

Find

COLLABORATORS

	<i>TITLE :</i> Find		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		October 27, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Find	1
1.1	Find V2.3 Documentation	1
1.2	Copyright	1
1.3	Introduction	2
1.4	Starting Find	2
1.5	Find Commands	6
1.6	Problems	8
1.7	Things To Do	8
1.8	Program History	8
1.9	Compiling Find	9
1.10	Contacting the Author	9

Chapter 1

Find

1.1 Find V2.3 Documentation

Find V2.3 Dec 1993 Copyright (C) 1989-93
Andrew Kemmis - All Rights Reserved

TABLE OF CONTENTS

- Copyright
- Introduction
- Starting Find
- Find Commands
- Problems
- Things To Do
- Program History
- Compiling Find
- Contacting the Author

1.2 Copyright

Find V2.3 Dec 1993 Copyright (C) 1989-93
Andrew Kemmis - All Rights Reserved

Find is supplied free of charge for personal use only. Commercial use or commercial distribution is prohibited without the direct consent of the author. Non-commercial distribution must include all files listed below and they must not be modified. Appropriate *.info files may be added and/or modified to suit the distribution.

The files in this distribution are:

- Find - executable
- Find.guide - program documentation (this file)
- Find.c - main source code
- Find.q1 - qualifier parser input (Find.qi source)
- Find.qi - qualifier parser output (helps reading C source)
- makefile - Aztec 'C' 3.6 makefile (for Find and others)

Find comes with NO WARRANTIES WHATSOEVER. The author is not responsible for any loss or damage arising from the use of this program; the user takes all such responsibility.

The source provided is also Copyright (C) 1989-93 Andrew Kemmis - All Rights Reserved. It is provided to the user purely and for no other purpose than to show how some of the program was written.

In a non-commercial distribution no charge may be made for Find, other than a nominal copy fee.

However, permission is expressly granted to Fred Fish to distribute Find on his cool collection of disks or his own CD collection disks.

Although Find is free to non-commercial users, any donations would be gladly accepted, preferably programs that you yourself have written and find useful. See [Contacting the Author](#) for more information.

1.3 Introduction

Find is a tool for searching disk partitions. It SHOULD ONLY read your disk (i.e you can have it write-protected)

It should work with any 512 byte block structured virtual or physical disk partition i.e. it should work with any device that allows you to do any CMD_READ of 512 bytes at multiples of 512 byte Offsets - sequentially from any start block to any end block.

Find by-passes the FileSystem running on the device and does direct I/O to the device itself.

I wrote this program to help with another program of mine called Val. It should probably be distributed on this disk as well. Find can be useful on its own as well.

When Val is run in on an FFS partition it does not store the owner of every data block in the partition, just a bitmap table to say if a block has been used before or not. When it finds a pointer to a block already used it prints a message stating that the block is already in use and if Val is run in SEEK mode (see Val documentaion) or the block pointed to is any of the three types: Data, Unknown or Invalid (gave an error on reading) it does not know the block that originally pointed to the block already used. This is where Find comes in. Using the ANYRANGEDATA or ANYDATABLOCK qualifiers as documented below you can find the duplicate pointers to the block and thus take the appropriate action to fix the problem.

1.4 Starting Find

You can run Find on any disk (or partition) simply by typing:

```
Find /dev=disk:
```

However, disk: must be the Dos internal name for the partition
e.g. DF0: may be valid but Workbench1.3: is not if that is DF0:'s name.

Find has a number of options available at runtime. The options do not work in the standard way as yet (it's just a library function - that can and should be changed in the future) You specify options by typing at least the unique letters for that option (case is ignored) (shown in uppercase further down) after a '/' and if it is a value option using an '=' followed by the value e.g. to run Find searching for all blocks that point to blocks 50 to 100 displaying a count every 100 blocks read you could enter any of the following:

```
Find /Dev=hd0:/AnyRangeData=(50,100)/ShowReads/Count=100
Find /anyr=(50,100)/sh/c=100 /d=hd0:
Find /dEVICE=HD0: /aNYR=(50,100) /Show /cO=100
```

This does bring about the following problem (feature) you cannot do:

```
Find /d=/hd0: (if the device name starts with a '/')
```

The following is the output if you specify an invalid command line:

```
usAge: Find /Qualifiers
# Qual      Def      Min Max   Incl Excl List Atts
a Quiet      false
b Device     Mandatory 2    77      1
c Showreads  false      e
d Count      1000  1    10000000 c    e
e Tracks     false      cd
f Name       ""          1    30
g Ignorecase true
h Parent     0xffffffff 0x0 0x7fffffff
i Key        0xffffffff 0x0 0x7fffffff
j Hashchain  0xffffffff 0x0 0x7fffffff
k Extension  0xffffffff 0x0 0x7fffffff
l SEquencenum 0xffffffff 0x0 0x7fffffff
m Bytesize   0xffffffff 0x0 0x7fffffff
n BLockcount 0xffffffff 0x0 0x7fffffff
o Firstdatblk 0xffffffff 0x0 0x7fffffff
p Anydatblk  0x0      0x0 0x7fffffff 999
q ANYRangedata 0x0      0x0 0x7fffffff 98
r NExtdatblk  0xffffffff 0x0 0x7fffffff
s Ulong      0x0      0x0 0xffffffff 999
t SString    ""          1    80
u CACHe      88  1    10000
v CHecksum   false
w FIlenames  true
x FRomblock  0x0      0x0 0x7fffffff
y TOblock    0x7fffffff 0x0 0x7fffffff
z Validonly  true
{ Once       true
```

Def is the default value (a highlighted 'Mandatory' means no default - you must supply a value)

Min & Max are the minimum and maximum for numbers

Incl shows the qualifiers that must be specified if this one is

Excl shows the qualifiers that musn't be specified if this one is
List means that the qualifier accepts a list of up to the number of
specified elements or a file containing a comma or line
separated list using: /Qual=@filename
Atts shows the special attributes ('l' means the value is converted to
lowercase)

Numbers can be entered in Hex, (with a leading '0x'), Octal
(with a leading '0') or Decimal

The options are as follows:

QUIET

if set true don't display startup message at runtime.
/QUIET or /!QUIET

DEVICE

this qualifier is mandatory - you must specify it. It is just
the device name of the partition you wish to search i.e. it
cannot be just an ASSIGN to the partition or just the name for
the partition specified in the root block e.g. /DEVICE=hd0:

SHOWREADS

if true every Count blocks read display a line showing how many
have been read. /SHOWREADS or /!SHOWREADS

COUNT

the Count value used in SHOWREADS e.g /COUNT=100 (N.B. this
requires that you also specify /SHOWREADS)

TRACKS

display a line showing how many blocks required have been read
every time a track boundary is about to be crossed. /TRACKS or
/!TRACKS (N.B. you cannot specify this with SHOWREADS or COUNT)

NAME

match any block that has the name specified (see IGNORECASE as well)
e.g. /NAME=Fubar (see also STRING for cahracter translations)

IGNORECASE

if true /NAME and /STRING are not case sensitive. /IGNORECASE or
/!IGNORECASE

PARENT

match any block whose Parent pointer is the value specified
e.g. /PARENT=1234

KEY

match any block whose Key is the value specified e.g. /KEY=2345

HASHCHAIN

match any block whose Hash Chain is the value specified
e.g. /HASHCHAIN=3456

EXTENSION

match any block whose Extension is the value specified
e.g. /EXTENSION=4567

SEQUENCENUM

match any block whose Sequence Number is the value specified
e.g. /SEQUENCENUM=5

BYTESIZE

match any block whose Byte Size is the value specified
e.g. /BYTESIZE=67890

BLOCKCOUNT

match any block whose Block Count is the value specified
e.g. /BLOCKCOUNT=78

FIRSTDATABLK

match any block whose First Data Block pointer (either of the 2)
is the value specified e.g. /FIRSTDATABLOCK=8901

ANYDATABLK

match any block that points to any block specified in the list
e.g. /ANYDATABLK=(90,100,105)

ANYRANGEDATA

match any block that points to any block specified in the ranges
given in the list e.g. /ANYRANGEDATA=(86,198,250,307) means
anything in the range $86 \leq x \leq 198$ or $250 \leq x \leq 307$. If you
specify an uneven number of values the last one is ignored

NEXTDATABLK

match any block whose Next Data Block is the value specified
e.g. /NEXTDATABLK=6534

ULONG

match any block that contains any of the values specified in the
list anywhere in the block e.g. /ULONG=(0xffffffff,0x444f5300)

STRING

match any block that wholly contains the string specified
e.g. /STRING=#define N.B. the string can contain any of the
following which will be translated as follows:

```
\ a single \  
\b a back space  
\f a form feed  
\n a newline character  
\r a carriage return  
\t a tab  
\xmm the hex value mm  
\nnn the octal value nnn
```

a \ followed by any other character will produce exactly the same
2 characters

CACHE

this specifies how many blocks to read in advance each time a
block is required e.g. if /CACHE=5 and you are searching from
the beginning of the partition, the first read will read in all the
first 5 blocks and the next 4 reads will just return a pointer to

the cache buffer matching the appropriate block. Up to some limit this will speed up the Find program the larger the CACHE is, however beyond some point (maybe a few hundred) it should make very little difference.

CHECKSUM

if true, whenever a match is found, check that the block's Checksum is valid. If it is not valid, do not display the match. Don't use this if you are searching for FFS datablocks (i.e. the blocks themselves, not pointers to them) because they do not have a checksum and will usually always fail this check e.g. /CHECKSUM or /!CHECKSUM

FILENAMES

if true attempt to determine the full directory/file specification for a block if it gives an error or warning. /FILENAMES or /!FILENAMES

FROMBLOCK

the block to start searching at e.g. /FROMBLOCK=1000. If it is greater than the end of the partition, it will be set to the end

TOBLOCK

the block to finish searching e.g. /TOBLOCK=2200. If it is greater than the end of the partition, it will be set to the end (the default guarantees it will be the end)

VALIDONLY

this modifies the behaviour of the search for ANYDATABLEK, ULONG and ANYRANGEDATA. The default of true means that the block type must be either Control or Extension before the block will be considered as a candidate. If set false, every single block will be searched. I suppose this should also work with all the other search qualifiers - maybe in the next version (if anyone wants it) e.g. /VALIDONLY or /!VALIDONLY

ONCE

if true and the qualifiers ANYDATABLEK, ANYRANGEDATA or ULONG are specified with a list, the search will display the first match and then skip to the next block. With the case of ANYRANGEDATA and ANYDATABLEK, if both are specified, ANYRANGEDATA is checked first. If it succeeds ANYDATABLEK is not checked e.g. /ONCE or /!ONCE. If set false, and you are using ULONG with a list of values, if the block contains the same value more than once or the block contains more than one of the values - then for each match the block will give a message. The default of /ONCE is best suited for most searches - read the source code for more help

1.5 Find Commands

Find will read the disk from FromBlock (or Beginning) to ToBlock (or End) (whichever is smaller)

Find does not do a detailed check of each block it reads, it just looks at the Block Type and the Secondary Block type if Block Type is

Control (2.) If you specify VALIDONLY it will only check blocks of type Control, Extension or OFS Data (this is the default action). If you specify CHECKSUM, it will check each matching block's checksum and output the match only if the Checksum is valid.

Find will search the disk and output a message for each block that matches any criteria specified e.g. if you specify /Parent=555 and /Extension=777 then it will match either (both do not have to be true)

A brief explanation of a match message follows:

An example: "Match: ULong 0x00000000 Block Offset: 0x00000063, (98)
Type: File Workbench:DPointVII" (there is no forced line-break)

All message start with "Match:"

The next 9 characters is the right justified name of the match parameter

This is followed by the fixed length hex actual value of the match parameter found in the block

Then follows the words: "Block Offset:"

Then follows the fixed length hex value of the block that matched the value

The rest is varying in length:

Next follows a comma and the decimal value of the block in curved brackets

If the FILENAMES qualifier is false the message ends there otherwise:

If the type of the block is unknown the message ends with
"(Unknown block type)"

If the type of the block is not totally unknown then follows "Type:" and a string defining the type. One of "Root", "Directory", "File", "Soft Link", "Hard Dir Link", "Hard File Link", "Extension", "Data" or "Control???" - the last meaning the block type is Control but the secondary type is unknown

If known, next it attempts to display the [[Device:]Directory]/Filename pointing to the block starting at the block and working its way back up the directory structure. If the full name is more than 2048 bytes it will just show the end preceded by ".../" If any errors are encountered while getting the name the search will stop and an error message will be displayed at the front of the name - error messages start and end with the '*' character - 3 '*'s in a row means the back-tracking of the filename has been aborted, just one '*' means that it hasn't. The messages and descriptions are:

InvalidBlock The block to get this part of the name is outside the area of the partition (i.e. invalid)

BadBlock The block gave an error on reading

InvalidName The first byte of the name is the length of the name - in this case it was invalid i.e. 0 or > 30

NameContainsNull The file name has a null in it so was not displayed

1.6 Problems

I have not tried this program on many different device types. Below is the list which I have tried (and all have worked)

- Standard 880K low density drives
- A Meager Mega Ram Card with WD1002-05 and hard disks (on A1000)
- A590 SCSI hard-disk devices (on A1000)
- A1200 internal IDE device

I know you cannot use this program on the RAM: device or other same devices.

I have been using this program for a few years now so most parts of it should be pretty reliable.

1.7 Things To Do

Any suggestions would be very welcome.

OK it really does need some nice icons (send me suggestions and if I like and use them I'll put your name in the docs)

Find does not yet handle Directory Cache Blocks under the extended FFS.

There are plenty of other types of block information worth searching for - I just add them as I need them (or people request them :-)

See the VALIDONLY qualifier for another suggestion of something to do

I have tried to keep the program 1.3 compatible (especially since my main hard-disk is actually connected to an A1000) so don't suggest anything that is above 1.3 that cannot be optional.

Also need to tidy up and release a whole collection of disk utilities I have written (in times of dire need and have used to solve the problems.)
Look out for a few more, hopefully in the near future (like this one :-)

I have absolutely no idea what would be necessary to make this program work with disk blocks that were not 512 bytes in size (assuming that a full Root/Directory/FileHeader/etc. block is not just 512 bytes on this type of device spread over multiple blocks if <512 bytes or part of a block if >512)

1.8 Program History

OK there isn't much here - but maybe I'll be able to add to it as time goes on :-)

V2.3

First public release

(Added more FFS support to match changes in Val 2.3)

V2.2 and below:

Pre-release

1.9 Compiling Find

Find has been written to compile under Aztec C V3.6a, however not all of my include files or library objects have been provided in the distribution thus you should not (will not be able to) compile the program yourself from the distribution.

1.10 Contacting the Author

I can be reached with comments, suggestions, bug reports etc. at the following address till around March 1994:

Andrew Kemmis Smith
2/16 Nelson Street
Penshurst NSW 2222
Australia

or always via my parents place:

Andrew Kemmis Smith
32 Kenneth Street
Longueville NSW 2066
Australia

or by email:

and@praxa.com.au
