.FOR
Read       ==> \GRP\SRC\SELECT.FOR
Read       ==> \GRP\SRC\STRCHR.FOR
Read       ==> \GRP\SRC\STRFAR.FOR
Read       ==> \GRP\SRC\STRNBL.FOR
Read       ==> \GRP\SRC\STRPRS.FOR
Skip/Excl  ==> \GRP\SRC\TEST.FOR
Read       ==> \GRP\DOC\GRPCOM.DOC
Read       ==> \GRP\SRC\DFNEXT.VAX
Read       ==> \GRP\SRC\DFNTAD.VAX
=============> Directive Use Statistics
Used      1 ==> GATHER
Used      1 ==> SOURCE (\GRP\SRC\*,INC)
Used     17 ==> SOURCE (\GRP\SRC.*.FOR,FOR)
Used      1 ==> TEXT    (\GRP\DOC\*,DOC,DOC)
Used      2 ==> PRIORITY (\GRP\SRC\*.VAX,FOR)
Used      1 ==> EXCLUDE  (\GRP\SRC\TEST.FOR)
Used      1 ==> COMPOSITE (\GRP\DAT\GRIPS.SRC)
(not used) ==> SELECT    (SYSTEM='VAX/VMS')
=============> Input/Output Stastics
In     2551 ==> Number of Cards Read
Out    2574 ==> Number of Cards Written
=============> End Grips Execution
```

Summary of GRIPS Directives

All GRIPS directives have the format:

Keyword (*FirstArgument* , *SecondArgument*)

The Keywords and arguments are summarized in the following table.

Keyword	*FirstArgument*	*SecondArgument*	GS	Page
COMPOSITE	*CompositeFileName*		GS	5,6
EXCLUDE	*ExcludeFileName*		G	5
EXTRACT	*SelectName*	[*SelectValue*]	GS	10
GATHER			G	5
INCLUDE	*IncludeFileTemplate*		GS	8
NOCOMMENTS			GS	7
NOTRIMMING			GS	8
PARAMETER	*ParameterName*	*ParameterValue*	GS	7
PRIORITY	*SourceFileSpecs*	[*FileTag*]	G	5
REMOVE	*RemoveString*		GS	7
SCATTER			S	6
SELECT	*SelectName*	[*SelectValue*]	GS	10
SOURCE	*SourceFileSpecs*	[*FileTag*]	GS	5,6
TEXT	*TextFileSpecs*	[*FileTag*]	GS	5,6
UNINCLUDE			GS	9
UPDATE			G	5

Notes:

1) Keyword may be in any case and may be truncated to 4 characters.
2) *FileSpecs* means either *FileName* or *FileTemplate*.
3) Second argument is optional if enclosed in [...].
4) The GS column has a G and/or S if used with GATHER and/or SCATTER.
5) The Page column is the page number in the text where the directive is discussed.

Appendix A. Managing Computer-Dependent Fortran Source Code

Fortran77 has done much to eliminate computer dependencies in Fortran source code. However, there are still differences in computer filing systems, subroutine libraries, and execution environments with which we must contend. The GRIPS code was developed to help programmers contend with these computer dependencies.

Four different methods for handling computer dependencies have been implemented in GRIPS. These methods are described in the following sections. Other methods can be developed and incorporated in GRIPS. The author welcomes comments and suggestions.

A.1 Method 1, Multiple Subroutines

Assume that a program has been developed on several computers and that a single version is now to be maintained on one computer, usually the VAX 8550. From this single version we must be able to generate versions of the program suitable for each target computer. The easiest way to do this is to develop separate computer-dependent versions of each subroutine. These versions must have unique file names. If the extent field of the file name is used to designate the computer, then GRIPS provides an easy way to gather up the proper versions for a target computer. Assume that a 3-character extent is used to designate the target computer. These could be VAX, SCS, IBM, CRA, etc. Computer independent subroutines will have the extent FOR. Then, to gather source code for the SCS computer, the following GRIPS directives would be used.

```
PRIORITY ([path]*.SCS)
SOURCE   ([path]*.FOR)
```

where [path] is the full directory path leading to the source. GRIPS will gather up all files which have the extent SCS and all files which have the extent FOR and which did not have the same name as a file with extent SCS. If [path]XYZ.FOR is a source file name, then it will be gathered only if [path]XYZ.SCS does not exist. If it exists, [path]XYZ.SCS will be gathered.

A.2 Method 2, Single Subroutine

If we have (as a first step) developed the computer-dependent subroutines as separate files as indicated above, it might be advantageous (as a second step) to combine the separate files into a single file with extent FOR and then let GRIPS "select" the proper statements for the target computer. GRIPS supports this with the SELECT CASE statements. For example, assume that our single file is named [path]XYZ.FOR and has the following statements and sections.

```
* SELECT CASE (SYSTEM)
* CASE ('CRAY/CTSS', 'CRAY/CTSS-1', 'CRAY/CTSS-XMP')
*       (All of the statements previously in the file named [path]XYZ.CRA)
* CASE ('IBM/MVS', 'IBM/MVS-3081', 'IBM/MVS-3083')
*       (All of the statements previously in the file named [path]XYZ.IBM)
* CASE ('SCS/CTSS')
*       (All of the statements previously in the file named [path]XYZ.SCS)
* CASE DEFAULT
        (All of the statements previously in the file named [path]XYZ.FOR)
* END SELECT
```

This is easy to accomplish with a text editor. Note that all sections except for the one following the * CASE DEFAULT have been "deactivated" using a '*' in column 1. Compiling this file we would give the same result as obtained by compiling the previous [path]XYZ.FOR file. However, if we run GRIPS and include the directive

```
SELECT (SYSTEM='SCS/CTSS')
```

then GRIPS will "activate" the statements previously in [path]XYZ.SCS and "deactivate" all the other sections. If we want to eliminate the "deactivated" sections, we would use the following GRIPS directive instead of the SELECT directive above.

```
EXTRACT (SYSTEM='SCS/CTSS')
```

Method 2 has two advantages over Method 1. First, there is only one source file which contains all versions. Second, the SELECT CASE syntax is far more flexible than the file name extent syntax. We have used the case name SYSTEM above with a set of case values like 'SCS/CTSS'. The last section of this memorandum discusses the choice for case name and values and lists some suggested case values.

A.3 Method 3, Selected Statements

There will be considerable duplication of statements in both Methods 1 and 2. If we start from the single file of Method 2, then we could eliminate the duplicate statements (those that apply to all computers) and use the SELECT CASE syntax only for those sets of statements that are computer-dependent. This is simply an editing job which results in a more compact (and perhaps more readable) source file.

A.4 Method 4, Parameters

Many computer dependencies can be handled by defining the dependent item as a Fortran parameter. Three examples are given here: array dimensions, file names and lengths, and file OPEN parameters.

Frequently it is necessary to change the storage layout or maximum problem size via array dimensions defined as parameters. For example, the following statements might appear in the source code.

```
PARAMETER (ND1X=200,ND2X=100)
DIMENSION A(ND1X),B(ND2X),C(ND1X,ND2X)
DO 10 ND1=1,ND1X
DO 20 ND2=1,ND2X
etc.
```

GRIPS can change these parameters by using the GRIPS directives

```
PARAMETER (ND1X=50)
PARAMETER (ND2X=75)
```

which will change all occurrences of these parameters.

File name lengths are arbitrary on the VAX but must be 8 or less on the SCS and CRAY. The following statements might appear in the source code and define both the file name length (FNL) and file name (IFN) as parameters.

```
INTEGER FNL
PARAMETER (FNL=64,IFN='[S8999.DIF.DAT]CASE1.DAT')
CHARACTER*(FNL) FILE1
FILE1=INFILE
OPEN (2,FILE=FILE1)
```

These statements would be suitable for the VAX computer. For the SCS computer we might change these parameters using the GRIPS directives

```
PARAMETER (FNL=8)
PARAMETER (IFN="'CASE1'")
```

Finally, we might have the value for the OPEN keyword STATUS defined as the parameter STAT. For example,

```
PARAMETER (STAT='OLD')
OPEN (2,FILE=F1,STATUS=STAT)
```

On some computers it is better to change STAT using the GRIPS directive

```
PARAMETER (STAT="'UNKNOWN'")
```

Note that using the GRIPS PARAMETER directive removes some of the computer dependencies from the source code and places them in the GRIPS directives. There must then be a GRIPS directives file for each target computer.

A.5 A Suggestion for SYSTEM Names

The author and Jane McCort suggest that the case name in the SELECT CASE syntax be the name SYSTEM (rather than the obvious COMPUTER) since it is to describe a combination of computer system and operating system. A partial list of the suggested values is as follows:

```
SYSTEM = PC/DOS
         PC/DOS-RM/F        (Ryan McFarland Fortran)
         PC/DOS-MS/F        (Microsoft Fortran)

         VAX/VMS

         SCS/CTSS

         CRAY/CTSS
         CRAY/CTSS-1
         CRAY/CTSS-XMP

         IBM/VM
         IBM/TSO
         IBM/MVS
         IBM/MVS-3081
         IBM/MVS-3083
```