sc_prob

COLLABORATORS			
	TITLE : sc_prob		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 28, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

sc_prob

1.1 Common Problems

This help screen contains information about some of the problems that people frequently have with the compiler. The button "Changes to the Documentation" leads to a help screen which documents last minute changes to functionality, errors in the documentation that we didn't have time to correct, and any other information about the product that we think that you should know. Most of this information can also be found in the read.me file on disk 1. THE LAST MINUTE CHANGES HELP SCREEN IS THE ONLY PLACE WHERE THIS INFORMATION CAN BE FOUND. Please review both that screen and this screen carefully before calling Technical Support .

- Problems Converting to Version 6
- Problems Compiling
- Problems Linking
- Problems using CodePRobe (CPR)
- Problems with Library Functions
- Problems using the Editor (SE)
- Miscellaneous Problems

1.2 Problems Converting to Version 6

If you are having problems converting your Version 5 programs to run under Version 6, you will probably want to consult the documentation provided to guide you through this process. Refer to Appendix 5 of the User's Guide Volume 1, Converting from Version 5 to Version 6.

If you continue to have problems or questions after consulting this section, contact Technical Support for assistance.

1.3 Problems Compiling

1.4 Compiler Problem 1

I have installed my compiler correctly, but I keep getting the message "se not found" or "sc not found" whenever I try to do anything.

Your s:user-startup may not be set up correctly. The installation program places three assign statements and the path statement at the end of your s:user-startup file. If you are invoking any programs that create a new Shell (such as PopCLI) in your s:user-startup file before the path sc:c add statement, sc:c may not be in new shells created by the program. To correct the problem, move the three assign statements and the path statement above any calls to programs that create a new Shell.

If you are running under AmigaDOS Version 1.3, you may have a different problem. The installation program places the necessary assigns in a file named s:user-startup, which is called from s:startup-sequence. You may have to edit s:startup-sequence and make some changes. See Chapter 1, "Installing Your SAS/C Development System," for information on modifying your startup file.

1.5 Compiler Problem 2

After switching to Version 6, my old programs will not compile, or they compile with a lot of warning messages. Why is this happening and how can I fix it?

There are two solutions:

In Version 6, the compiler assumes that your code will provide prototypes for all functions. (In Version 5, you had to specify the -cf option for the compiler to identify missing prototypes.) If your code does not define prototypes for all functions, the compiler may produce several warning 100, 154, and 161 messages. Specify ignore = 100+154+161 to suppress these warnings, or use the genprotos option to generate prototypes for your functions and #include the resulting header file.

Unlike previous versions, the header files and libraries for Version 6 of the SAS/C Compiler are ANSI-compliant. The ANSI specifications affect both ANSI and non-ANSI functions and data names, so some non-ANSI functions and data names have also changed.

Convert your program to use the new ANSI-compliant libraries and headers. For any functions mentioned in error messages, read the description of the function in the SAS/C Development System Library Reference. This reference manual describes the parameter list, return value, and necessary header files required by each function. Make sure that you are including the correct header for each of the SAS/C or AmigaDOS functions that you are calling.

1.6 Compiler Problem 3

When I compile, the options I requested do not act as expected.

When you compile your program, whether you compile from the Shell or the Workbench screen, the compiler looks for an scoptions file in your current directory. If it does not find one, it looks for the file ENV:sc/scoptions. If it finds either of these files, it uses any additional options specified in the file. You can run the scopts utility or edit the scoptions file to review the options specified in these files. If you want to prevent the compiler from reading the options specified in any scoptions file, specify the resetoptions option as the first option in the sc command. If your program then runs as expected, you have a problem with the options that you have specified. If you specify the verbose option, the compiler tells you the location of the scoptions file used.

1.7 Compiler Problem 4

During compilation, I get the message "semi-colon expected" in a header file.

Check for extra characters at the beginning of your file in front of your #include statements or in one of your header files. You may have accidentally pressed a key while starting the editor.

1.8 Compiler Problem 5

I keep getting the message Error 25: Modifiable lvalue required , but I am declaring all my variables for a function just like it says in the library reference manual.

Do not include the const keyword in your declarations. The SAS/C Development System Library Reference contains many variables defined with the const keyword, especially pointers to strings. This keyword is required by the ANSI Standard and indicates that these variables will not be modified in the library function to which they are passed. The const keyword is present in the parameter list of the prototype that is found in the associated header file. This prototype, not the declaration you should include in your program, is what is described in the synopsis for each function in the SAS/C Development System Library Reference. Use the synopsis as a guideline for your declarations, but do not include the const keyword.

If you are not using the const keyword, then it is possible that there is an error in the documentation. If you suspect that this is the case, you can check it yourself by looking at the specified header file, or you may contact Technical Support . If you do discover an error, please contact Technical Support so that the error may be corrected with the next release.

1.9 GST Problem

I can't seem to get GSTs to work correctly and I can't find very much information in the manuals about GST.

There are two sections in the manual covering GST's. The first is in the User's Guide Volume 2 in the Utilities section. The second starts on page 21 in the Library Reference Manual. Both sections contain information needed to understand how to use GSTs.

1.10 Compiler Problem 7

I think that I am compiling with the correct header files and libraries, but I keep getting error messages like

```
 LONG getchar( void );
 include:clib/alib_stdio_protos.h 25  Error 72:  conflict with previous
   declaration
```

```
   See line 119 file "include:stdio.h"
```

One of Commodore's header files, <clib/alib_stdio_protos.h>, contains function prototypes that conflict with the ones in SAS/C's <stdio.h> file. While there are prototypes in Commodore's header files that have the same names as the functions described in the SAS/C Library Reference, these prototypes are for AmigaDOS versions of the functions. If you compare the prototypes in <clib/alib_stdio_protos.h> with the Synopsis sections of the functions in the Library Reference documentation, you will see that they do not match.

If you want to use the normal SAS/C libraries, you should NOT include the include:clib header file(s). The Synopsis section of each function in the Library Reference specifies which header file to include for the function described.

If you want to use the AmigaDOS versions the functions you call, include the include:clib header file and link with amiga.lib BEFORE linking with sc.lib. You will need to link explicitly with the slink command to do this.

In either case, you should not #include Commodore's file and the SAS/C header file in the same program.

1.11 Problems Linking

Problem 1 When I compile and link, I get messages reporting "Undefined Symbol: _CXnnn" and saying that the proper math library has not been included.

Problem 2 I am getting BLTN and/or CXERR errors, and I am using math libraries and header files.

Problem 3 When I link my project, I get undefined symbols. This code worked in Release 5.10.

Problem 4 I get the message "absolute reference to <name>" from the linker.

1.12 Linker Problem 1

When I compile and link, I get messages reporting "Undefined Symbol: _CXnnn" and saying that the proper math library has not been included.

In your program, you have accessed floating-point math routines, but you have not linked with the appropriate math library. Specify the appropriate math option for the library with which you want to link, such as math=standard . If you are using the link option to link your program, the compiler links with the correct math library. If you call the linker separately from the compiler, you must specify the correct math library in the slink command after the libs keyword. Refer to

SAS/C Development System Library Reference, Version 6.0 for more information about math libraries.

1.13 Linker Problem 2

I am getting BLTN and/or CXERR errors, and I am using math libraries and header files.

You may be including the math header files for one type of floating-point format and linking with libraries for another type of floating-point format. For example, you may be including m68881.h and linking with scmffp.lib. Different floating-point formats are incompatible and should not be mixed. Make sure that your included header files match the math library with which you are linking.

If you find no problem with the compatibility between your header files and your math library, contact the Technical Support Division.

1.14 Linker Problem 3

When I link my project, I get undefined symbols. This code worked in Release 5.10.

Some of the library symbol names have been changed to comply with the ANSI Standard for symbol names. The ANSI Standard states that any symbols that are not mandated by the Standard must begin with an underscore and a capital letter or two underscores.

When you compile your program, the compiler adds an additional underscore to the beginning of any symbols defined in C code. Therefore, a C symbol with two underscores has three underscores when the linker finally sees the symbol.

If your code refers to a symbol that has changed names from Version 5 to Version 6, the linker may produce an undefined symbol message when you link your program. If you receive this message, first check the SAS/C Development System Library Reference for different versions of the symbols with different numbers of underscores. Remember, the linker reports one more underscore on the symbol than the SAS/C Development System Library Reference shows.

Change your C code to refer to the new name for the symbol as described in the SAS/C Development System Library Reference. If this solution is not practical because of the number of references to the symbol, you can solve this problem in one of two other ways:

- Use the define option on the sc command or a #define statement in a header file to define the old name to the new name.

- Use the define option on the slink command to force all references to

one of the names to refer to the other. If you define the item in your code (in other words, you declare it without the extern keyword), you should use the define option to define the new name (used by the library) to the old name (used by your code). If you declare the item with the extern keyword and, therefore, use the library definition for the item, use the define option to define the old name (used by your code) to the new name (used by the library).

One very common technique used in Version 5 was to declare the main routine of your program as _main instead of main to bypass the overhead required to set up stdio and argument parsing. You should change this name to __main and add the keyword __stdargs to be compatible with Version 6. If you do not change this name, the linker issues a message saying that the symbol _main is undefined. (Remember, the linker puts in an extra underscore, so the symbol it is really looking for is main).

1.15 Linker Problem 4

I get the message "absolute reference to <name>" from the linker.

You are linking a resident program with the cres.o startup module, or you are creating a resident library or device, but the linker detected a reference to an absolute symbol. Make sure that the symbol being referenced is not being modified.

If the item is not being modified, then you can ignore the warning message, or you can suppress the message by adding the const keyword to the definition of the symbol.

If the symbol is being modified, you cannot link with cres.o or with the shared library startup module you are using.

1.16 Problems using CodePRobe (CPR)

Problem 1 I do not get all of the information I expect out of CodeProbe, or CodeProbe does not know about variables and functions it should know about.

Problem 2 CodeProbe does not work if I double-click on the Debug icon from the WorkBench screen.

1.17 CPR Problem 1

I do not get all of the information I expect out of CodeProbe, or CodeProbe does not know about variables and functions it should know about.

You may not be compiling with the correct debug option. If you are

compiling with nodebug or debug=line , try compiling with debug=sf .
For a complete description of each debug option, see Chapter 8,
"Compiling and Linking Your Program."

1.18 CPR Problem 2

CodeProbe does not work if I double-click on the Debug icon from the
WorkBench screen.

To invoke CodeProbe from the WorkBench screen, click on the Debug icon.
Then, hold down the Shift key and double-click on the icon for your
program's executable module.

1.19 Problems with Library Functions

Problem 1 printf (or sprintf, fprintf, and so on) does not print the correct values.

Problem 2 getchar does not work as expected.

Problem 3 My program compiles and links without errors, but I am getting the wrong results from some of my SAS/C function calls.

Problem 4 I am using the asctime function to convert the time to Greenwich Mean Time (GMT) correctly, but the result is exactly an hour (or two or three...) off.

Problem 5 I wrote a replacement function for one of the SAS/C library routines, but the library routines don't seem to be using my replacement function.

1.20 Library Problem 1

printf (or sprintf, fprintf, and so on) does not print the correct
values.

This problem can happen for one or more of the following reasons:

- You are asking printf to print a variable, but the variable conversion characters are incorrect. For example, you are trying to print a long integer using %d as the conversion character. A long variable requires a %ld if the shortint compiler option is active. Often, if you have one conversion character wrong, the rest of the line you are printing is also incorrect. Check your entire printf (or fprintf , etc.) format string carefully.

- You are trying to print a float or a double, but you have not linked with the proper math library. You must link with a

floating point math library to perform any floating point operations. If you are linking with a floating point library, make sure that it is positioned in your slink command line before sc.lib so that the linker can find the correct version of printf. If you specify the link option in the sc command, the compiler links the libraries in the correct order. If you specify the link option, you must still use the math= option to specify a math libary.

- You are linking with amiga.lib before the math library and sc.lib. You should specify amiga.lib as the last library in the slink command. If you specify the link option in the sc command, do not specify amiga.lib.

1.21 Library Problem 2

getchar does not work as expected.

Input and output on the Amiga system are buffered. On most other systems, getchar immediately gets a character from stdin (usually the keyboard). However, the CON: device on the Amiga system buffers its input, so your program does not actually read any characters until you do one of the following:

- fill up the console input buffer
- press Return
- enter the Amiga End-of-File character, Ctrl-\.

The SAS/C Development System libraries include two functions that you can use to deal with this problem:

getch gets a character from the console in RAW mode. The character is returned as soon as the user types it.

rawcon turns RAW mode on and off. rawcon(1) sets the console into RAW mode. rawcon(0) restores the console to non-RAW mode.

When the console is in RAW mode, any characters typed by the user are passed immediately to the application.

getch returns a single character from stdin just like getchar, but getch gets the character in RAW mode. If your console is not in RAW mode, using getch is equivalent to the following sequence:

```
rawcon(1);
c = getchar();
rawcon(0);
```

If your console is in RAW mode, getch is equivalent to getchar .

For more information about getchar, refer to SAS/C Development System Library Reference .

1.22 Library Problem 3

My program compiles and links without errors, but I am getting the wrong results from some of my SAS/C function calls.

You can get incorrect results from a function if you do not include the correct header files for the function you are calling. The header files contain prototypes for each of the SAS/C functions, as well as definitions for many common data structures required by these functions. By including the correct header file, you are providing the compiler with a prototype for the SAS/C functions you call.

The ANSI Standard requires that the compiler assume that functions that are called without the previous inclusion of a prototype return an int. For example, this means that if you call atof, which returns a double, without including math.h, the compiler assumes that atof returns an int and allocates only 4 bytes for the return value. When atof is called and the correct 8-byte double value is returned in the 4-byte space allocated for it, the value appears to be incorrect.

You may be including the incorrect header file for one of these reasons:

- In Version 6, the prototype is contained in a different header file than in Version 5. For example, in Release 5.10, the function atof was included in math.h. In Version 6, the atof prototype was moved to stdlib.h as required by the ANSI Standard. Check the SAS/C Development System Library Reference for the correct header file for the functions that you are using.

- You may be including the right header file from the wrong directory. For example, the include:dos directory contains many header files with the same names as those in the root level include: directory. However, these header files are AmigaDOS header files and not ANSI header files. The AmigaDOS header files do not contain the necessary prototypes and data declarations. For example, you may be including dos/stdio.h instead of stdio.h. Do not substitute header files from the include:dos directory for standard headers.

- You may be including header files from the proto directory instead of those in the root level include: directory. Although the files in the proto directory do contain prototypes for many of the SAS/C functions, these files are intended to be included in addition to the standard header files, not in place of them. Do not use header files in include: subdirectories as substitutes for the root level header files.

1.23 Library Problem 4

I am using the asctime function to convert the time to Greenwich Mean Time (GMT) correctly, but the result is exactly an hour (or two or three...) off.

Your machine is probably set to the default time zone, Central Standard Time (CST), and has not been set to your actual time zone. To correct the problem, set your machine's time zone environment variable to your actual time zone with the following AmigaDOS command:

```
setenv TZ= your-time-zone
```

For your-time-zone, enter your zone's standard three letter abbreviation followed by the number of hours difference between your time zone and Greenwich Mean Time. For example, for Eastern Standard Time, the command would be as follows:

```
setenv TZ=EST5
```

You can also initialize the _TZ variable as described in the SAS/C Development System Library Reference. Initialize the _TZ variable with the same time zone data as you would enter with the AmigaDOS command described above (for example, EST5). NOTE: The AmigaDOS environment variable does not have a leading underscore that the SAS/C data name has.

1.24 Library Problem 5

I wrote a replacement function for one of the SAS/C library routines, but the library routines don't seem to be using my replacement function.

If you write a function that has the same name as a library function, you need to add the __regargs keyword to your function or compile with the parms=both or parms=register option. For example, you may want to replace the SAS/C library function malloc with your own version of malloc. If you compile with the parms=stack option or define your version of malloc with the __stdargs keyword, then two versions of malloc are linked into your executable. If you use other SAS/C library functions that call malloc, these functions use the version of malloc in the SAS/C libraries. However, your functions that call malloc use your version of malloc. To make sure that all calls to malloc are using your version of malloc, define your version with __regargs or compile with the parms=both or parms=register option.

For information on using registerized parameters, refer to the description of the __regargs keyword in Chapter 11, "Using SAS/C Extensions to the C and C++ Languages," and the description of the parameters compiler option in Chapter 8, "Compiling and Linking Your Program."

1.25 Problems using the Editor (SE)

Problem 1 When I am in SE, I can't get the configuration screen to come up. What is wrong?

1.26 SE Problem 1

When I am in SE, I can't get the configuration screen to come up.
What is wrong?

You did not install the extra utilities when you ran the installer for SAS/C. The utility sekeymap.library must exist in sc:libs in order for se to bring up the configuration screen. To correct the problem, you should run the installer and install the extra utilities.

1.27 Miscellaneous Problems

Problem 1 How can I turn off all of the informational messages the compiler and linker produce every time I compile and link?

Problem 2 An annoying window opens when I run my program from the Workbench screen. How do I get rid of this window?

Problem 3 My program compiled and linked without any errors, but when I run it, my machine crashes. What am I doing wrong?

Problem 4 I can't seem to get GSTs to work correctly and I can't find very much information in the manuals about GSTs.

Problem 5 How can I turn off Control-c checking in my program?

1.28 Miscellaneous Problem 1

How can I turn off all of the informational messages the compiler and linker produce every time I compile and link?

You can specify the following options on the sc command line:

```
sc  nover    link  filename.c
```

The compiler option nover automatically turns on the linker option quiet if you also specify the link option.

1.29 Miscellaneous Problem 2

An annoying window opens when I run my program from the Workbench screen. How do I get rid of this window?

The SAS/C startup code opens this window when you run a program from the Workbench that uses stdin, stdout, or stderr. If necessary, you can change the attributes of this window. For information on managing this window, see Chapter 9, "Running Your Program from the Workbench Screen."

1.30 Miscellaneous Problem 3

My program compiled and linked without any errors, but when I run it, my machine crashes. What am I doing wrong?

The following list contains some of the more common errors that crash programs. Most of these errors are caused when your program accesses memory that it is not supposed to access.

- You have declared a pointer to an array or structure for which you have not allocated memory. This mistake is especially easy to make when calling functions that fill the array or structure with information, such as the fstat, lstat, and stat functions.

  To correct this problem, call malloc or calloc to allocate the appropriate amount of memory for the array or structure.

- You have assigned an inappropriate value to a pointer, so the pointer no longer points to a valid memory space. This type of error can be quite difficult to track down. You may want to run CodeProbe, Enforcer, or the Mungwall program if you think these types of errors may occur. Enforcer and Mungwall are programs that detect memory errors. They are available from Commodore Applications and Technical Support (CATS) and are shipped with Version 6.

- You have specified the nostackcheck option and are overwriting your stack. Do not specify nostackcheck until your program is completely debugged. The stackcheck option is the default unless you are creating a shared library using the libcode option.

1.31 Miscellaneous Problem 5

How can I turn off Control-c checking in my program?

To turn off Control-C checking, you can either compile your program with the nocheckabort option, or you can include the following function prototype and definition in your code:

```
void __regargs __chkabort(void);

void __regargs __chkabort(void)
{
}
```

This code replaces the function normally called for a Control-C with a function that does nothing.

To retain checking for Control-C, but to change the action taken when Control-C is pressed, include the following:

```
void __regargs _CXBRK(void);
```

```
   void __regargs _CXBRK(void)
   {
       /* your-code-here */
   }
```

If you include these function definitions in your program, do not compile with the nocheckabort option.

1.32 HELP

You have reached this Help window by either clicking on the Help button or by hitting the Help key within the SAS/C Help utility. Unlike other help topics present in the SAS/C Help utility, the Help help topic opens its own window. You must close this window by clicking on the close gadget or hitting escape before returning to the SAS/C help utility. You cannot hit the Retrace button to return.

To quit the SAS/C Help utility, select Quit from the Project menu or click on the close gadget. You may also hit escape.

Most help screens will display one or more buttons as part of the text. Clicking on these buttons will provide further information on the topic listed on the button. You can also reach these help topics through the main Contents screen or one of its sub-screens.

In addition, double-clicking in the help window will bring up a help screen for the word under the mouse cursor, if such a help screen exists.

While in the SAS/C Help utility, you may retrace your steps through the help screens you have selected by clicking on the Retrace button.

The Browse buttons will move you forward and backwards between help screens. The help screens are usually arranged alphabetically by command or topic.