STDERRF

A STDERR Redirection Utility

Version 1.10

User's Guide

MoonWare ShareWare
16005 Pointer Ridge Drive
Bowie, MD 20716-1744

raymoon@moonware.dgsys.com

Limited Warranty—STDERRF is provided AS IS.

MoonWare ShareWare makes no representations or warranties, expressed or implied, about the contents hereof, including, without limitations, any implied warranties of merchantability or fitness for a particular purpose, all of which MoonWare ShareWare expressly disclaims. Neither MoonWare ShareWare, nor anyone involved in the distribution of this software shall be liable for any direct, indirect, consequential, or incidental damages arising out of the use, the results of use, or inability to use such product even if I have been advised of the possibility of such damages or claim.

User's Guide Copyright Notice

Users for their personal use may reproduce this manual by printing it or copying it onto disk for automated access.

Software Copyright Notice

The STDERRF software is copyright © 1997, 1998 by Raymond Moon. It may be used without fee for non-commercial use. Commercial use requires registration.

Source Code Copyright Notice

The source code is copyrighted © 1997, 1998 by Raymond Moon. The source code, in part or whole, and any derivative work can be used royalty free for non-commercial use as long as my copyright is included in the documentation. To distribute the source code, in part or in whole, and any derivative work must include my copyright and source code restrictions. Commercial use requires registration.

Shareware Distribution

I have chosen this method of distributing the STDERRF software so you, the user, can obtain and use STDERRF with a minimum of inconvenience. Users are authorized to upload the zip file, STDERRF1.ZIP, to any Internet Node providing shareware software distribution or dial-up Bulletin Board System as long as the zip file has not been modified.

Charges For Distribution

STDERRF cannot be sold without the author's prior written approval. If a third party distributes STDERRF on a diskette, which third party can charge a small fee not to exceed $5.00 per diskette or $30.00 for CD-ROM to cover the cost of the media and any shipping costs.

Table Of Contents

Introduction

1.1 What is STDERRF

STDERRF is an STDERR (standard error) redirection utility. The DOS command line only allows for redirection of STDOUT, standard out, to be redirected from the screen to a file or device. This feature allows the software author to ensure that certain messages that require human intervention are not redirected into a file but displayed on the screen. STDERRF will force STDERR to be redirected to a file of the user's choice.

STDERRF and its assembly language source code provide a complete working example on redirecting STDERR to a file. STDERRF source code was too large to include it in the Frequently Asked Questions (FAQ) for the alt.lang.asm and comp.lang.asm.x86 newsgroups. Therefore, I have distributed it as Shareware so that STDERRF is readily available to anyone who wants it. This program is meant to be an instructional not a commercial product.

1.2 Summary of STDERRF Usage

STDERRF FILENAME EXENAME ["COMMAND LINE"]

where:

FILENAME is the name of the file into which the user wants the STDERR to be redirected. If the drive or path specified is not included, the file will be created in the default directory. If the file exists, STDERRF will ask the user whether to continue. If the user wants to continue, STDERRF will ask if the file is to be truncated, i.e., contents overwritten, or the new output appended to the current contents.

EXENAME is the file name and extension; the extension is required, of the program to execute

1.3 Hardware Requirements

- IBM PC/XT/AT/PS/2 or 100% compatible personal computer
- PC-DOS or MS-DOS 2.0 or higher
- 3K memory
- 80286 or better processor

1.4 User Responsibilities

As mentioned earlier, STDERRF is being distributed as ShareWare. STDERRF is meant to provide a full working example of how to redirect STDERR to a file. While there is no request for payment for non-commercial use, STDERRF is not released to the Public Domain. If you use my code or derive code from my code for non-commercial purposes, you must include my copyright in your source code and documentation, and include my restricted rights for usage if the source code is published. This is all I ask. If you want use my code or derive code for my code for commercial purposes, you must register your copy. See Registration, page 11. If you implement the algorithm without referring to my code, no restrictions apply.

1.6 User Comments

User comments are appreciated at any time. Please send any comments to:

raymoon@moonware.dgsys.com

or

Raymond Moon
16005 Pointer Ridge Drive
Bowie, MD 20716-1744

STDERRF Basics

2.1 STDERRF Purpose

The purpose of STDERRF is to teach and not be commercial utility. I did write a full-featured utility so that you could see a full and working implementation. The source code can be used for more than just learning how to redirect STDERR. My source code can show you how to have standardized startup code that will provide command line arguments in argc and *argv[] format just as in C. Also, this source code illustrates how to write a procedure that will assemble in any memory model. STDERRF.ASM also is a good example of how to load and execute another program with all the supporting structures. GETVLDC.ASM is a general-purpose library procedure that will return the user's response to a question. This procedure can go directly into your library. Lastly, EXIT.ASM shows any easy way to write an error exit routine that lets the user know why the program is terminating.

2.2 STDERRF Algorithm

The basic algorithm for STDERRF is:

1. Accepts three command line arguments:
    a. The full path and filename of the file into which STDERR is to be written.
    b. The full path and filename of the program to be executed.
    c. The command line for the program to be executed (should be delimited by double quotes to allow multiple arguments). This argument is optional.
2. Releases all memory above the program using Int 21h function 4ah so that there will be room enough to load and execute the designated program.
3. Opens the file from step #1.a above into which STDERR is to be written. If file exists, ask user to continue. If to continue, ask user if the file is to be truncated or output appended. Perform the requested actions.
4. Duplicates STDERR filehandle using Int 21h function 45h.
5. Uses Int 21h function 46h, force STDERR filehandle to have the filehandle of the opened file from step #2.
6. Uses Int 21h function 4b00h to load and execute the program from step #1.a. Use the default environment and the command line from step #1.c above.
7. Upon return from the function 4b00h, closes the file opened in step #2.
8. Restores STDERR using Int 21h function 46h to force STDERR to point to the filehandle saved from step #3 above.

Note that this algorithm can be used to redirect any standard device by just substituting that standard device for STDERR above.

2.3 STDERRF Source Files

The STDERRF source files with general descriptions are:

STARTUP1.ASM	General Purpose ASM Startup Code. Used to parse command line and frees up memory above the program
STARTUP.INC	Include file used to support STARTUPx.ASM that provides external references to data defined in SUDATA.ASM
SUDATA.ASM	Defines global variables used by STARTUP1.ASM
STDERRF.ASM	Main Procedure for STDERRF. Implements the basic algorithm
STDERRF.INC	Include file for all STDERRF source files.
GETVLDC.ASM	Library procedure to get a user's response to a question.
EXIT.ASM	STDERRF abnormal termination procedure.
PROCESOR.INC	Ensures all files assembled for the same processor.

STARTUP1.ASM implements steps #1 and #2 of the algorithm. STDERRF.ASM implements steps #3 through #8.

2.4 Detailed Description of Source Code Files

2.4.1 Startup1.asm

As stated above, this is one of my standard assembly startup procedures. This startup code performs the following functions:

1. Parses the command line into argc and *argv[] similar to C. Argv[0] is the first command line argument not the program name as in C.

2. Initializes the following global variables:
   a. DGRP, segment address of DGROUP
   b. STACK_BOTTOM, offset to stack bottom in DGROUP
   c. PSP, segment address of PSP
   d. ENVIRON, segment address of passed copy of the ENVIRONMENT
   e. OSMAJOR, integer part of OS system
   f. OSMINOR, decimal part of OS system

3. If DOS version is less than 2.0, aborts with error message.

4. Initializes DS and ES segment registers to DGROUP.

5. Shrinks memory down to size of program by releasing all memory above program.

This procedure is one of a set of startup code range from the very simple to the one above this one that also parses the environmental strings. The entire startup code is available in STRTUP10.ZIP.

2.4.2 Startup.inc

This file is new with Version 1.10 of STDERRF. The reason is that I updated STDERRF to the latest release of my Startup Code. So that all of my startup procedures can coexist in the same

.lib file, I had to move the common variables into another source file. This include file contains the externdefs to allow the startup code to reference these variables.

2.4.3 SuData.asm

This file also is new with this version. It is the common Startup Data file for all my startup procedures.

2.4.4 Stderrf.asm

This procedure is the heart of this program. As stated above, this procedure implements steps #3 through to the end of the algorithm.

STDERRF.ASM starts executing by telling the user who it is. Then, it checks to see that there are at least two command line arguments, the redirection file and the program to execute. If there is not at least two arguments, the program displays an error message and terminates.

The next step is the first that implements the algorithm. A duplicate file handle for STDERR is created. To be a well-behaved program, STDERR needs to be restored at the end of the program so a copy of it must be saved. I save it to a global variable because the EXIT procedure checks to see if it is zero. If it is not zero, the EXIT procedure will know that STDERR has not been restored and will attempt restore it.

The next step is to initialize the command tail for the program to be executed. The command tail is copied into the last 128 bytes of the PSP of the program to be executed. I created a structure to make the addressing of its various parts easier. The structure is:

```
CommandTail_S   struc
ctLength        db  ?           ; Length of command tail minus end CR
ctLeadingSpace  db  ?           ; Initial space to start command tail
ctText          db  126 dup (?) ; Command Tail
CommandTail_S   ends
```

The first byte is the length of the command tail that does not include the terminating carriage return. The command tail starts with a space; this is the space between the end of the filename typed in at the DOS prompt and the first command line argument. Microsoft recommends that a space is placed there, and I have followed their recommendation. That leaves 126 bytes for the command line. My code does not check the length because the command line for the executed program is one of three command line arguments so it must be less than 126. The code should be straightforward. I load the bytes one at a time and count each one. At the end, I terminate the string with a carriage return and store the length in the first byte.

The next nine blocks deal with opening the file into which STDERR will be redirected. I first try to open the file to see if the file already exists. If the file does exist, STDERRF asks the user if he wants to continue. If the user wants to continue, STDERRF asks the user if the redirection is to be appended or overwritten. Based upon the user's response, the read/write pointer is moved to the end of the file or the start of the file and zero bytes written. If the file does not exist, it is created.

Now, the next block forces STDERR to have the filehandle of the just opened file. When STDERRF opened the file, the NO_INHERITANCE flag was not set. Therefore, when

STDERRF loads and executes the target program, that program will inherit STDERR pointing to the file vice to the console.

Everything is now set to load and execute the desired program. The next block does this. Now I need to explain the ParamBlk structure. This structure is defined as:

```
LoadExec_S      struc
leEnvironment   dw    ?      ; Environment block segment address
leCommandTail   LPtr  <>     ; Address of Command Tail
leFCB_1         LPtr  <>     ; address of default FCB, #1
leFCB_2         LPtr  <>     ; address of second FCB, #2
LoadExec_S      ends
```

(The LPtr is another structure that allows me to address the segment and offset portions of a long pointer independently.)

The initialization of this structure is done in the .DATA segment. leEnvironment is set to 0 so that STDERRF's environment is used. This variable provides you with the capability to define a tailored environment for the executed program. The command tail is explained above. The last two entries are the default and second File Control Blocks (FCBs). Again, Microsoft recommends that these FCBs be initialized to blank FCBs. A minimum blank FCB is defined as:

```
EmptyFCB        db  11 dup (20h), 5 dup (00h)
```

You can see that both FCBs are initialized to the same empty FCB.

If the load and execute is successful, STDERRF restores STDERR by forcing it to point to the saved filehandle for STDERR. Then the redirected file is closed, and STDERRF terminates by returning the exit code of the executed program.

2.4.5 Exit.asm

While this is not a standard library routine of mine, it is a regular procedure in all my programs. This procedure follows the same format. When I encounter a terminal error in some part of the program, I call this procedure with an Equate stored in the AX that identifies the error type. The EXIT procedure uses this value as an index into arrays that store the address and length.

2.4.6 Getvldc.asm

This is a standard library procedure. It is actually new in that I modified an earlier procedure that was hard coded for the "(y/n)" response. Now, it is a general procedure. In fact, my first interaction had one question to the user as "truncate or append? (t/a): ". Looking at it, I realized that it should be "overwrite or append? (o/a): ". The modification was just in the string variable that contained the question and passed valid choices in STDERRF.ASM. No changes were required in GETVLDC.ASM. This is how procedures should be written.

2.4.7 Stderrf.inc

There is nothing unique there. I just have combined equates, macro, etc. used in two or more procedures in this include file. This ensures that the information is uniform in all source files.

2.4.8 Procesor.inc

This include file contains only the processor directive. Again, using this include file ensures that all files are processed using the same processor instruction set. It also provides a single location to modify to change the processor.

2.5 Code History

Version	Date	Remarks
1.00	5 Mar 95	Original
1.10	3 Oct 98	Corrected error when there are not any command line arguments to pass. CommandTail was not initialized correctly. Moved declaration to .DATA and initialized. Modified Code to use new Startup Code structure

STDERRF Error Messages

3.1 Error Messages

STDERRF displays two different types of error messages. The first type is displayed when a non-DOS related error condition occurs. The second is displayed when the error is DOS related. Error Messages are displayed on the screen when STDERRF must terminate due to an unrecoverable condition or at user direction.

3.2 Non-DOS Related Error Messages

The following error messages are the non-DOS related messages with an explanation for when they occur.

3.2.1 Usage

The usage message which explains the basic command line options is displayed whenever the number of command line arguments is less than two, the minimum allowed. This is a quick way to display the help when the program is executed without any command line arguments.

3.2.2 STDERRF Terminated At User's Request

This message is displayed when the file exists into which STDERR was to be redirected and the user requested for STDERRF to terminate.

3.3 DOS Related Errors

DOS related error messages display information on the nature of the error and the error was the result of a failed DOS call. Read the error message explanation and take to appropriate action to correct the problem and try again.

3.3.1 Unable to duplicate STDERR filehandle

This message is displayed when STDERRF can not create a duplicate filehandle for STDERR. This duplicate filehandle is required so that STDERR can be restored when STDERRF terminates.

DOS error codes are:

04h - Too Many Open Files. Since STDERRF at this point has only five standard files open and 20 files are available to programs, the limit must be in total files opened. Close some applications, or, at least, the files opened in other applications, and try again.

06h - Invalid Handle. STDERR was not available to STDERRF. STDERRF is a child program of another application that has closed STDERR. Execute STDERRF from the DOS command line.

3.3.2 Unable to create redirection file

STDERRF could not create the file into which STDERR was to be redirected. The following DOS error codes are possible.

03h - Path Not Found. Check the path specified for the redirection file as DOS could not find it. Correct and try again.

04h - Too Many Open Files. Since STDERRF at this point has only five standard files and one other file open and 20 files are available to programs, the limit must be in total files opened. Close some applications, or, at least, the files opened in other applications, and try again.

05h - Access Denied. The redirection file already exists in the specified path and is read-only or the file was to be created in the root directory that was full. Change the path of filename and try again.

3.3.3 Unable to force STDERR to redirection filehandle

STDERRF could not force the STDERR filehandle to the value of the open redirection filehandle. DOS error codes are:

04h - Too Many Open Files. Since STDERRF at this point has only five standard files open and 20 files are available to programs, the limit must be in total files opened. Close some applications, or, at least, the files opened in other applications, and try again.

06h - Invalid Handle. Something has corrupted STDERRF data. This error should not occur.

3.3.4 Unable to load and execute program

STDERRF could not load and execute the specified application. The possible DOS errors are:

01h - Invalid Function. This error should not occur. If it has something as corrupted STDERRF code segment. Try again. If error persists, remove other applications and try again.

02h - File Not Found. Check the spelling of the file to execute. Check the drive and path if used. Try again.

03h - Path Not Found. Check the spelling of the drive and path. Try again.

04h - Too Many Open Files. Since STDERRF has not exhausted the 20 filehandles normally available, the limit must be in total files opened. Close some applications, or, at least, the files opened in other applications, and try again.

05h - Access Denied. I do not know under what conditions that this error code is returned. If this error occurs and is repeatable, please contact the author with the circumstances.

08h - Not Enough Memory. Since STDERRF releases all memory above it, there still was not enough to load and execute the desired program. Remove any TSRs and try again.

0ah - Bad Environment. This error should not occur as STDERRF uses the passed environment.

0ch - Bad Format. I am not sure under what conditions that this error code is returned.

3.3.5 Unable to restore STDERR

This error should not occur. If this error occurs and is repeatable, please contact the author with the circumstances.

3.3.6 Unable to close redirection file

This error should not occur. If this error occurs and is repeatable, please contact the author with the circumstances.

Registration

Registration Information

Mail the registration to:

Raymond Moon

16005 Pointer Ridge Drive

Bowie, MD 20716-1744

STDERRF Registration Form

Name: _____

Company: _____

Address: _____

City, State & Zip: _____

Registration

Basic One Computer Registration ($5.00) $_____

Multiple Computer License ($1.00 each) $_____

TOTAL: $_____

Comments: