Release Notice
CONVEX FORTRAN Compiler V7.0.1

Document No. 720-001830-014

January 1992

Richardson, Texas USA
CONVEX Computer Corporation

Release Notice
CONVEX FORTRAN Compiler V7.0.1

CONVEX COMPUTER CORPORATION

1

Introduction

This document describes Version 7.0.1 of the CONVEX FORTRAN compiler. This document is intended to enhance and clarify the existing permanent documentation for this product with information that is up-to-the-minute or was developed too late for inclusion in the permanent documentation.

The remaining sections in this document describe the contents of this release.

- Section 2 describes the contents of this distribution.
- Section 3 contains notes and cautions about the use of this software.
- Section 4 describes how to use CONVEX FORTRAN after installing this new release.
- Section 5 describes the new features provided in this release.
- Section 6 lists previously reported problems which have been fixed.
- Appendix A contains source code examples for known software problems that can cause an application to generate incorrect answers.
- Appendix B contains source code examples for other software problems, and a description of known documentation problems.

The CONVEX FORTRAN Compiler is a scalar optimizing compiler (automatically performing local and global scalar optimizations), a vectorizing compiler (automatically performing vectorization of loops), and a parallelizing compiler (automatically performing parallelization of loops). This release consists of the compiler driver (*fc*), the compiler (*fskel*), the error message file (*errmsg.fc*), the runtime system libraries, the *man* pages, the new documentation, and other FORTRAN-specific utilities.

2

Contents of This Distribution

The distribution package for this release of the CONVEX FORTRAN Compiler consists of this document, distribution media for the software, and a complete documentation set. The specific contents of the distribution are described in the following tables:

FORTRAN Compiler Media

Item	Qty	Type	Part Number	Description	Format
1	1	Mag.	720-000515-226	CONVEX FORTRAN Compiler V7.0.1	installsw

FORTRAN Compiler Documentation

Item	Qty	Type	Part Number	Description
1	1	Manl.	720-000130-101	*CONVEX FORTRAN Guide* First Edition (includes the *Language Reference Manual*, the *User's Guide*, and the *man* pages)
2	1	Manl.	720-002430-002	*CONVEX FORTRAN Quick Reference*, Second Edition
3	1	Manl.	720-000930-202	*CONVEX FORTRAN Optimization Guide*, Third Edition
4	1	Manl.	720-001930-014	*CONVEX FORTRAN Installation Instructions*
5	1	Manl.	720-001830-014	*CONVEX FORTRAN Release Notice*

All of these documents have been updated for this release.

3

Notes and Cautions

This section contains general information and words of caution about the product.

3.1 Serial Numbers

The compiler checks the serial number of the machine on which it is running. If the serial number does not match the expected one, a message is printed and execution is terminated.

3.2 Prerequisites

The following versions of the listed products are recommended:

Product	Version	Status
ConvexOS	9.1 or newer	required
CONVEX C	4.1 or newer	required
Consultant	8.2	optional
CXpa	1.2	optional
CXdb	1.1	optional

The required versions of ConvexOS and CONVEX C should be installed before installing CONVEX FORTRAN V7.0.1.

Versions of CONVEX Consultant, CXdb and CXpa older than those shown in the above table are not compatible with CONVEX FORTRAN V7.0.1.

The following board revision levels are required for CONVEX FORTRAN V7.0.1:

Model	Board Revision levels
C1 XL	
C1	
C210	
C220 C230 C240	VPC rev. G

A hardware upgrade is available for machines that did not include these features as original equipment.

The following revisions of the Diagnostic Database are recommended for CONVEX FORTRAN V7.0.1:

Model	Diagnostic Database Revision Levels
C1	V1.15 or higher
C210, C220, C230, C240	V3.5 or higher

These versions of the diagnostic databases ensure that the ATAN instruction will produce the same answers as the ATAN library routine (more accurate than older versions).

4

Using This Compiler

When installed, this compiler replaces the existing CONVEX FORTRAN compiler in */usr/convex*, and may be invoked with the *fc* command if */usr/convex* is in your path.

The previously installed compiler will be moved to the */usr/convex/oldfc* directory. It may be invoked as follows:

```
/usr/convex/oldfc/fc -B/usr/convex/oldfc +dlib=/usr/convex/oldfc ...
```

If the old compiler was installed in an alternate directory, the symbolic links in */usr/convex* and */usr/lib* will be moved into */usr/convex/oldfc*. The files in the alternate directory will not be affected.

5

New Features

5.1 Faster I/O

Certain I/O statements with implied DO loops will execute much faster.

5.2 Cray Binary I/O Enhancements

A conversion program for Cray "blocked" sequential access unformatted files is included in this release. The conversion program will convert Cray "blocked" sequential access unformatted files to a form CONVEX FORTRAN can read. CONVEX FORTRAN requires a data format specifier of "cray" (usually in the OPEN statement) to read a converted file.

Usage:

```
fcUnblock <inputfile >outputfile
```

In addition, support for Cray "unblocked" sequential access unformatted files is now available. A program reading one of these files should specify a dataformat of "crayub" in the OPEN statement (instead of "cray"). Note that these files, as on a Cray, do not conform to the FORTRAN 77 standard. There are no records, and BACKSPACE is not fully supported.

Note that Cray "unblocked" direct access unformatted files are still supported (READs and WRITEs) when a data format of "cray" is specified.

5.3 Short Circuit Evaluation of IF conditional Expressions

This release of CONVEX FORTRAN automatically short circuits certain expressions in IF statements. When a conditional expression is short circuited, it is possible that part of that expression will not be evaluated when the value of the whole expression will not be affected by the value of a particular part of the expression. For example, if A is a logical variables, and A is TRUE, then executing the statement

```
IF (A .OR. C.LT.D) THEN
```

will NOT cause C.LT.D to be evaluated, since the value of the whole expression is guaranteed to be TRUE if A is TRUE.

When the conditional expression in the IF statement contains .AND. and .OR. operators, these will usually be short circuited. Note that some expected function call side effects may not occur in the presence of short circuiting. The *-nosc* flag disables this optimization.

5.4 New Cross Reference

An all new cross reference, with a different output format, is now the default (when the *-xr* flag is specified). The old cross reference package is still available, but is no longer supported, and does not accept all language features. Use the *-xro* flag if you wish to use the old xref.

The new cross reference allows multiple source files to be cross referenced together. The *-xra* flag is used to generate cross reference data files but not a report. The *fcxref* program can then be invoked to generate a report after several source files have been processed. See Chapter 4 of the *CONVEX FORTRAN User's Guide* for more information.

5.5 Loop Peeling

At optimization level *-O2* and above, loops are automatically peeled. The following description of loop peeling is simplified in order to convey the general algorithm. A more complete description appears in the *CONVEX FORTRAN Optimization Guide* that is included with *CONVEX FORTRAN V7.0.1*.

Loop peeling removes IF statements which test for the first or last iteration of the enclosing loop. The code which is executed during the first and/or last iteration is moved outside the loop.

A copy of the loop body is generated for the initial iteration (with or without the IF statement body as appropriate), then a loop for the 2nd thru last-1 iterations is generated, and then a copy of the loop body is generated for the last iteration. Note that the initial and final copies of the loop bodies are only created as appropriate based on the IF statements within the original loop.

Example:

```
! original source code
  DO I=1,10
      A(I) = I
      IF (I.EQ.1) THEN
          A(I) = 99
      ENDIF
  END DO
```

```
! equivalent peeled code
      A(1) = 99
      DO I=2,10
          A(I) = I
      END DO
      END
```

Loop peeling allows the compiler to vectorize/parallelize the resulting loops more effectively.

By default (when *-O2* or *-O3*) is specified, a modest amount of loop peeling is attempted. When successful, the resulting program will usually be bigger.

Loop peeling may be disabled with the *-nopeel* flag. Somewhat more aggressive peeling is enabled with the *-peel* flag. Very aggressive loop peeling may be enabled with the *-peelall* flag. Programs with large complicated loop nests which are peeled may sometimes cause the compiler to exceed certain internal limits. In these cases, the *-peelall* and *-peel* flags should not be used.

5.6 Redundant Test Elimination

The compiler now automatically eliminates conditional expressions which are redundant. The body of code conditionally executed is either retained or eliminated as appropriate.

Example:

```
! original source code
  do j=1,n
      if (j.gt.0) then
          print *,"hi"
      end if
  end do
```

```
! equivalent code
  do j=1,n
      print *,"hi"
  end do
```

5.7 Test Promotion

At optimization level *-O2* and above, the compiler will automatically promote some loop constant IF statements within a loop out of the loop.

Example:

```
! original source code
  do i=1,10
      if (zz) then
          a(i) = i
      end if
  end do
```

```
! equivalent test promoted code
      if (zz) then
          do i=1,10
              a(i) = i
          end do
      end if
```

Test promotion allows the compiler to vectorize/parallelize the resulting loops more effectively.

By default (when *-O2* or *-O3*) is specified, a modest amount of test promotion is attempted.

Test promotion may be disabled with the *-noptst* flag. Somewhat more aggressive test promotion is enabled with the *-ptst* flag. Very aggressive test promotion may be enabled with the *-ptstall* flag. Programs with large complicated loop nests in which test promotion occurs may sometimes cause the compiler to exceed certain internal limits. In these cases, the *-ptstall* and *-ptst* flags should not be used.

Note that test promotion and loop peeling together may substantially increase the size of programs with many large complicated loop nests. These optimizations can interact with each other to exponentially increase the size of a program (and the time to compile it) under certain circumstances. Indiscriminate use of the *-peelall* and *-ptstall* flags can result in excessive compilation times as well as causing the compiler to exceed certain internal limits. The default level of loop peeling and test promotion was chosen to minimize the added overhead in compilation time and program size while still providing significant benefit from these optimizations.

5.8 Fortran 90 support

This release of CONVEX FORTRAN supports the *-f90* flag which enables support for the following Fortran 90 features:

- Automatic Arrays

  Automatic arrays are implicitly defined by the occurrence of a local array (not in COMMON and not

a dummy argument) which is dimensioned with a non-constant expression. These arrays are allocated on the stack at runtime when the procedure is called.

Automatic arrays are allowed with the *-cfc* and *-f90* flags. Note that this is a change in *-cfc* functionality.

- Allocatable Arrays

Allocatable arrays are declared with the ALLOCATABLE statement. Storage for these arrays must be declared with the ALLOCATE statement, and storage should be freed with the DEALLOCATE statement.

Example:

```
integer b(:)
allocatable (z,a(:),b,c(:,:,:))

allocate (a(6),b(-3:2),c(100,100,100))

do i=1,6
   a(i) = i
   b(i-4) = i-4
end do

deallocate (a,b,c)
```

- Limited array notation

The *-f90* flag enables support for Fortran 90 array objects, array sections, and array expressions in assignment statements.

Example:

```
integer t(300,300)
real q(300,300)

C  triangular matrix initialization
   do i=1,300
      t(i:300,i) = i
   end do

   q = 7
   t = t * 2 + q
   end
```

- Assorted Fortran 90 Array Intrinsics

The following Fortran 90 array intrinsics are supported under the *-f90* flag:

MERGE, SPREAD, PACK, UNPACK MATMUL, DOT_PRODUCT, TRANSPOSE, SUM, PRODUCT, MAXVAL, MINVAL, COUNT, ANY, ALL, MAXLOC, and MINLOC.

5.9 Miscellaneous Changes

The preprocessor, *fcpp1*, which shipped with CONVEX FORTRAN V6.1, has been deleted. The functionality it provided has been included in the compiler in a more general optimization.

5.10 TASK COMMON

Thread local common blocks may be declared with the "TASK COMMON" keyword. The *-cfc* flag is required to use TASK COMMON. TASK COMMON blocks are allocated in thread local storage. If a program has concurrently executing threads, each thread will have its own copy of each TASK COMMON block. A program should only reference a TASK COMMON block within a parallel region. See Appendix G, "Cray FORTRAN Compatibility," of the *Language Reference Manual* for more details.

5.11 Faster Power Functions

Most of the power library routines have been rewritten to be faster and more accurate. The new routines have a smaller maximum error and a smaller average error compared to the previous release. Answers from the "**" operator may be slightly different than in previous releases.

5.12 Summary of New Flags

-f90 Allows use of array expressions, sections, and objects. Allows use of automatic arrays and allocatable arrays.

-nocc Disables runtime conformance checking of Fortran 90 array operations. All Fortran 90 array operations are checked for shape conformance by default. All possible compile time checks are performed, but some operations require a runtime check. This option turns off runtime conformance checking, which will improve execution speed, but will not check for illegal array operations.

-nopeel Disables the loop peeling optimization.

-noptst Disables the test promotion optimization.

-nosc Disables the short circuiting optimization in conditional expressions in IF statements.

-peel Increases the number of loops considered for loop peeling at optimization levels -O2 and -O3.

-peelall Maximizes the number of loops considered for loop peeling at optimization levels -O2 and -O3.

-ptst Increases the number of loops considered for test promotion at optimization levels -O2 and -O3.

-ptstall Maximizes the number of loops considered for test promotion at optimization levels -O2 and -O3.

-tm C3[48] Machine types for the -tm flag now include C34 and C38. C32 is also accepted (same as C2). C34J is also accepted (same as C34).

-xr Invokes the new cross reference facility.

-xra Used when several source files need to be cross referenced in one report.

-xro Invokes the old cross reference facility.

6

Fixes

The following software and documentation problems have been resolved in this release.

CPU #	PR #	X #	Symptom of fixed problem
11	14263	14637	Compiler accepts invalid program
25	17116	17178	FORTRAN source utilities
51	15116	15396	Compiler generates incorrect message
109	22397	21687	Profiler interface problem
109	22682	21816	Compiler terminates abnormally
109	22706	21828	Compiler terminates abnormally
116	20185	17178	FORTRAN source utilities
133	11352	12636	Executable generates wrong answers
147	6702	8938	Executable terminates abnormally
147	14851	15131	Documentation error
190	21743	21672	Executable terminates abnormally
202	17530	17519	Documentation error
202	22268	21513	Man page error
209	15277	15532	Documentation error
223	18044	17519	Documentation error
223	19067	18785	Man page error
223	20868	16922	Executable generates wrong answers
223	21682	21029	Compiler terminates abnormally
223	22677	21861	Compiler terminates abnormally
228	17183	17251	Installation procedures problem
228	21102	20535	Compiler directive ignored by compiler
236	14882	15162	FORTRAN source utilities
8201	13596	14164	Profiler interface problem
8201	13836	14398	Compiler terminates abnormally
8201	14737	15013	Documentation error
8201	14738	15017	Documentation error
8201	14904	15189	Documentation error
8201	15021	15355	Documentation error
8201	15852	15532	Documentation error
8201	16001	16224	Documentation error
8201	19837	19493	Documentation error
8201	20954	20408	Installation procedures problem
8201	22247	21465	Run-time library problem
8222	20801	20382	Executable generates wrong answers
8234	13387	14000	Input/output problem
8234	22727	21847	Executable generates wrong answers
8259	17475	17412	Profiler interface problem
8259	18434	18257	Documentation error

CPU #	PR #	X #	Symptom of fixed problem
8259	18435	18258	Documentation error
8259	22329	21541	Compiler terminates abnormally
8259	22695	21839	Executable terminates abnormally
8259	22748	21864	Input/output problem
8284	14989	15261	Compiler terminates abnormally
8351	21429	14637	Compiler accepts invalid program
8354	16797	16922	Executable generates wrong answers
8355	16814	16928	Compiler terminates abnormally
8401	14282	14651	Compiler generates incorrect message
8401	14713	14637	Compiler accepts invalid program
8401	15426	15667	Documentation error
8401	16756	16880	Documentation error
8401	16932	17016	Documentation error
8401	17157	17244	Documentation error
8401	17862	17796	Miscellaneous
8401	18118	17979	Documentation error
8401	18237	18075	Man page error
8401	18238	18076	Documentation error
8401	18620	18457	Documentation error
8401	19377	19074	Man page error
8401	19957	19562	Executable generates wrong answers
8401	20749	20278	Documentation error
8401	21167	20569	Documentation error
8401	21559	20897	Compiler terminates abnormally
8401	22320	21540	Compiler terminates abnormally
8401	22772	21869	Compiler terminates abnormally
8417	17343	17412	Profiler interface problem
8447	17618	17575	Miscellaneous
8447	21800	15261	Compiler terminates abnormally
8447	22718	21858	Long compile time
8537	22317	21670	Executable generates wrong answers
8548	17975	18276	Executable terminates abnormally
8548	18071	17941	Executable generates wrong answers
8548	18072	17941	Executable generates wrong answers
8572	16901	16977	Documentation error
8605	19363	19072	Compiler terminates abnormally
8605	22708	21835	Executable generates wrong answers
8690	22571	21723	Compiler terminates abnormally
16389	11929	13086	Compiler terminates abnormally
16408	22433	21640	FORTRAN source utilities
16430	21289	15261	Compiler terminates abnormally
16446	22242	21462	FORTRAN source utilities
16455	17018	17176	Documentation error
16455	21165	20668	Compiler terminates abnormally
16456	18799	18601	Executable generates wrong answers
16462	19736	19465	FORTRAN source utilities
16471	22386	14637	Compiler accepts invalid program

A

Compiler Problems which produce Invalid Results

This appendix includes sample source code or a concise description for compiler problems that are known to produce invalid results.

```
--------------------------------------------------------------------------------
No X-ID for this problem

The use of NO_RECURRENCE and FORCE_VECTOR directives on a loop containing
a call to a user function may cause erroneous answers or runtime aborts
to occur.

This problem, although identified recently, has existed for many years
and is NOT a new problem with this release.

--------------------------------------------------------------------------------
X-ID  =       21464
PR-IDs=       22246
The following silently return wrong answers :
1. besyn with code compiled -p8, -pd8, -cfc
2. any quad precision bessel function ( dbes[sy][01n] compiled -cfc, -p8 )
3. dbes[jy]n -pd8

This problem, although identified recently, has existed for many years
and is NOT a new problem with this release.
```

B

Other Software Problems

This appendix includes sample source code or descriptive text for compiler problems that cause inappropriate compiler or library behavior.

These problems include compiler aborts, erroneous error messages, and failures to detect errors. Some of these reported problems have not been fully researched; therefore, the reported analysis may be incorrect.

Errors in the documentation provided with CONVEX FORTRAN are also described.

```
-------------------------------------------------------------------------------
X-ID  =        1170
Compiler can fail to generate "code unreachable" message.
         ...
         do idummy = 1, 3
            i = idummy
            if(0)x=x
            a(i) = 1
         ...
The compiler does not note that "x=x" statement is unreachable in the above
code fragment.
-------------------------------------------------------------------------------
X-ID  =        9768
PR-IDs=        7494
The resid value returned from TSTATE is not always correct.
-------------------------------------------------------------------------------
X-ID  =       11669
PR-IDs=       10157
Compiler directive is ignored by the compiler.
        program main
        dimension X(128,128), Y(128,128)
  c$dir begin_tasks
        CALL sub1(X(1,j), j)
  c$dir next_task
        CALL sub2(Y(1,i), i)
  c$dir end_tasks
        end
-------------------------------------------------------------------------------
X-ID  =       11840
PR-IDs=       10102
PR-IDs=       10443
PR-IDs=       12745
The compiler aborts with a system error in /bin/ld on certain very large
source files (i.e., with greater than 2**16 symbols) when the -db option
is specified on the command line.
-------------------------------------------------------------------------------
X-ID  =       12002
PR-IDs=       10638
The compiler terminates with an error indicating that the maximum number
of scalars exceeded at -O3.  The code consists of 1026 integer arrays
being assigned in 1026 DO loops.
With fc7.0, the following error occurs.
Warning: Out Of Memory - expect the worst!
Compiler abort/assert may have been caused by source on 4510.5.
-------------------------------------------------------------------------------
X-ID  =       12194
PR-IDs=       10639
```

Under some circumstances, at optimization level -O3 with the -re flag, the compiler's optimization report does not indicate that a parallel loop has been replicated (scalar). Instead, it indicates that a previous scalar loop has been replicated.

```
        integer a(128), b(10,10)
        integer i, j
        n1 = 1
        n2 = 10
        n3 = 1
        do i = n1,n2,n3
          b(i,i) = i
          do j = 1, 128
            a(j) = j
          enddo
        enddo
        end
```

X-ID = 13482
PR-IDs= 12423

The 'filename.ext;0' convention of VMS to designate the most recent version of file filename explicitly, is not recognized by INQUIRE. INQUIRE looks for an actual filename ending on ';0'. On the other hand the OPEN function works properly with this convention and opens the most recent version.

X-ID = 13564
PR-IDs= 12640

When tskipr is used to forward space a file to (or attempted over) end-of-file, end-of-file is detected but the eoff flag is not set for interrogation by the tstate function.

X-ID = 13878
PR-IDs= 13164

The FORTRAN tape io function "tstate" does not always reflect the correct status and position of the tape.

X-ID = 14338
PR-IDs= 13664

The compiler aborts on a large (6500 line) subroutine consisting of two large nested DO-loops. The compile options used are -O1 -pd8.

X-ID = 14350
PR-IDs= 13748

Attempting to backspace (via tskipr) over a eof marker is not properly handled when using asynchronous tape io.

X-ID = 14979
PR-IDs= 14672
PR-IDs= 19941

The following program aborts with a run-time error at -no.

```
      program test
      character*80 cbuf(2)
      call err (cbuf, 2)
      write (*, 10) (cbuf(i), i = 1, 2)
10    format (a)
      end
      subroutine err (cbuf, n1)
      character*80 cbuf(n1)
10    format('roses are red'/'violets are blue')
      write(cbuf,10)
      return
      end
```

The compiler is having problems reconciling the adjustable-length character array and the slash descriptor (/). Hard-code the length of the character array in the subroutine to avoid this error.

X-ID = 15074
PR-IDs= 14790
PR-IDs= 14791

The compiler does not do any limiting on the size of an output section in the SOFF object file. Instead, the compiler quietly wraps the size of the section

```
when it reaches 2^31-1.  It should do something like starting another section,
or telling the user about it, and aborting.
-------------------------------------------------------------------------------
X-ID  =       15451
PR-IDs=       15177
When compiling at -O2, the loop summary for an IF/GOTO loop gives
the wrong line number; it always quotes the line before the statement
that starts the loop.
        program main
          a=1.0
 1      a=a+1.0
        if(a.lt.1000.0) goto 1
          stop
          end
-------------------------------------------------------------------------------
X-ID  =       15572
PR-IDs=       15309
A loop carried dependency is reported where one does not exist thereby
inhibiting parallelization.
      program main
  c$dir no_parallel
      do i = 1, n
  c$dir force_parallel
        do j = 1, n
          call sub1 (j)
        enddo
        n = j
        call sub2 (n)
      enddo
      end
-------------------------------------------------------------------------------
X-ID  =       16856
PR-IDs=       16717
The compiler, with options -O0 -p8 (or -cfc), aborts while compiling a
code with many assignments like the following:
      wn(1)=.25d0-.375d0*p-.375d0*e+.5d0*pe+.125d0*p3+.125d0*e3-
            .125d0*p3*e-.125d0*p*e3
      wn(2)=.125d0-.125d0*p-.125d0*e-.125d0*p2+.125d0*pe+
            .125d0*p3+.125d0*p2*e-.125d0*p3*e
      wn(3)=.125d0-.125d0*p-.125d0*e+.125d0*pe-.125d0*e2+
            .125d0*p*e2+.125d0*e3-.125d0*p*e3



-------------------------------------------------------------------------------
X-ID  =       17155
PR-IDs=       17087
Fsplit fails to see the name of a subprogram when it is contained on
a continuation line.
      subroutine
     1test1
      return
      end
      subroutine
     1test2
      return
      end
-------------------------------------------------------------------------------
X-ID  =       17565
PR-IDs=       17504
When a write only file (mode 200) is OPENed and a READ is attempted, an
appropriate error message is issued, but the contents of the file are
destroyed.
      program test
      character*80      line
      open (1, file = 'test.in', status = 'old')
      read (1,*) line
      write(6,*) line
      end
-------------------------------------------------------------------------------
```

```
X-ID  =       17795
PR-IDs=       17876
VAX structures containing field names which are the same as Fortran's
relational operators confuses the compiler.
field names which appear to be relational operators when used in
relational expressions will confuse the compiler.  Enclose the
record.fieldname reference in parentheses, or rename the field.
Note that in a Fortran 77 context the structure references
are decidable, but beyond the capabilities of the current compiler.
In a Fortran 90 context, with array objects, the syntax is sometimes ambiguous
and undecidable, even with complete semantic knowledge of the program.
-------------------------------------------------------------------------------
X-ID  =       17902
PR-IDs=       17973
The asynchronous I/O routines (aset, awrite) do not work correctly if a program
is compiled without the -cfc flag, but then loaded using the -cfc flag.  The
reverse situation causes the program to yield unexpected results also.
-------------------------------------------------------------------------------
X-ID  =       17978
PR-IDs=       18120
The FORTRAN Master Index contains uncapitalized proper names as
entires.  For instance, look under "Hollerith" or "Cray".
-------------------------------------------------------------------------------
X-ID  =       18061
PR-IDs=       18098
In chapter 7 of the FORTRAN Optimization Guide, the description of loop
unrolling does not mention the restriction of unrolling only the
innermost loop.  There also is no description of the -ur flag.  Read pages
7-3, 6-3 and B-16.
-------------------------------------------------------------------------------
X-ID  =       19882
PR-IDs=       20328
real*16 sin/cos are only accurate out to 19-20 places.
-------------------------------------------------------------------------------
X-ID  =       20329
PR-IDs=       20845
When REWIND is called from a Fortran program compiled with -cfc that
performs asynchronous io using BUFFER OUT, it dumps core giving the
following error: "rewind: [139] synchronous io not allowed on this file."
The LRM doesn't state that REWIND cannot be used in asynchronous io.
         program main
         real a(1000)
         open (unit=13,form='unformatted')
         do i = 1, 10
            bufferout (13,0) (a(1),a(100))
         enddo
         b = unit (13)
         rewind (13)
         end
-------------------------------------------------------------------------------
X-ID  =       20816
PR-IDs=       21445
Compiler aborts when a statement function which concatenates two of its
arguments is called with an argument which is also a concatenation of
two character variables.
       call sub1 ( 'abc', 'def' )
       end
       subroutine sub1 ( s1, s2 )
       character*(*) s1, s2
       character*(9) tack
       tack( s1, s2 ) = ( s1//s2 )
       write (6,*) tack (s1//s2, s2 )
       return
       end
-------------------------------------------------------------------------------
X-ID  =       21248
PR-IDs=       21987
When using the options -O2 -ptstall, the compiler asserts with the following
message:
```

>>>>> COMPI

>>>>> See your sys
Error : Compiler e

The source consisted of many DO loops containing logical IFs with GOTOs.

X-ID = 21360
PR-IDs= 22131
fpp should check for read/write failures and call perror
if there was an error.

X-ID = 21401
PR-IDs= 22172
fc should check for read/write failures and call perror
if there was an error.

X-ID = 21629
PR-IDs= 22432
Pattern matching for MAX/MIN loops is slower at -O2 than when using -uo
on both the C3200 and 3400s.

```
        program tst
        real*8 a(128), b
        integer i,j
        do i=1,10
           a(i) = i
        enddo
        a(5) = 256.0
        do i=1,1280000
           call tst7(a,b)
           if (b .ne. 256.0) write(*,*)"Got the wrong answer"
        enddo
        end
        subroutine tst7(a, b)
        real*8 a(128), b
        do i=1,10
           if(a(i).gt.b) b=a(i)
        enddo
        return
        end
```

X-ID = 21672
PR-IDs= 21743
The compiler generates an executable that terminates abnormally when compiled
at -O1.

X-ID = 21682
PR-IDs= 22513
The man page for signal, specifically (man 3f signal) says to
See Also:
CONVEX Fortran User's Guide (chapter 8)
Chapter 8 in the User's Guide no longer exists.

X-ID = 21683
PR-IDs= 22515
The compiler suuports an environmental variable called FCOPTIONS which lets
the user select default compiler options. This is not discussed in either
the User's Guide or the Language Reference Manual.

X-ID = 21762
PR-IDs= 22625
The following additional information needs to be added to the FORTRAN User's
Guide, page 1-3, in the paragraph about the -ep option:

```
        This option is available only at optimization level -O3.
```

The compiler issues messages otherwise.

X-ID = 21763
PR-IDs= 22626
The discussion of the -ep option does not state that this option is only
good when used with -O3. Otherwise, the compiler driver issues a diagnostic.

X-ID = 21858
PR-IDs= 22718

```
The compiler suffers from excessive compile time/aborts when using the
following options:
    fc -c -O3 -na -vfc -pa test.f
--------------------------------------------------------------------------------
X-ID  =       21859
PR-IDs=       22737
When compiling with -O3 -cfc, the compiler terminates with the following
message:
Warning: Out Of Memory - expect the worst!
Compiler abort/assert may have been caused by source on 40.13.
fc: /mnt/lang/bin/fskel was terminated by signal 'SIGILL'
--------------------------------------------------------------------------------
X-ID  =       21864
PR-IDs=       22748
Given the following source code, the compiler produces an executable that
produces a 64 (not 32) character result.
         real*16 r
         r = 1.0
         write(*,'(z32.32)') r
         stop
         end
--------------------------------------------------------------------------------
X-ID  =       21865
PR-IDs=       22730
Using a user-supplied preprocessor with the -pp option causes the compiler
not to be able to find files included with the VAX-style include statement.
Instead, the compiler issues a message similar to the following:
fc: Error on line 3.0 of /tmp/pofc1003750.i: Can't find include file 'a.inc'
Replace the VAX-style INCLUDE statements with #INCLUDE statements and invoke
-fpp when compiling.  The FORTRAN pre-processor (fpp) precedes the
user-specified pre-processor and will avoid not finding the 'missing' INCLUDE
files.
--------------------------------------------------------------------------------
```