Find

COLLABORATORS			
	TITLE : Find		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 18, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Find

1.1 Find V2.3 Documentation

TABLE OF CONTENTS

1.2 Copyright

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.ql   - 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.

In a non-commercial distribution no charge may be made for Find WHATSOEVER.

However, permission is expressly granted to Fred Fish to distribute this
on the Fish and Aminet CD collection disks that they sell.

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 with this program as well. Find can be useful
on its own as well.

Find allows you to search a partition for strings, bytes and also
filesystem related information (e.g. all other blocks pointing at a
a block in a certain manner)

Val related information:
When Val is run 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: /options
```

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 a '/' followed by at least the unique letters for that option (case is ignored) (shown in uppercase further down) 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 data 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 value contains a '/')
```

The following is the default 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      l
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 Firstdatablk 0xffffffff 0x0 0x7fffffff
```

```
p Anydatablk   0x0    0x0 0x7fffffff     999
q ANYRangedata 0x0    0x0 0x7fffffff     98
r NExtdatablk  0xffffffff 0x0 0x7fffffff
s Ulong        0x0    0x0 0xffffffff     999
t STring       ""         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) The value shown above may change if some command
  line processing has been successful before displaying the usAge
Min & Max are the minimum and maximum for numbers
Incl shows the qualifiers that must be specifed if this one is
Excl shows the qualifiers that musn't be specified if this one is
List with a value 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: /Qualifier=@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 e.g.
  /DEVICE=hd0: It cannot be just an ASSIGN to the partition or
  the name for the partition specified in the root block

SHOWREADS
  if true, every Count blocks read displays 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 special character translations
  that are also allowed here)
```

```
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 tha range 86 <= x <= 198 or 250 <= x <= 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)
  (only on 4 byte boundaries)

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. If you wish to search for a '/' you will have to
  use \x5C or \134

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. This should be of no use if you are using a good
  disk cache program on your system.

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 ANYDATABLK, ULONG
  and ANYRANGEDATA. The default of true means that the block type
  must be either Control, Extension or OFS Data 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 ANYDATABLK, 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 ANYDATABLK, if both are
  specified, ANYRANGEDATA is checked first. If it succeeds
  ANYDATABLK 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)
Thus the current version of Find is a logical OR of all given search values

  A brief explanation of a match message follows:

An example:

Match:     ULong 0x00000000 Block Offset: 0x00000063, (98) Type: File
Workbench:DPointVII

N.B. 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 brackes

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 mesages
 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 Usage Examples

```
Most of these examples assume your partition is called dh0:
and they all use the minimum required letters for each option


Example: Find /d=dh0:/s/p=0xf00/ch

Meaning: Search device dh0:, displaying a block count every 1000 blocks
   searched, for any block that has block 0xf00 as it's parent
   block
   Only match blocks which have a valid checksum


Example: Find /d=dh0:/t/n=Foo/!fil

Meaning: Search device dh0:, displaying a block count every track read,
   for any block containing a name that starts with 'Foo' ignoring
   the case (e.g. directory or file)
   Don't bother to display the full directory path of the block.


Example: Find /d=dh0:/u=@list.dat/fr=0x2/to=0x1000/!o

Where:   list.dat is a file in the current directory containing:

   0xfefefefe, 0x12121212, 0x12345678

Meaning: Search device dh0: from block number hex 0x2 to block number
   hex 0x1000 for any block containing any of the 3 hex values
```

```
   0xfefefefe, 0x12121212 or 0x12345678 (only on 4 byte
   boundaries)
   If any block contains any of the above 3 values more than once
   list it again for each time another one of the 3 values is found
   within the block
```

1.7 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.
```

1.8 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)

Maybe the ability to do a logical AND of search requirements is needed
(e.g. /AND overiding the default OR)
```

1.9 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
```

1.10 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.
```

1.11 Contacting the Author

```
I can be reached with comments, suggestions, bug reports etc.
at the following address:

    Andrew Kemmis Smith
    7 Fleet Street
    Carlton NSW 2218
    Australia

or by email:

    and@praxa.com.au
```