SLAC COMPUTATION GROUP
Stanford, California

CGTM No. 198
October 1979

GOODGNUS

Using Object Decks for Monitoring a Program's Use of Variables

Roger B. Chaffee
Computation Research Group
Stanford Linear Accelerator Center

Program GOODGNUS reads the object decks produced by Fortran X, which is John Steffani's version of the Fortran H compiler. It writes information about the variables which are used in the compiled routines. Some of the information is the same as is given in the 'MAP' in the compiler output, but it is arranged and selected by GOODGNUS, and more is given.

With this output, it is relatively easy to determine variables which are used only in assignments ("write-only" variables), or which are used only in fetches ("read-only" or uninitialized variables). By combining and sorting the output for all the routines in a program, it is possible to see where each variable in common is set and used. Various program errors, such as uninitialized variables, common blocks of different sizes, inconsistent common block declarations, and unused variables, can all be found from this information.

For large programs, for which subroutines are compiled separately and kept in a library of load modules, the GOODGNUS output of each compilation could be kept in a Wylbur file, and the appropriate cards replaced each time a routine is recompiled. The GNUDEX system used by Group B at SLAC monitors the group's production code in just that manner. GOODGNUS was written as a part of the GNUDEX system.

Information is not put out for all variables. To decrease the amount of output, if a variable is not in common, is both stored and fetched, and is not initialized in a data statement, then its use is considered so unremarkable that it is not listed. Similarly, if a variable appears in a common statement but is not referenced in that routine, it is not listed.

(One problem is that references to a variable, in evaluating the logical part of a logical IF statement, are not counted by the compiler as "fetching" the variable. Variables which are used only in evaluating these logical expressions are flagged by GOODGNUS as not being fetched at all. Another problem is that, except for logical variables, a variable which is declared in a type statement (REAL, COMPLEX, etc.) but is otherwise not used, is not allocated space in the program, and is not seen by GOODGNUS at all. For this reason, variables which are flagged as 'NR' ('not referenced') in the compiler map do not appear in the GOODGNUS output, even though their existence probably signifies an error.)

A Sample Job

Input

```
      COMMON B1,B2,B3
      COMMON /FGHIJ/ F,G,H,I,J
      DIMENSION H(10)
      DO 20 K=1,MAX
 20   CALL SUB1(H,I,J)
      STOP
      END

      BLOCK DATA
      COMMON /ABCDE/ A,B,C,D,E
      DATA A,C,E /1.,2.,3./
      END

      SUBROUTINE SUB1(Q,MMMM,N352)
      DIMENSION Q(5)
      DATA IDATA /35/
      IF (IDATA.EQ.MMMM) Q(3)=UNINIT
      RETURN
      END
```

Output

```
MAIN                MAX       F  000088  000004            I*4
MAIN      BLANKCOM  $UNRF$                         00000C  COMN
MAIN      FGHIJ     H         SFA 000008  000028  000038  R*4
MAIN      FGHIJ     I         SFA 000030  000004  000038  I*4
MAIN      FGHIJ     J         SFA 000034  000004  000038  I*4
MAIN                SUB1        X                         CALL
MAIN                IBCOM#      X                         CALL
MAIN                            000105  000009  00010E
MAIN      $TOTAL                        000030  000044
MAIN      $DATE$                0       79.267/17.34.13
$BKDT$    ABCDE     A         S   000000  000004  000014  R*4          D
$BKDT$    ABCDE     C         S   000008  000004  000014  R*4          D
$BKDT$    ABCDE     E         S   000010  000004  000014  R*4          D
SUB1                MMMM          00007C  000004          I*4      V
SUB1                N352          000080  000004          I*4      V
SUB1                SUB1          000084  000004          R*4
SUB1                IDATA         000088  000004          I*4          D
SUB1                UNINIT    F   00008C  000004          R*4
SUB1                          0000FB  000015  000110
SUB1      $TOTAL                        000000  000000
SUB1      $DATE$                0       79.267/17.34.21
```

Output Format

For each 'interesting' variable, one line is put out. The output dataset consists of card images, 80 columns long.

Column	Format	Contents
1-10	Blanks	Member name
11-18	A8	Subroutine name
19-26	A8	Common name
28-34	A7	Variable name
35	A1	'Store' flag
36	A1	'Fetch' flag
37	A1	'External' or Formal Reference flag
39-44	Hex	Offset
47-52	Hex	Length (bytes)
55-60	Hex	Common length
62-65	A4	Type
66-73	A	BEQTVNID Flags

Columns which are not specifically mentioned contain blanks.

Flags

B	Variable appears in a structure statement.
E	Variable is equivalenced. (Q is usually set too.)
Q	Variable has appeared in an equivalenced group.
T	Variable appears in a 'Type' statement. (REAL,...)
V	Variable is a call-by-value parameter.
N	Variable is a call-by-name parameter.
I	Variable is used as a subscript.
D	Variable appears in a data statement.

Each flag appears in a particular column, for instance, if the D flag is present, it appears in column 73, regardless of the presence or absence of the other flags.

Comments

The Member Name is left blank by GOODGNUS. It can be filled in by the user if the subroutine in question is included in a particular member in a loadmodule library.

The Subroutine Name is the name of the csect that the variable appears in, or the csect containing the reference to the common block that the variable appears in. Block data routines are identified with the subroutine name $BKDT$.

The Common Name is the name of the common block that the variable appears in, or blank if it is local to the subroutine.

The Variable Name is just what the name implies. Equivalenced variables are often listed by only one name.

The Store Flag is the letter S, or blank, signifying that the variable is, or is not, used in the left-hand side of an equals sign.

The Fetch Flag is the letter F, or blank, signifying that the variable is, or is not, used in evaluating the expression on the right-hand side of an equals sign. (N.B. Using the variable in the logical expression in a logical IF statement will not set this flag.)

The External Flag or Formal Reference Flag can be the letter X, or R, or blank. 'X' signifies that the 'variable' is an external reference, used in either a subroutine call or external function reference. 'R' signifies that the variable appears in the subroutine argument list, and a local copy is made. (Arrays appearing in the argument list do not appear in the GOODGNUS output, since there is no local storage assigned to them.)

The Offset is the location, relative to the start of the subroutine or common block, of the variable or the first element of the variable.

The Length is the number of bytes assigned to the variable or array. (Remember that REAL and INTEGER variables take 4 bytes, and others may take 1, 2, 4, 8, 16, or 32.)

The Common Length is the total length of the common block in which the variable appears, or blank if it is a local variable.

The Type can be 'R*4 ','L*1 ', 'C*1', etc., in the case of a variable, or it can be 'CALL' or 'COMN'.

Summary Lines

The information for each routine is followed by three summary lines, each of which has blanks instead of the variable name.

Local Memory Usage

Column	Format	Contents
11-18	A8	Subroutine name
19-26	A8	Blanks for the "Common Name"
39-44	hex	Bytes used for code.
47-52	hex	Bytes used for local variables.
55-60	hex	Total csect length.

Common Block Usage

Column	Format	Contents
11-18	A8	Subroutine name
19-24	A6	'$TOTAL'
47-52	hex	Common space referred to.
55-60	hex	Common space defined.

Compilation Date & Time

Column	Format	Contents
11-18	A8	Subroutine name
19-26	A6	'$DATE$'
39	I1	Optimization level.
47-61	A	yy.ddd/hh.mm.ss

Unreferenced Common Blocks

If a common block is declared in a routine, but none of the variables in the block are used in the routine, then a line is written using '$UNRF$' in place of the usual variable name.

Column	Format	Contents
11-18	A8	Subroutine name
19-26	A6	Common name
28-33	A6	'$UNRF$'
55-60	Hex	Common length
62-65	A	'COMN'

JCL

Compile

GOODGNUS requires the DECK output from the October '79 (or later) version of the FORTX compiler. Until this version is installed as the default, you must add the parameter FORTSL1='WYL.CG.JPS.LMEDS' to your exec statement.

The following is a possible way to compile a routine and get a GOODGNUS output for it.

```
// JOB
// EXEC FORTXC,FORTPRM='DECK',FORTSL1='WYL.CG.JPS.LMEDS'
//FORT.SYSPUNCH DD DSN=&&OBJECT,DISP=(NEW,PASS),
//    SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA
//FORT.SYSIN DD *
     Fortran card input here.
//GNU       EXEC  PGM=GOODGNUS,REGION=30K
//STEPLIB   DD    DSN=WYL.CG.RBC.LOADMODS,DISP=SHR
//OBJECT    DD    DSN=&&OBJECT,DISP=OLD
//GNUCARDS  DD    SYSOUT=A
```

Compile and Execute

For compile-and-execute, the idea is the same. This example directs the GNU output to a dataset on a temporary volume.

```
// JOB
// EXEC FORTXCLG,FORTPRM='DECK',
//    FORTSL1='WYL.CG.JPS.LMEDS'
//FORT.SYSPUNCH DD DSN=&&OBJECT,DISP=(NEW,PASS),
//    SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA
//FORT.SYSIN DD *
     Fortran card input here.
//LKED.SYSIN DD *
      Linkage-editor input, if any, goes here.
//GO.SYSIN DD *
      Data for the program, if any, goes here.
//GNU       EXEC  PGM=GOODGNUS,REGION=30K
//STEPLIB   DD    DSN=WYL.CG.RBC.LOADMODS,DISP=SHR
//OBJECT    DD    DSN=&&OBJECT,DISP=OLD
//GNUCARDS  DD    DSN=WYL.gg.uuu.GNUOUT,DISP=(NEW,CATLG),
//     UNIT=SYSDA,SPACE=(TRK,(4,2),RLSE)
```

Compile and Add to a Loadmodule Library

There is no FORTXCL catalogued procedure, so if you want to compile a routine and add it to a loadmodule library, you will have to use FORTXCLG and dummy out the GO step.

```
// JOB
// EXEC FORTXCLG,FORTPRM='OPT=3,DECK',
//     FORTSL1='WYL.CG.JPS.LMEDS',
//     LKEDPRM='NCAL,LIST',
//     GOCND='(32,GT,FORT)'
//FORT.SYSPUNCH DD DSN=&&OBJECT,DISP=(NEW,PASS),
//     SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA
//FORT.SYSIN DD *
       SUBROUTINE SUBNAM(A,B,C,D,E)
       A=B
       C=D
       E=0.
       RETURN
       END
//LKED.SYSLMOD DD DSN=WYL.gg.uuu.LOADMODS,DISP=SHR
//LKED.SYSIN   DD *
   NAME  SUBNAM(R)
//GNU       EXEC  PGM=GOODGNUS,REGION=30K
//STEPLIB   DD    DSN=WYL.CG.RBC.LOADMODS,DISP=SHR
//OBJECT    DD    DSN=&&OBJECT,DISP=OLD
//GNUCARDS  DD    SYSOUT=A
```

The DCB for the GNUCARDS output is set by the program to RECFM=FB,LRECL=80,BLKSIZE=1600. In any of the above, you may also specify PARM=DUMP in the EXEC PGM=GOODGNUS statement. This will cause a listing of the object deck input to appear in the GNUCARDS output.