C Language Software Release Document

Software Release SR9.5

Part No. 005500

Revision 04

This document describes the C compiler (version 4.65) and library for DOMAIN software release 9.5.

APOLLO COMPUTER INC.
330 Billerica Road
Chelmsford, Massachusetts 01824

First Printing: January, 1987
Latest Printing: January, 1987

This document was formatted using the FMT tool distributed with the DOMAIN computer system.

Reader Notice

This document resides on line in the /doc directory. To print a copy of this document, use the PRF command with the -npag and -pr options.

```
PRF <file_pathname> -PR <printer_name> -NPAG
```

Contents

Section

CHAPTER 1 OVERVIEW OF C AT SR9.5

1.1 Optimization Improvements 1-1

1.2 New Warning Messages 1-2

1.3 Choosing a Compiler Version 1-2

1.4 Changes to CLIB 1-3

1.5 On-Line Example Programs 1-4

CHAPTER 2 INSTALLATION

2.1 Conventions and Terminology. 2-1

2.2 Planning the Installation 2-2

2.3 Additional Information About Administrative-mode . . . 2-3

2.4 Installation Procedure 2-4

2.4.1 Administrative Mode 2-4

2.4.2 User Mode 2-8

CHAPTER 3 DOCUMENTATION

CHAPTER 4 BUGS AND BUG FIXES

4.1 Bugs Fixed in The C Compiler at SR9.5 4-1

4.2 Known Bugs in The C Compiler at SR9.5 4-3

4.3 Bugs Fixed in CLIB at SR9.5 4-4

4.4 Known Problems in CLIB at SR9.5 4-6

CHAPTER 1

OVERVIEW OF C AT SR9.5

SR9.5 introduced the following four significant changes to the C compiler:

- The C compiler optimizes code better. (See Section 1.1 of these notes.)
- The C compiler now allows you to distinguish between short (16-bit) enum variables and long (32-bit) enum variables.
- SR9.5 contains two new insert files /sys/ins/gmr.ins.c and /sys/ins/gpr.ins.c. These files contain features required to make the full graphics performance enhancements that exist in this release available to CAE programs written in C.
- The C compiler now allows you to use enumerated values and variables wherever int's are allowed.
- We've corrected the way that the C compiler handles formal array parameters.

For details on the latter three changes, see the document entitled "Making the Transition to DOMAIN SR9.5."

In addition to these major changes, we've also corrected a number of small bugs. (See Chapter 3 of this document for details.)

1.1 Optimization Improvements

The SR9.5 version of the C compiler contains major improvements to the global optimizer. New or significantly improved optimizations include:

- Global register allocation of up to 256 variables per procedure.
- Removal of loop-invariant expressions from inside of loops.
- Elimination of redundant or "dead" stores; that is, values that are computed but never used.

- Revision of the way multiplication by a constant is done so that a shift-add sequence is used. The shift-add is usually twice as fast as a MUL instruction.

In addition, procedure call overhead has been significantly reduced in many cases. When no registers need to be preserved, the minimal overhead time has been reduced by 40 percent.

For the DNx60 machines, or other machines with a 68020-based processor, the compiler reorders instructions to improve throughput and avoid stalls.

1.2 New Warning Messages

Because of the changes to the optimizer, some of your existing programs that used to compile without errors or warnings might now compile with warnings. The document Making the Transition to DOMAIN SR9.5 describes conditions which can trigger these new warnings and suggests ways to eliminate the messages.

1.3 Choosing a Compiler Version

The Making the Transition to DOMAIN SR9.5 document also describes conditions under which you might need to use the older, SR9.2 version of the compiler. When you install the SR9.5 C compiler, you also will get the SR9.2 compiler. If you need to use the older version, compile your file(s) with the following command:

```
$ cc_sr9.2 filename [-options]
```

To compile with the SR9.5 version, use the usual command; that is:

```
$ cc filename [-options]
```

1.4 Changes to CLIB

This is a list of the functional changes we've made to CLIB since SR9.2:

- A nil default ACL (i.e., ACL $NIL) now causes a UNIX ACL to be constructed and applied to the file. This permits non-UNIX Apollo programs to create files with UNIX ACL's.

- We changed setjmp and longjmp. Now, if you pass zero (0) as an argument to longjmp, the call changes it to one (1) before performing the non-local goto. We provided this change for SVID compatibility. Also, calling longjmp now triggers the execution of any outstanding cleanup handlers encountered while unwinding the stack.

- We've improved timezone handling for Europe and the Far East. Prior to SR9.5, you had to separately set the timezone on the DOMAIN system and the DOMAIN/IX systems. At SR9.5, once you've set the timezone in the DOMAIN system, it's automatically set for the DOMAIN/IX system as well. In other words, the system automatically sets the 'TZNAME' environment variable to the correct value when the node is booted. (You can get timezone information from the 'tz' program if the 'TZ' environment variable is not set.)

- In 1987, the USA will change the range of dates for Daylight Savings Time. Therefore, we changed the ctime and localtime calls to reflect the new range.

- We now provide a new higher-performance version of 'malloc'. The new version eliminates the need for the 'set sbrk size' function. (The 'set_sbrk_size' function is now a no-op.)

- We improved the 'getrusage' call. It now returns resource usage information on delivered signals, and correctly returns all child resource usage information to the parent. The 'wait3' call now also returns the correct resource usage information.

- The new UNIX signal 'SIGAPOLLO' replaces 'SIGPWR'. (We no longer support 'SIGPWR'.) Faults not corresponding to UNIX signals are translated into 'SIGAPOLLO'.

- 'acl_cache' is now automatically checked at boot time and automatically flushed if necessary. This should eliminate the need to ever manually flush the cache.

- We added the following socket-related ioctl's:

```
SIOCSHIWAT     /* set high watermark */
SIOCGHIWAT     /* get high watermark */

SIOCSIFADDR    /* set ifnet address */
SIOCGIFADDR    /* get ifnet address */

SIOCGIFFLAGS   /* get ifnet flags */
SIOCSIFFLAGS   /* set ifnet flags */

SIOCSIFNETMASK /* set net addr mask */
SIOCGIFNETMASK /* get net addr mask */

SIOCGIFBRDADDR /* get broadcast addr */

SIOCGIFMETRIC  /* get IF metric */

SIOCGIFCONF    /* get ifnet list */
```

- The externals 'etext', 'edata', and 'end' are now supported by the linker (ld) and runtime system.

- Project lists are no longer set up automatically for System 5 users, though they are still set up automatically for BSD4.2 users. System 5 users must now set the 'PROJLIST' environment variable to get project lists.

- The system now always sets the process ID of orphaned processes to one (1).

- At SR9.5, the system always turns off the 'close-on-exec' flag on a file descriptor obtained through the dup system call. We did this to meet SVID semantics.

DOMAIN/IX users should also see the DOMAIN/IX Release Notes.

1.5 On-Line Example Programs

At SR9.0, we introduced the getcc system for retrieving on-line sample programs written in C. These sample programs illustrate features of the C language, and demonstrate programming with DOMAIN graphics calls and system calls.

For SR9.5, we've added some new programs to the system, deleted some obsolete ones, and corrected a few bugs. In some cases, the source code stored on-line no longer matches the source code printed in the manual. In these cases, you should trust the on-line example over the printed one.

We realize that the on-line examples require a lot of disk space. Therefore,

we recommend that the system administrator load only one copy of the examples on the network and that links be set up to point to this copy. This will allow all users to retrieve sample programs, but will minimize the amount of disk space required. For example, suppose that the system administrator loads the C examples onto a node named //file_server. If you want to access them from another node, you would create links by issuing the following commands:

```
$ crl ~com/getcc //file_server/domain_examples/cc_examples/getcc
$ crl /domain_examples/cc_examples @
$_    //file_server/domain_examples/cc_examples
```

After creating the links, you merely have to enter the following command to retrieve programs:

```
$ getcc
```

CHAPTER 2

INSTALLATION PROCEDURE

This chapter describes how to install the C language Version 4.65. You can add this software to a user node (one equipped with monitor and keyboard) or a DOMAIN server processor (DSP) that is running SR9.5 or a more recent version of the AEGIS or DOMAIN/IX operating system. If the user node or DSP is not running SR9.5 or a more recent version, follow the appropriate software update procedures as described in Installing DOMAIN Software (Order No. 008860) or in the appropriate release notes.

NOTE: The user node or DSP must have a minimum of 1800 blocks of available disk space for a successful installation of this software.

All C programmers should absolutely read the "Making the Transition to DOMAIN SR9.5" document before using the C compiler at SR9.5.

2.1 Conventions and Terminology

Before you start, make sure you understand these terms and conventions:

Work Node — The user node at which you perform the installation procedure.

Target — The directory into which you're installing software. The target can be a node entry directory (for example, //target) or any subdirectory (for example, //target/product). If the target is on a user node, then the work node and the user node can be the same node.

NOTE: When you are installing software to update a diskless node, the target is the node entry directory of the partner node.

Secure network — A network that uses a registry of user accounts and access control lists (ACLs) to control log-in privileges and access to files and directories. Note that an open network does not use a registry or ACLs.

Source area — An on-line master area of DOMAIN software. An administrator

installs software from the distribution media into the source area and users install software from the source area over the network. The source area can be a node's entry directory or any subdirectory.

Source media — The media (floppy disks, magnetic tape, cartridge tape, or another node in the network) that contains the software.

< > — Angle brackets (< >) enclose the name of a key on the keyboard.

2.2 Planning the Installation

There is one installation procedure. You can use the procedure in one of three modes: ADMINISTRATIVE mode, USER mode, or SPECIAL-CASE mode.

ADMINISTRATIVE mode creates a source area by copying the INSTALL program and the new software from the distribution media to the target. You use the administrative mode when no source area for this release exists in the network. See Section 2.3 for more information about administrative-mode installations.

USER mode involves copying your new software from a source area onto another node in the network; it's the simplest and most commonly used mode. You can install in user mode only AFTER an administrative-mode installation has initialized the source area with the INIT_SOURCE program.

Two default conditions apply to a user-mode installation. The defaults are:

- The INSTALL program automatically copies the new software over the network from the initialized source area, instead of asking you to specify the source area.
- The INSTALL program uses the SID "user.sys_admin" during the installation, rather than your own login SID.

To install in user mode, get the source area's pathname from your system administrator, then go on to Section 2.4.2.

SPECIAL-CASE mode involves special cases in which you need to override the user-mode defaults. The special cases are:

- You want to install software from an initialized source area on the network, but your own login SID gives you more rights to a target's protected directories than the default SID "user.sys_admin"
- You want to install software from a source other than an initialized source area (for example, source media)
- You want to install additional software in a source area that was initialized during a previous administrative-mode installation

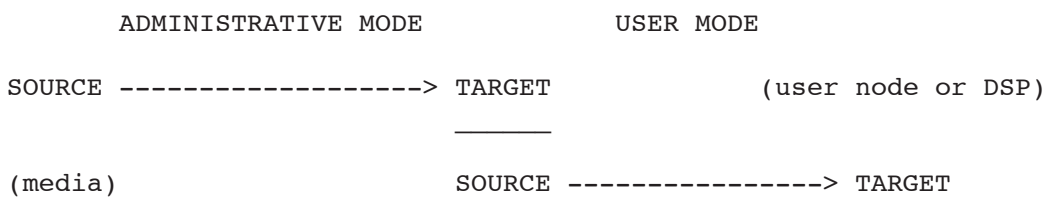
The installation procedure enters special-case mode when you invoke the INSTALL program with its -my_sid option. Specifying this option overrides the user-mode defaults, which means that the INSTALL program (1) uses your own login SID instead of “user.sys_admin” and (2) prompts for source media rather than automatically copying the software over the network from an initialized source area. In all other respects, special-case mode behaves like one of the other two modes of installation (your choice of source determines which one).

If you want to install software from an initialized source area on the network while using your own login SID, follow the directions for user-mode installations in Section 2.4.2. These directions include provisions for installing software in special-case mode.

If you want to install software from a source other than an initialized source area, or you want to install additional software in a previously initialized source area, follow the directions for an administrative-mode installation in Section 2.4.1. These directions also include provisions for installing software in special-case mode.

2.3 Additional Information About Administrative-mode Installations

The target of an administrative-mode installation generally serves as the source for subsequent user-mode installations (the administrative-mode target pathname is therefore the same as the user-mode source pathname). User-mode installations use both the INSTALL program and the software stored in the source area.



Your choice of target for an administrative-mode installation depends on whether you want the target node to RUN the software as well as act as a SOURCE for the software. If you also want the node to run the software, make the target the node’s entry directory (for example, //node). If you just want the node to contain the software, you should make the target a subdirectory (for example, //node/product/source area). In either case, users should then use the target of your administrative-mode installation as their source area.

If your installation target is a node’s entry directory, and you wish to use the target as a source area for future installations, the node entry directory must contain the following directories:

```
sys/help
sys/ins
domain_examples
```

When it copies software into an entry directory, the installation program does not create these directories if they do not already exist, because the program assumes they were deleted intentionally. The installation to the target succeeds, but subsequently the target is an incomplete source area. Future network installations from this source area will fail.

If the target is a subdirectory, the installation program creates the directories.

You can install different optional software products into the same source area or into separate source areas. Whichever route you take, you can then selectively install optional products on user nodes or DSPs from the source area(s).

If you have a secure network, you must have system administrator rights to install in administrative mode. Also, during the procedure you must initialize the source area by running the INIT_SOURCE program. This program marks the installation program in the source area with special privileges for subsequent user-mode installations, such as use of the SID “user.sys_admin” during installation. The INSTALL program can then install software in protected system directories, even though the user running the program does not have rights to modify these directories. In open networks, you create a source area but you don’t run INIT_SOURCE, since all users have rights to modify their system directories.

2.4 Installation Procedure

The following sections describe the administrative mode and the user mode of installation. To install software in special-case mode, consult Section 2.2 to determine which set of instructions you should follow.

2.4.1 Administrative Mode

NOTE: You can enter “q” or “quit” at any prompt in the INSTALL program to abort the installation and return to the Shell.

1. If you intend to create a source area for future installations, log on to a work node using a system administrator account (for example, my_name.sys_admin.%.%). Otherwise, log on using your own account (for example, my_name.%.%.%) .

2. Set your working directory to the installation target. This target will become the source area for user installations. It can be a node entry directory (like //node) or it can be any subdirectory created prior to the installation (like //node/product/source area). For

example:

wd //node (AEGIS Shell)

or

cd //node (DOMAIN/IX Shell)

3. Insert the source media into the drive and enter the RBAK command shown below. If you are using a tape cartridge, use the CT option shown in the example. If you are using a magnetic tape, use the M0 (Mzero) option. If you are using a floppy disk, use the F0 (Fzero) option.

   NOTE: If you are using multiple floppy disks, insert the floppy disk with the numeral 1 at the end of its label (for example, “FLP8_product_n.n_1”).

   All of the RBAK commands shown below create an INSTALL directory on the target and write the installation software to the directory. When entering the RBAK command, use lower-case characters to ensure visibility of the install directory in case-sensitive environments. Note that you can leave the source media in the drive for use in a later step; if you remove the source media after executing the RBAK command, the INSTALL program will later prompt you to re-insert the media.

   rbak -dev ct -f 1 install -as install -l -ms -force -sacl -du

   rbak -dev m0 -f 1 install -as install -l -ms -force -sacl -du

   rbak -dev f0 -f 1 install -as install -l -ms -force -sacl -du

4. Set your working directory to the INSTALL directory on the target. For example:

   wd //node/install (AEGIS Shell)

   or

   cd //node/install (DOMAIN/IX Shell)

5. Execute the INSTALL program and follow the prompts. If you are installing software in special-case mode, use the -my_sid option.

   For ADMINISTRATIVE MODE, type:

   install

   For SPECIAL-CASE MODE, type:

   install -my_sid

6. The program may prompt you to enter an installation type, based on what products already exist in the source area. If it does, answer OPT and proceed. For example:

```
**************************************
* SOFTWARE INSTALLATION -- Version n.n *
**************************************


Software installation TYPES are:

STD      --    Install standard software
RESTART  --    Restart the software installation
ACL      --    Set ACLs for existing software
CLEANUP  --    Run the Cleanup Procedure for ADD MODE installations
OPT      --    Install optional software (e.g., Pascal, FORTRAN)

Please enter installation TYPE: OPT
```

7. When the program displays the names of one or more optional products, enter the name of the optional product that you want to install. For example, to install C software you would type “CC”, as shown in the sample menu below.

```
Name     Description                    Disk Blocks Needed
                                       (Adding New Software)

CC       DOMAIN C Compiler SR9.5               1800

OTHER    If the optional product that you would like to install is not
         listed above, choose OTHER.
         *Note: When you choose OTHER, you are asked a few questions
         then shown a display of Apollo’s optional products. Check
         with your system administrator to determine which products
         your site has purchased and in which directory these products
         have been installed.

Enter the name of a single product you would like to install: CC
```

8. When prompted for the name of the target, enter the appropriate pathname (that is, the node entry directory or subdirectory that you specified in Step 2). For example:

   The TARGET is the node or subdirectory on which you are installing software. (e.g., '//my_node' or '//my_node/subdirectory')
   Enter Target: //node

9. The INSTALL program prompts for the source media. Enter your choice.

   Source MEDIA is one of:

CTAPE	--	Cartridge Tape
MTAPE	--	Magnetic Tape
FLOPPY	--	8" or 5 1/4" Floppies
NET	--	An area on the network with valid Software

   Enter Source Media:

10. The INSTALL program may ask you to insert the media into the drive. Insert the media and press <RETURN>.

11. The INSTALL program installs the software, listing each file it copies from the source media. If the software resides on multiple floppy disks, the program prompts you to mount (that is, insert) the next disk and to press <RETURN> to continue.

    When the INSTALL program finishes installing the software, it displays the following menu:

    Options:

RERUN	--	There were errors in the transcript pad and you wish to rerun the installation.
FINISH	--	The installation ran to completion error free. There is no additional optional software you wish to install.
CONTINUE	--	Install additional optional software.

    If you encountered any errors during the installation, correct the problem(s) and select RERUN. To locate error messages issued during installation, search backwards for the characters @? (an at sign followed by a question mark) in the installation's transcript pad.

    If there were no errors, choose CONTINUE or FINISH. Selecting CONTINUE brings you back to the beginning of the INSTALL program; selecting FINISH terminates the program. If you were installing software from magnetic tape, cartridge tape, or floppy disks, you can now remove the media from the drive.

12. If you have a secure network and you want the target of your installation to be used as a source area for future installations, run the INIT_SOURCE program (also run the program if you are adding software to a previously initialized source area). You must be logged in as a system administrator to perform this step.

    Invoke INIT_SOURCE at the Shell prompt. When prompted, enter the pathname of the new source area (which is currently the target of your administrative-mode installation). Here is an example:

    ```
    init_source

    Please enter the name of the SOURCE AREA
    for your network (e.g., '//NODE/SOURCE_AREA'):
    //node

    The source area for your network
    has been set to: //node
    ```

13. Perform this step only after completing an error-free installation and selecting FINISH.

    a. Use the Display Manager SHUT command to shut down the target node.

    ```
    <CMD> SHUT <RETURN>
    ```

    b. After the SUCCESSFUL SHUTDOWN message and the > prompt appear, reboot the node by typing the following at the prompt:

    ```
    > RE <RETURN>
    > <RETURN>
    MD REV xx/xx/xx
    > EX AEGIS <RETURN>
    ```

This is the end of the administrative-mode installation procedure.

2.4.2 User Mode

NOTE: You can enter “q” or “quit” at any prompt in the INSTALL program to abort the installation and return to the Shell.

1. Log on to a work node using your own account (for example, my_name.%.%.%).

2. Set your working directory to the INSTALL directory in the source area (if necessary, ask your system administrator for the pathname). For example:

   wd //node/install (AEGIS Shell)

   or

   cd //node/install (DOMAIN/IX Shell)

3. Execute the INSTALL program and follow the prompts. If you are installing software in special-case mode, use the -my_sid option.

   For USER MODE, type:

   install

   For SPECIAL-CASE MODE, type:

   install -my_sid

4. The program may prompt you to enter an installation type, based on what products already exist in the source area. If it does, answer OPT and proceed. For example:

```
**************************************
* SOFTWARE INSTALLATION -- Version n.n *
**************************************


Software installation TYPES are:

STD      --    Install standard software
RESTART  --    Restart the software installation
ACL      --    Set ACLs for existing software
CLEANUP  --    Run the Cleanup Procedure for ADD MODE installations
OPT      --    Install optional software (e.g., Pascal, FORTRAN)

Please enter installation TYPE: OPT
```

5. When the program displays the names of one or more optional products, enter the name of the optional product that you want to install. For example, to install C software you would type “CC”, as shown in the sample menu below.

Name	Description	Disk Blocks Needed (Adding New Software)
CC	DOMAIN C Compiler SR9.5	1800
OTHER	If the optional product that you would like to install is not listed above, choose OTHER. *Note: When you choose OTHER, you are asked a few questions then shown a display of Apollo’s optional products. Check with your system administrator to determine which products your site has purchased and in which directory these products have been installed.	

Enter the name of a single product you would like to install: **CC**

6. When prompted for the name of the target, enter the appropriate pathname (that is, a node entry directory or subdirectory). For example:

The TARGET is the node or subdirectory on which you are installing software. (e.g., ‘//my_node’ or ‘//my_node/subdirectory’)
Enter Target: **//node**

7. The INSTALL program may prompt for the source media. If so, enter NET.

Source MEDIA is one of:

CTAPE	--	Cartridge Tape
MTAPE	--	Magnetic Tape
FLOPPY	--	8” or 5 1/4” Floppies
NET	--	An area on the network with valid Software

Enter Source Media: **NET**

8. The INSTALL program may prompt for the source area. If so, enter the pathname (if you don’t know it, ask your system administrator). For example:

The SOURCE AREA is the node or subdirectory from which you are copying software. (e.g., ‘//node’ or ‘//node/subdirectory’)
Enter Source Area: **//node**

9. The INSTALL program installs the software, listing the name of each file it copies from the source area. Upon completion, the INSTALL program displays the following menu:

   Options:

   RERUN -- There were errors in the transcript pad and you wish to rerun the installation.

   FINISH -- The installation ran to completion error free. There is no additional optional software you wish to install.

   CONTINUE -- Install additional optional software.

   If you encountered any errors during the installation, correct the problem(s) and select RERUN (if necessary, consult your system administrator for assistance). To locate error messages issued during installation, search backwards for the characters @? (an at sign followed by a question mark) in the installation's transcript pad.

   If there were no errors, choose CONTINUE or FINISH. Selecting CONTINUE brings you back to the beginning of the INSTALL program; selecting FINISH terminates the program.

10. Perform this step only after completing an error-free installation and selecting FINISH.

    a. Use the Display Manager SHUT command to shut down the target node.

       <CMD> **SHUT** <RETURN>

    b. After the SUCCESSFUL SHUTDOWN message and the> prompt appear, reboot the node by typing the following at the prompt:

       > **RE** <RETURN>
       > <RETURN>
       MD REV xx/xx/xx
       > **EX AEGIS** <RETURN>

This is the end of the user-mode installation procedure.

CHAPTER 3

DOCUMENTATION

The “DOMAIN C Language Reference” manual (Revision 04, Order No. 002093) is current. However, programmers using C at SR9.5 should view the “Changes to C” section of the document entitled “Making the Transition to DOMAIN SR9.5.” In addition, readers should replace Section 6.2.17 of the manual with the following:

6.2.17 -Opt I -Nopt: Optimized Code

The -opt option is the default.

If you use the -opt option, the compiler optimizes the generated code. If you use DEBUG to debug an optimized program you may run into problems because of a fuzzy mapping between source code and generated machine code. See the “DOMAIN Language Level Debugger Reference” manual for details.

As part of the -opt command, you can specify a predefined level of optimization. The syntax is

```
-opt n
```

where n is an integer between 0 and 3. The higher the number, the more optimization that will be done. If you omit n or omit the -opt switch altogether, the compiler defaults to level 3. If you specify level 0, that is equivalent to -nopt, and -nopt tells the compiler not to optimize the generated code.

Following is a brief description of the optimizations you get at each level between 1 and 3:

1. Performs global common subexpressions and dead code elimination. Integer multiplication by a constant may be transformed into shift and add operations. Perform simple tree transformations, and possibly merge assignment statements.

2. Perform reaching definitions and global constant folding. (Also includes all of -opt 1.)

3. Perform live analysis of local variables, redundant assignment statement elimination, global register allocation, and instruction reordering. Also, use the CSE algorithm to search exhaustively for CSEs, and remove invariant arithmetic expressions from loops. (Also includes all optimizations for -opt 1 and opt 2.)

Because the compiler does more work at each higher level of optimization, it often takes longer to compile at those higher levels. Therefore, if you are just developing your program, and are compiling to find syntax errors, you might want to compile using a low number. Then, when the program is ready, you can compile with a higher number and so take advantage of all the optimizations.

Readers should also add the following note to the end of Section 7.1.

NOTE: When using sizeof(const) as an argument in a std_$call, you should explicitly type cast to the appropriate type expected by the operand.

CHAPTER 4

BUGS AND BUG FIXES

This chapter documents the bugs we've fixed since SR9.2 and the bugs that still exist at SR9.5.

4.1 Bugs Fixed in The C Compiler at SR9.5

We've fixed the following bugs at SR9.5:

- Prior to SR9.5, the DOMAIN C compiler incorrectly handled formal array parameters. We've corrected this problem for SR9.5, and the corrections are detailed in the document entitled "Making the Transition to DOMAIN SR9.5."

- Prior to SR9.5, the C preprocessor reported an error if it found a parameterized macro without any arguments but with spaces between the parentheses. For instance:

  ```
  define FOO()     /* This definition caused an error. */
  ```

  The preprocessor should not have produced an error, so we fixed it for SR9.5.

- Prior to SR9.5, an assignment of the following form would not work correctly:

  ```
  a_char_variable = ! (a && b)
  ```

  We fixed this bug at SR9.5.

- At SR9.2, the C compiler was confused by a declaration and definition that looked something like this:

  ```
  extern char foo[];
  static char foo[] = "HI FRED";
  ```

  The problem was that the compiler would allocate zero (0) bytes for array foo based on the "extern" declaration, and thus ignore the

data initialization found in the “static” definition. We’ve fixed this problem at SR9.5.

- Prior to SR9.5, the C compiler was incorrectly reporting an error if you tried to take the address of a register array; for example:

```
register int a[10];
int *b = &(a[0]);       /* This line used to cause an error. */
```

  The SR9.5 C compiler no longer reports an error for this condition.

- Prior to SR9.5, the C compiler looped indefinitely when you created recursive struct or union declarations. That is, the problem appears if you specify a struct/union name as a member of the struct/union with the same name; for example:

```
struct bad {
             struct bad {     /* multiple declaration of “bad” */
                          int a;
                          char b;
                        } fld1;
              int fld2;
            } a;

...

if (a.fld1.b) /* The compiler loops infinitely */
              /* on this statement. */
      { ... }
```

  We fixed this problem at SR9.5. Such declarations no longer hang the compiler.

- Prior to SR9.5, the C compiler was making casting errors on expressions of the following form:

```
((double) ((float) (a_floating_point_constant)))
```

  In this case, the C compiler should have cast to float and then cast the result to double. Instead, the compiler simply ignored the cast to float and just cast to double. We fixed this problem at SR9.5 so that both casts are honored.

- Fixed a bug where we were disallowing static initialization of a var to a constant cast to float. It was an omission, which has now been fixed.

- Prior to SR9.5, the declaration of a function via a typedef would cause multiple global symbol error if the definition of the function itself was in the same file; for example:

```
typedef int foo();
foo f2;           /* declaration */
f2() {return 1;}  /* definition */
```

  We've fixed this bug for SR9.5.

- Fixed a bug where assignment of overlayed fields (in a union) was not getting emitted, for example:

```
union { long l; short s } u;
u.l = u.s;  /* The compiler generated no code */
            /* for this expression.*/
```

- A bug was reported where the compiler would emit a "no temp created" error if it encountered a post-increment of a bit field within a structure.

- Prior to SR9.5, the compiler would intermittently crash with a reference to illegal address. We fixed this problem at SR9.5.

4.2 Known Bugs in The C Compiler at SR9.5

The following bugs exist at SR9.5:

- The DOMAIN C compiler evaluates side effects in an assignment expression somewhat differently than most C compilers. For example, consider the following statement:

```
*a++ = *a;
```

  Now compare how DOMAIN C and other C's evaluate side-effects:

DOMAIN C	Other C's
*(a+1) = *a;	*a = *a;
a += 1;	a += 1;

  Strictly speaking, this is not a bug. After all, in C, the order in which side-effects are evaluated is implementation dependent. However, to avoid confusion, we wish to document this difference. We will be more compliant at a future release.

- The C compiler can generate bad code in two instances when you compile with the -cpu 3000, -cpu 580, -cpu 570, -cpu 560, -cpu 330, or -cpu 90 options. The first problem concerns a structure in which

a field is odd-aligned. For example, in the following struct, the xx field is odd-aligned:

```
struct a { char dum1;
           char xx;
           int yy;
         } *x;
```

If you assign constant values to two fields in such a struct, the optimizer may incorrectly merge the assignments. For example, the optimizer will generate bad code for the following code sequence:

```
x->yy = 100;
x->xx = 0;
```

To avoid this problem, use the -nopt option when you compile.

The second problem is that the compiler does not correctly generate array references to multidimensional arrays if part of the expression is parenthesized. For example:

```
(X[y]) [z]
```

To avoid this problem, either remove the parentheses or remove the -cpu option.

4.3 Bugs Fixed in CLIB at SR9.5

We've fixed the following CLIB bugs at SR9.5:

- 'get login' now works on pty's.
- Prior to SR9.5, when a tty device was opened with O_NDELAY, the flag was propagated to the tty nodelay bit which affected reads and writes. As of SR9.5, O_NDELAY no longer affects subsequent reads and writes on a tty device.
- stdio should have been returning an error code when you wrote to a file opened for reading only, or read from a file open for writing only. We've fixed it so that at SR9.5 CLIB returns the proper error codes.
- Prior to SR9.5, fcvt had a limit to the number of digits it could return. As of SR9.5, fcvt will behave in a unix-like fashion.
- fseek did not work correctly on unbuffered files. It now works correctly.
- At SR9.5, several printf bugs have been corrected. In particular, printf no longer has difficulty handling double-precision values. In

addition, it no longer prints a leading 0 on e format, nor does it drop the e on e format when the exponent exceeds two digits. Finally, printf now prints a leading + in the exponent in e and g format.

- The ioctl system call now complies with SVID requirements by returning EBADF on invalid file descriptors.

- The scanf call no longer discards white space in the input stream when processing a %[^foo] format.

- We've fixed a bug in 'fcntl'. Prior to SR9.5, the call some times incorrectly returned an error status from a successful call.

- We've fixed a bug in 'creat'. Prior to SR9.5, the call some times left an error status in errno even when the call succeeded.

- 'memset' and 'memcpy' now return values as per the SVID.

- 'rand' now uses the standard SVID algorithm.

- Symbolic links are now always created with a protection mode of 0777, to avoid problems with inadvertent checking of access rights when resolving names through symbolic links.

- 'getgroups' now returns the primary group as well as the project list.

- Writes of greater than PIPE_MAX bytes to a pipe now work correctly instead of returning an error status.

- We changed ecvt and fcvt to generate a SIGFPE when attempting to convert a value that is not a valid floating-point number. Formerly, ecvt and fcvt looped indefinitely.

- We fixed a bug in the open system call. The bug occurred when you tried to open a file in append mode. This incorrectly caused the system to set the file pointer to point to the end of the file. At SR9.5, the call now performs the actions described in the "DOMAIN C Library (CLIB) Reference" manual.

- The shmget system call now maintains a limit of 32 segments or 1K of pages.

- An fstat call on a named pipe now reports the number of bytes waiting to be read in the "st_size" field.

- The abort system call now works properly.

In addition, the following include files were inadvertently left out of the SR9.2.3 release of CLIB:

```
assert.h          (BSD4.2, SYS5)
memory.h          (SYS5)
search.h          (SYS5)
values.h          (SYS5)
varargs.h         (BSD4.2, SYS5)
sys/lock.h        (SYS5)
signal.h          (BSD4.2)
sys/file.h        (BSD4.2)
sys/resource.h    (BSD4.2)
```

These include files have been correctly added to the SR9.5 version of CLIB.

4.4 Known Problems in CLIB at SR9.5

The following is a list of known problems in CLIB at SR9.5:

- On DOMAIN/IX systems, a close updates the modified time of a file, while a write does not. In other UNIX implementations, close does not update the modified time, while write does.

- Normally, if anyone except the super-user writes to a file, the user-id (uid) bit for that file is turned off. On DOMAIN/IX systems, a write does not clear the setuid value.

- Only MBX channels opened through streams can be passed off to child processes via fork.

- The link and unlink calls do not work on directories.

- You cannot open a named pipe for both reading and writing.

In addition, there are some important changes to the DOMAIN system at SR9.5 that will affect CLIB users. All CLIB users should read the document entitled “Making the Transition to DOMAIN SR9.5.”