A Techniques Development Laboratory
puter Program NWS TDL CP 89-2

STRING SEARCH

Silver Spring, Md.
March 1989

U.S. DEPARTMENT OF COMMERCE / National Oceanic and Atmospheric Administration / National Weather Service

PREFACE

The Techniques Development Laboratory's (TDL's) computer program (CP) series is a subset of TDL's technical memorandum series. The CP series documents computer programs written at TDL primarily for the Automation of Field Operations and Services (AFOS) computers.

The format for the series follows that given in the AFOS Handbook 5, Reference Handbook, Volume 6: Applications Programs, Part 1: Policy and Procedures, published by the Office of Technical Services/AFOS Operations Division.

NOAA Techniques Development Laboratory
Computer Program NWS TDL

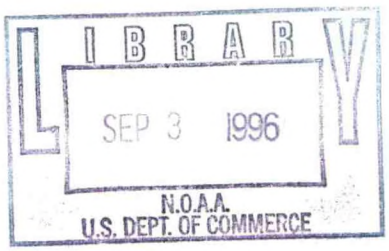
CP 83-1 Cross Sectional Analysis of Wind Speed and Richardson Number. Gilhousen, Kemper, and Vercelli, May 1983. (PB83 205062)
CP 83-2 Simulation of Spilled Oil Behavior in Bays and Coastal Waters. Hess, October 1983. (PB84 122597)
CP 83-3 AFOS-Era Forecast Verification. Heffernan, Newton, and Miller, October 1983. (PB84 129303)
CP 83-4 AFOS Monitoring of Terminal Forecasts. Vercelli, December 1983.
CP 83-5 Generalized Exponential Markov (GEM) Updating Procedure for AFOS. Herrmann, December 1983.
CP 84-1 AFOS Display of MDR Data on Local Map Background. Newton, July 1984.
CP 84-2 AFOS Surface Observation Decoding. Perrotti, September 1984.
CP 84-3 AFOS-Era Forecast Verification. Miller, Heffernan, and Ruth, September 1984.
CP 85-1 AFOS Monitoring of Terminal Forecasts. Vercelli and Norman, May 1985.
CP 85-2 AFOS Terminal Forecast Decoding. Vercelli, Norman, and Heffernan, October 1985.
CP 85-3 AFOS-Era Forecast Verification. Ruth, Miller, and Heffernan, October 1985.
CP-87-1 AFOS Terminal Aerodrome Forecast Formatting. Wantz and Eggers, July 1987.
CP-87-2 AFOS-Era Forecast Verification. Ruth and Alex, July 1987.
CP-87-3 Forecast Review. Wolf, July 1987.
CP-87-4 AFOS Monitoring of MDR Data Using Flash Flood Guidance. Norman and Newton, October 1987.
CP-87-5 AFOS Terminal Forecast Quality Control. Vercelli and Leaphart, December 1987.
CP-88-1 AFOS Terminal Forecast Decoding. Vercelli and Leaphart, August 1988.
CP-89-1 Structure Flow Diagram Generator. Adams, March 1989.

NOAA Techniques Development Laboratory
Computer Program NWS TDL CP 89-2

STRING SEARCH

Susan M. Adams

Techniques Development Laboratory
Silver Spring, Md.
March 1989



UNITED STATES
DEPARTMENT OF COMMERCE
Robert A. Mosbacher
Secretary

National Oceanic and
Atmospheric Administration
William E. Evans, Under Secretary

National Weather Service
Elbert W. Friday, Jr.
Assistant Administrator

TABLE OF CONTENTS

STRING SEARCH

Susan M. Adams

1. INTRODUCTION

Developing and maintaining application programs used by the Automation of Field Operations and Services (AFOS) system on the Data General S/230 Eclipse minicomputer are sometimes long and troublesome tasks. These programs consist of many FORTRAN and Assembly Language files which are separately compiled or assembled creating, respectively, relocatable binary (.RB) files. Executable program save files (.SV) are created by the Real-Time Disk Operating System (RDOS) using the Relocatable Loader (RLDR) utility. RLDR builds certain required tables, modules, directories, and the required Task Scheduler into the executable program file (Data General, 1979). It also links the relocatable binary files specified in the RLDR command (NWS, 1987), usually referred to as the "load line", into the executable save file. Because the programs are made up of separate files, global searches for a particular data object, CALL statement, SUBROUTINE, or common block, for example, become a tedious chore. Previously, these types of searches were done manually. The String Search software performs a global search, a search on the main program and all its non-library functions and subroutines, for a specified text string. This support software is designed to be a useful tool in maintaining, enhancing, and developing programs.

2. METHODOLOGY AND SOFTWARE STRUCTURE

The String Search software consists of a single program; FINDER. FINDER is designed to locate a word or group of words within a program and its subordinate routines. It displays the names of routines within which the string is found on the terminal from which the program is initiated. It writes to a file the name, line number, and line of source code containing the specified text string. FINDER accesses the program load line and the FORTRAN or Assembly Language files of its functions and/or subroutines. It creates a "Search" file for the program with a .SE extension on the program name. Fig. 1 illustrates the data flow for the String Search software. The overall program structure of FINDER is shown in Fig. 2.

3. PROCEDURES AND USE OF SWITCHES

FINDER should be installed in an applications directory with links from the directory where the source code and load line for the program to be searched reside. Several switches are available to pass information to the String Search software at runtime and to vary the output from the software. Switches are needed to pass to the software the program name and search string, along with the option of printing the names of all the routines called (not only those containing the desired string).

FINDER is initiated from the terminal by entering:

FINDER/A progname/N "text string"/S

The two required local switches, /N and /S, pass the name of the program to be searched and the search string, respectively, to FINDER. The program to be searched must have a load line (.LD) file in the directory being accessed. The search string may be one or more words, enclosed in double quotation marks. Single quotation marks may not be used. For example, the command:

FINDER GMOD/N "IWINDOW"/S

will search the main program, GMOD.FR, and all functions and subroutines contained in the load line, GMOD.LD, for the string "IWINDOW". When the string is located, the name of the file in which the string appears will be written to the terminal. The filename, line number, and line of source code containing the string will be written to the output file GMOD.SE (see Fig. 3). The optional global switch /A allows the user to specify that all routines are to be written to the output file, not just those containing the string (see Fig. 4).

4. CAUTIONS AND RESTRICTIONS

Due to the limited amount of main memory available on the Data General S/230 computers, several limitations have been set. Programs that exceed these limitations should not be used with FINDER. These limitations are:

1. The program load line may not exceed 50 lines in length. If this happens, the error message "FATAL ERROR - LOAD LINE TOO LONG" will be printed at the terminal.

2. The program load line may contain no more than 300 subroutines. If this limit is exceeded, the error message "FATAL ERROR - TOO MANY SUBROUTINES IN LOAD LINE" will be printed at the terminal.

3. Filenames may contain no more than 12 characters, excluding the extension. Files with names exceeding 12 characters in length will be ignored by the program and will not be searched.

The text string entered on the command line may contain one or more words, but it must be enclosed in double quotation marks (") if it has more than one word or if it contains any special characters. If a string containing more than one word is entered without the double quotes, FINDER will recognize only the last word. This will still locate the desired string, but it may also locate occurrences of the last word without the rest of the string. For example, the command

FINDER GMOD/N CALL INITAR/S

will find all occurrences of "INITAR". This does include all occurrences of the entire string, but may also include other lines (see Fig. 5).

When entering the specified text string on the command line, it cannot be enclosed in single quotation marks. If this happens, the desired string will not be located. The single quotation marks are considered to be part of the text string; therefore, the command

FINDER GMOD/N 'COMMON'/S

would cause a search for all occurrences of "'COMMON'", including the single quotes. If the command

FINDER GMOD/N 'CALL INITAR'/S

were entered at the terminal, the program would search for the text string "INITAR'".

If a text string includes a special character, such as a slash (/), the string must be enclosed in double quotes, even if it has only one word. The command

FINDER GMOD/N "COMMON/WIN"/S

would find all occurrences of the string "COMMON/WIN". However, if the string had not been enclosed in double quotes, the program would interpret the command

FINDER GMOD/N COMMON/WIN/S

as

FINDER GMOD/N COMMON/W IN/S

FINDER does not recognize the local /W switch, which is, therefore, ignored. The remaining portion of the string immediately preceeding the /S switch, is taken to be the text string. FINDER would search for all occurrences of the string "IN" in the program GMOD.

The String Search software is designed to be a useful tool in maintaining and updating other software. The running time varies slightly depending on the number of subroutines to be searched, the length of the subroutines, and the number of lines found containing the search string. The higher these numbers, the longer the program takes to run.

5. REFERENCES

Data General Corporation, 1979: Real Time Disk Operating System (RDOS) Reference Manual, Data General Corporation, Westboro, MA, 204 pp.

National Weather Service, 1987: National Weather Service AFOS Handbook, No. 5, Vol. 6, Pt. 2, National Weather Service, NOAA, U.S. Department of Commerce, (in preparation).

6. PROGRAM INFORMATION AND PROCEDURES FOR INSTALLATION AND EXECUTION

STRING SEARCH

PART A: PROGRAM INFORMATION and INSTALLATION PROCEDURES

PROGRAM NAME: FINDER

AAL ID: MSC013
Revision No.: 01.00

FUNCTION: This program searches the main program and its subroutines for a text string. The program name and text string are specified on the command line. When the string is located, the program or subroutine name is printed to the terminal, and the name, line number and line of source code are printed to an output file. An optional switch may be used to also print the names of subroutines not containing the text string. FORTRAN and Assembly Language source files to be searched are determined from the program load line. Functions found in the progam load line are searched; library routines are not searched.

PROGRAM INFORMATION:

Development Programmer:

Susan M. Adams

Location: Techniques Development Laboratory

Phone: FTS 427-7639

Language: FORTRAN IV/Revision 5.57
Macro Assembler/Revision 6.30

Maintenance Programmer:

Harry Lebowitz

Location: Techniques Development Laboratory

Phone: FTS 427-8065

Type: Standard program

Save file creation dates: FINDER.SV
Original release/Revision 01.00 - December 1988

Running Time: Approximately 2-3 seconds per 100 lines of source code. Running time varies with number and length of subroutines.

Disk space: Program files - 25 RDOS blocks

PROGRAM REQUIREMENTS

Program files:

NAME
FINDER.SV

Data files:

NAME	LOCATION	READ/WRITE	COMMENTS
<progname>.FR	User's Directory	R	
<progname>.LD	User's Directory	R	
<subrtn>.FR	User's Directory	R	
<subrtn>.SR	User's Directory	R	
<progname>.SE	User's Directory	W	Created by FINDER

LOAD LINE:

```
RLDR FINDER FINDREV CONVRT INITAR ISRCH RDARY OCHN TROUBL WMOV
  BG.LB UTIL.LB FORT.LB SYS.LB AFOSE.LB
```

PROGRAM INSTALLATION

1. Move FINDER.SV to the applications directory. A link should be made from the directory with the required source code and load line to the applications directory.

STRING SEARCH

PART B: PROGRAM EXECUTION and ERROR CONDITIONS

PROGRAM NAME: FINDER

AAL ID: MSC013
Revision No.: 01.00

PROGRAM EXECUTION:

1. From the terminal, enter:

   FINDER/A progname/N "string"/S

   Definition of required switches:

   .xxx/N - Main program name to be used. FINDER reads the load line file for the program, <progname>.LD, to determine files to be searched.

   xxx/S - Text string to search for. The string should be enclosed in double quotes. This allows for spaces and special characters within the string. Single quotation marks should not be used.

   Definition of optional switch:

   /A - Print all program/subroutine names including those not containing the specified text string.

2. Before execution, be sure that your program does not exceed the following limitations:

   a. The load line for the program may be no longer than 50 lines in length.

   b. The load line for the program may contain no more than 300 subroutines.

   c. Filenames may contain no more than 12 characters, not including the extension. Subroutines with names exceeding 12 characters will be ignored.

3. During execution, the names of the source files containing the specified string are written to the terminal. The procedure name, line number, and line of source code containing the string are written to a file, <progname>.SE. If the /A switch is used, all procedure names are printed to the terminal and file, not just those containing the specified string.

4. Upon completion, the output file, <progname>.SE can be viewed either displaying it on the terminal (enter TYPE <progname>.SE), or by printing a hard copy (enter PRINT <progname>.SE).

ERROR CONDITIONS

TERMINAL MESSAGES	MEANING
1. "INCORRECT SWITCH SPECIFIED/ SWITCH NOT FOUND"	Either /N or /S switch was not found on the command line. Re-enter the command.
2. "FATAL ERROR - LOAD LINE TOO LONG"	Load line contains more than 50 lines.
3. "FATAL ERROR - TOO MANY SUBROUTINES IN LOADLINE"	Number of subroutines in load line exceeds limitation of 300 subroutines.
4. "OPENING ..."	Cannot open program load line file specified with /N switch. Restore the file or link, and rerun program.
5. "CREATING ..."	Cannot create output file <progname>.SE. Clear the file and rerun.

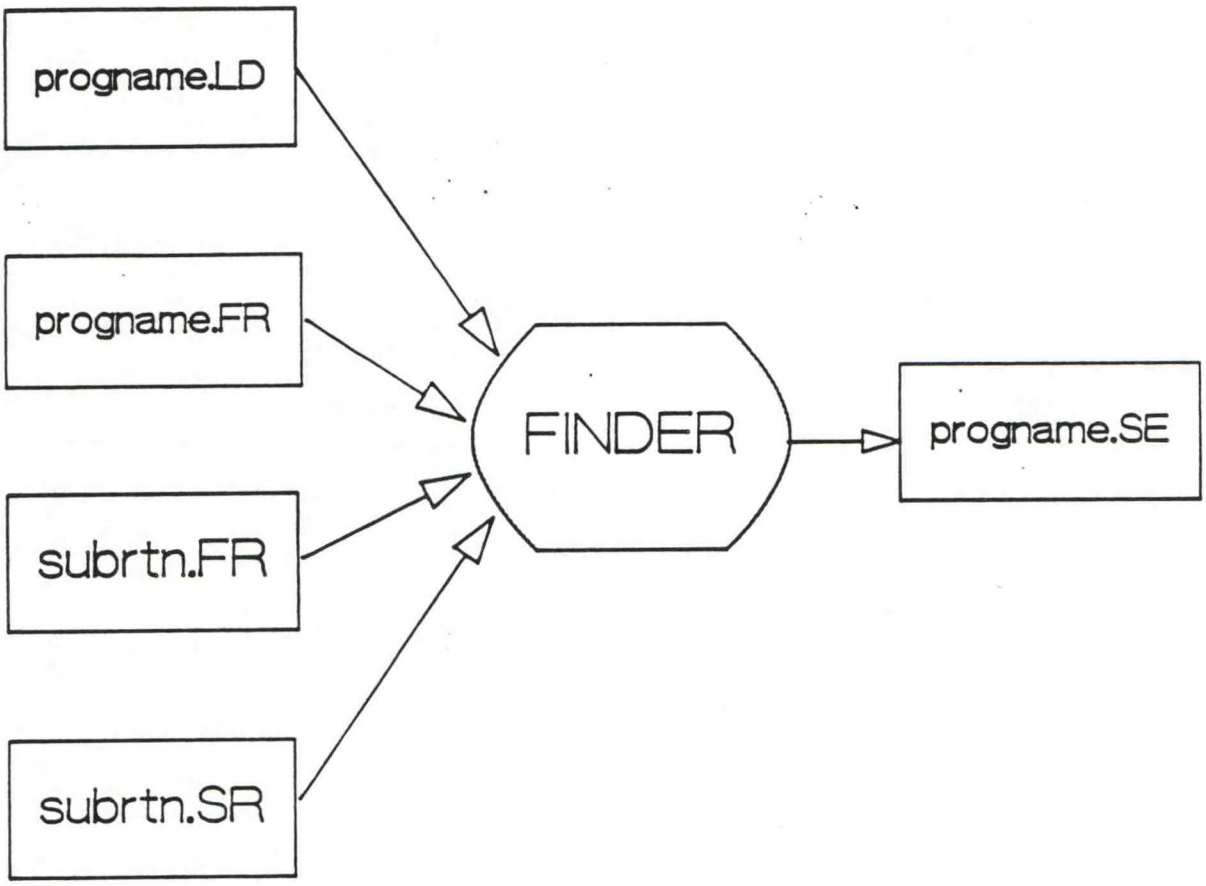


Figure 1. Data flow for String Search software

MAIN PROGRAM	SUBROUTINES	
FINDER		
.	INITAR	
.	OCHN	
.	.	TROUBL
.	RDARY	
.	.	INITAR
.	.	ISRCH
.	.	WMOV
.	TROUBL	
.	WMOV	
.	ISRCH	
.	CONVRT	

LOAD LINE

```
RLDR FINDER FINDREV CONVRT INITAR ISRCH RDARY OCHN TROUBL WMOV BG.LB UTIL.LB
  FORT.LB SYS.LB AFOSE.LB
```

Figure 2. Software structure and load line for program FINDER.

```
COMMAND:                FINDER GMOD/N "IWINDOW"/S

OUTPUT IN GMOD.SE

GMOD.FR

0123  C       IWINDOW()  =  AREA THROUGH WHICH REFERENCES TO EXTENDED MEMO
0151        COMMON/WIN/IWINDOW(1024)
0175        CALL MAPDF(NUM,IWINDOW,1,IER)
0202        CALL WRCLS(ICP1,IWINDOW,IGCHN,ICON,ICHNL,IOVL1,IOPT,IER)
0207        CALL WRCLS(ICP2,IWINDOW,IGCHN,ICON,ICHNL,IOVL1,IOPT,IER)


FMINT.FR

0048  C                 ICP  =  CURRENT POSITION IN IWINDOW
0080        COMMON/WIN/IWINDOW(1024)
0116        CALL MKTIME(IWINDOW,ICP,IXLN,IYLN,IDX,IDY,MPRJ,IRC)
0120        IWINDOW(ICP)=IOR(K1200,IXDMC)
0121        IWINDOW(JJ)=IYDMC
0122        IF(IZ.EQ.1.OR.IZ.EQ.ITHREE)CALL ISET(IWINDOW(ICP),12)
0123        IF(IZ.GT.1) CALL ISET(IWINDOW(JJ),12)
0125        CALL CONDAT(IWINDOW,IDATE,ICP,IRC)
0129        IWINDOW(ICP)=IOR(K1200,K)
0130        IWINDOW(JJ)=ILY(1)-16          ;MIDDLE OF CHARACTER IN Y DIRECTI
0131        IWINDOW(ICP+ITWO)='<<0>'       ;PUT '<' IN LEFT BYTE
0134        IWINDOW(ICP+ITHREE)=IOR(K1200,K)
0135        IWINDOW(ICP+IFOUR)=IWINDOW(JJ) ;MIDDLE OF CHARACTER IN Y DIRECT
0136        IWINDOW(ICP+5)='><0>'          ;PUT '>' IN LEFT BYTE
0144        IWINDOW(ICP+ITWO)=ITWO
0147        IWINDOW(ICP)=KVAL
0151        IWINDOW(ICP+1)=KVAL
0157        IWINDOW(ICP+ITHREE)=KVAL
0162        IWINDOW(ICP+IFOUR)=KVAL
0166  X     TYPE'IWINDOW()=',(IWINDOW(MN),MN=1,ICP)
0173        IWINDOW(ICP)=IOR(K1200,IXPR+IDX/ITWO)
0174        IWINDOW(ICP+1)=IYPR
0176        CALL WMOV(IPRH(NPTR),MPRJ,IWINDOW(IC))
0180        IWINDOW(ICP)=IOR(K1200,IXPCM)
0181        IWINDOW(ICP+1)=IOR(K6000,IYPCM)
0183        CALL WMOV(MAP(IP),LPCM,IWINDOW(IC))
0188        IWINDOW(ICP)=IOR(K1200,IXF)
0189        IWINDOW(ICP+1)=IYF
0191        CALL WMOV(MAP(IP,ITWO,IWINDOW(IC))
0196        IWINDOW(ICP)=IOR(K1200,IXDIG)
0197        IWINDOW(ICP+1)=IYDIG
0199        CALL WMOV(MAP(IP),LDIG,IWINDOW(IC))
```

Figure 3. Sample output of FINDER without local /A switch.

```
COMMAND:     FINDER/A GMOD/N "IWINDOW"/S

OUTPUT IN GMOD.SE

CONVRT.SR

CTASK.FR

GDCLR.FR

INTCLS.FR

INITAR.SR

INTZOPT.FR

RDCUR.SR

TROUBL.FR

WMOV.SR

GMOD.FR

0123  C         IWINDOW()  =  AREA THROUGH WHICH REFERENCES TO EXTENDED ME
0151           COMMON/WIN/IWINDOW(1024)
0175           CALL MAPDF(NUM,IWINDOW,1,IER)
0202           CALL WRCLS(ICP1,IWINDOW,IGCHN,ICON,ICHNL,IOVL1,IOPT,IER)
0207           CALL WRCLS(ICP2,IWINDOW,IGCHN,ICON,ICHNL,IOVL1,IOPT,IER)

GGDM.FR

CONDAT.FR

FBRDR.FR
```

Figure 4. Sample output from FINDER using local /A switch.

```
COMMAND:                FINDER GMOD/N CALL INITAR/S

OUTPUT IN GMOD.SE

INITAR.SR

0001     .TITL    INITAR
0002     .ENT     INITAR
0009  ;          SUBROUTINE INITAR(IBUF,NW,VAL)
0027  INITAR:    JSR       @.CPYL          ;


MKTIME.FR

0042             CALL INITAR(IASCII(ICP),MPRJ,' ¶')


MODIFY.FR

0156             CALL INITAR(IRV,MAXCOL*MAXROW,IOFF)


PCHR.FR

0071  C             INTRC,WRCLS,WMOV,INITAR
0186             CALL INITAR(NSEL(NPTR),N,"<11> ")


CONVAL.FR

0047  C             GTDIG,DISPER,MOVCIR,SGN,INITAR
0056             CALL INITAR(IDGB,NDIG,0)


GTDIG.FR

0049  C             DGBOX,WRCLS,INITAR
0057             CALL INITAR(IHOLD,3,IZERO)


SWXCL.FR

0036  C             TROUBL,INITAR,WMOV
0049             CALL INITAR(IDATA(NSTRT),NUM,0)


ASCICLD.FR
```

Figure 5. Sample output of FINDER using more than one word for the text string, not enclosed in quotation marks.