Rtfs

Rtfs Command Shell Reference

Revised June 2009

EBS Inc. 39 Court Street Groton MA 01450 USA

http://www.ebsembeddedsoftware.com

TABLE OF CONTENTS

Rtfs Command Shells

By default the Rtfs sample project start-up routine calls the basic interactive shell program provided in rtfscommon\apps. The basic shell along with the RtfsProPlus extended sub shell, described in the next section, provides useful commands for maintenance, debug experiments and testing.

Basic Command Shell Reference

- Random Access File Operations
  - **RNDOP** - Open a random access file
  - **CLOSE** - Close a random access file
  - **READ** - Read and display a random access record
  - **WRITE** - Write data to a random access file
  - **SEEK** - Seek to a record in a random access file
  - **LSTOPEN** - List file descriptors opened by **RNDOP**
- Test and Miscelaneous Operations
  - **OPENSPEED** - Measure file creation and open speed
  - **REGRESSTEST** - Execute the Rtfs regression test
  - **ESHELL** - Enter the extended command subshell
  - **QUIT** - Exit basic command shell
- Failsafe Operations
  - **FS** autodisable D: - Disable callback based Failsafe configuration.
    - ***Note:*** *It is recommended that you execute the autodisable before executing enter, abort, exit, jcommit, jflush, commit or restore. These other commands will not behave as expected until automatic mode is disabled.*
  - **FS** enter D: - Begin Journalling
  - **FS** abort D: - Abort the current mount without flushing
  - **FS** exit D: - Flush Journal file and synchronize FAT volume, stop Journalling
  - **FS** jcommit D: - Flush Journal file but don't synchronize. Stop Journalling and abort. Leaves a restorable Journal file.
  - **FS** jflush D: - Flush Journal file but don't synchronize and continue Journalling. Creates a flushed frame.
  - **FS** commit D: - Flush Journal file and synchronize FAT volume
  - **FS** info D: - Display information about current Failsafe Session
  - **FS** clear D: - Reset Journal IO statistics
  - **FS** restore D: - Restore the volume from the current Journal
  - **FS** test D: - Run Failsafe test sequence
  - **FS** autoenable D: - Re-enable callback based Failsafe configuration.
- Drive and System Operations
  - **DSKSEL** - Set default drive
  - **DSKCLOSE** - Abort the current mount without flushing
  - **DSKFLUSH** - Flush all files and buffers
  - **DEVINFO** - Print all disks and their drive designators
  - **SHOWFILEEXTENTS** - Display block or cluster extents of a file
  - **SHOWDISKFREE** - Display free space map of a drive

- **SHOWDISKSTATS** - Display format information and access patterns
- **DEVICEFORMAT** - Low level media format
- **FDISK** - Partition a disk
- **DUMPMBR** - Print contents of a partition table
- **DUMPBPB** - Print contents of a volume's BPB
- **FORMAT** - Format a volume within a partition or disk
- **EXFATFORMAT** - Format an exFAT volume.
- **CHKDSK** - Perform a check disk procedure on a drive
- **EJECT** - Simulate a device removal event
- **RESET** - Reinitialize RTFS

- Utility Operations
  - **CD** **-** Set or display working directory
  - **DIR** **-** Print a directory listing
  - **RDIR** **-** Print a directory listing using reverse scan.
  - **ENUMDIR** - Recursive directory listing using **pc_enumerate()**
  - **STAT** **-** Print file properties
  - **GETATTR** - Print file attributes
  - **SETATTR** - Change file attributes
  - **MKDIR** - Create a directory
  - **RMDIR** - Delete an empty directory
  - **DELTREE** - Delete a directory and it's descendants
  - **RENAME** - Rename or move a file or subdirectory
  - **DELETE** - Delete a file
  - **CHSIZE** - Truncate or extend a file
  - **FILLFILE** - Create a file and fill it with a text pattern
  - **FILLHUGEFILE** - Create a file and fill it with a numeric pattern.
  - **READHUGEFILE** - Read a file and check a numeric pattern.
  - **COPY** - Copy a file to another
  - **DIFF** - Compare two files
  - **CAT** **-** Display contents of a file

RNDOP *Open a random access file.* This routine will open or reopen a file for use by our random access file I/O test commands **READ**, **WRITE** and **SEEK**. It must be given the file name and the record size for the file. The record size is stored internally and is used to pad write operations to the correct width. (Record size should not exceed 512). Use **CLOSE** to close a file that was opened with **RNDOP** and use **LSTOPEN** to display all open files. **RNDOP** does not return the file handle so use **LSTOPEN**.
Note: The file handles are always returned 0, 1, 2, 3... Use this knowledge if you want to use random access files in a script.

Example: RNDOP TESTFIL 200

CLOSE *Close a random access file.* This command closes a random access file that was opened with **RNDOP**. See **RNDOP** for a discussion of random access files.
Example: CLOSE 1

LSTOPEN *Display all open random access files.* This command lists all open

random access files along with their file handles. This is especially handy since after the initial open all accesses are done *via* the handle, and it is easy to forget which handle goes with which file.

Example: LSTOPEN

READ *Read and display a random access record.* This command reads data from the random access file and prints its value to the console. (See **WRITE** for how to write data to the file, **SEEK** for how to seek to a record in the file, **LSTOPEN** to list all random access files by handle and, **RNDOP** for how to open a random access file.).

Example:
```
RNDOP \TEST\FILE 100- open(returns handle=0)
SEEK 0 0
WRITE 0 "This is record zero"
SEEK 0 1
WRITE 0 This is record one"
SEEK 0 0
READ 0 - This will print "This is record zero"
CLOSE 0
```

WRITE *Write data to a random access file.* This command writes data to the current record of a random access file. The data is filled to the correct width (with spaces) internally. Multi word strings should be quoted.

Example: See the **READ** command for an example.

SEEK *Seek to a record in a random access file.* This command seeks to a record number in a random access file. It takes a file handle and a record number as an argument.

Example: See the example for the **READ** command

OPENSPEED *Measure the file creation and re-open speed.*

Example: OPENSPEED fname 0 500 – Open or create files fname0 to fname500

Example: OPENSPEED fname 1000 100 – Open or create files fname1000 to fname1100

REGRESSTEST *Execute the Rtfs basic regression test*.

Example: REGRESSTEST C:

ESHELL *Enter the extended command subshell*.

Example: ESHELL

QUIT *Exit the command shell.* This command exits the command shell and returns to the caller.

DSKSEL *Set default drive.* This command sets the default drive so that subsequent commands that do not explicitly contain a drive specifier will refer to this drive.

Example: DSKSEL D:

DSKCLOSE *Abort the current mount without flushing.* This command aborts the current mount without flushing. Any unflushed file operations or FAT buffers are lost. The next access forces a re-mount of the drive.

Example: DSKCLOSE D:

DSKFLUSH *Flush all files and buffers.* This command flushes all files and FAT buffers. If Failsafe is enabled the Journal file is updated but it is not flushed. That must be done using the **FS** command.

Example: DSKFLUSH D:

DEVINFO *Print the names of all disks and their drive designators*.

Example: DEVINFO

```
ERTFS Device List
==================
Device name: HOST DISK HOSTDISK.DAT Is mounted on G:
Device name: RAM DISK Is mounted on I:
Device name: STATIC ROM DISK Is mounted on J:
Device name: FLASH DISK Is mounted on H:
```

SHOWFILEEXTENTS *Display block or cluster extents of a file.*

Example: SHOWFILEEXTENTS C:\data\collected.dat Y (display extents in clusters)

Example: SHOWFILEEXTENTS C:\data\collected.dat N (display extents in blocks)

SHOWDISKFREE *Display free space map of a drive.*

Example: SHOWDISKFREE C: Y (show total freespace and all free fragments)

Example: SHOWDISKFREE C: N (show total freespace only)

SHOWDISKSTATS *Display format information and access patterns.*

Example: SHOWDISKSTATS C:

DEVICEFORMAT *Perform a device level format of a disk.* This routine will prompt you for the disk letter of the device that you would like to format. It then calls the device driver to perform a low level media format. Most drivers will not need to perform a device format and will simply return success. Some drivers, though, do perform low level media formats. For example disk and NAND simulators create a host, file based media simulation when this function is called.

Example:

In this example DEVICEFORMAT is called to format a 16 megabyte virtual drive.

```
CMD> DEVICEFORMAT
```

```
Enter the drive to perfrom device level format on A:, B: etc C:
Calling media format
What size do you want the hostdisk to be?
   1) FAT12 (4M)
   2) FAT16 (16M)
   3) FAT32 (1G)
   4) Current (~15M)
   5) Custom
   6) FAT32 (16G)
: 2
Making disk 16M.

CMD>
```

FORMAT *Format a disk or a partition on a disk.* This routine will prompt you for the disk letter of the device that you would like to format. It then calls the Rtfs API which formats the volume.

- If the drive letter is for a device with a dynamic device driver, then the device driver is called to retrieve format parameters.
- If the drive letter is for a partition inside a drive then the volume is formatted using the information stored in the partition table.

- If the drive letter is for a drive that is not partitioned then the volume is formatted using information from the device geometry provided by the device driver.

Example:

In this example a disk containing no partition table is formatted.

```
CMD> FORMAT
Enter the drive to format as A:, B: etc C:
No partition table found using default geometry
CMD>
```

In this example a disk containing a partition table is formatted.

```
CMD> FORMAT
Enter the drive to format as A:, B: etc C:
CMD>
```

HACKWIN7 *Make a device writable on windows or writable with Rtfs.*.

Example:

Make a device writable with Rtfs when running on Windows..

```
CMD> HACKWIN7
Enter the drive A:, B: etc C:
Invalidate MBR so volume it is writeable
Now Change it back so it is accesable on Windows..
CMD> HACKWIN7
Enter the drive A:, B: etc C:
Fixing MBR so volume it is readable on windows
```

EXFORMAT *Format a device for use with exFat*.

Example:

Format a device.

CMD> EXFORMAT
Enter the drive to format as A:, B: etc C:

FDISK *Partition a disk.* This routine will prompt you for the disk letter of the device that you would like to partition. It then prompts you for the size of each partition in cylinders and the type of partition (FAT12, FAT16, FAT32). It then writes the partition table(s).

Example:

In this example a partition table is created with just one partition that occupies the entire device.

CMD> FDISK
Enter the drive to partition A:, B: etc C:
Defining partition number : 0
This many cylinders remain: 520
This many sectors remain: 32760
Select the number of cylinders for this partition or return for all:
Select the partition type 0.) FAT16(0x04), 1.) FAT12(0x01): 0

In this example a device is partitioned into five separate partitions. Because there are more than 4 partition an extended DOS partition is created.

CMD> FDISK
Enter the drive to partition A:, B: etc C:
Defining partition number : 0
This many cylinders remain: 520
This many sectors remain: 32760
Select the number of cylinders for this partition or return for all:100
Select the partition type 0.) FAT16(0x04), 1.) FAT12(0x01): 0

Defining partition number : 1
This many cylinders remain: 420
This many sectors remain: 26460
Type X or x to stop selecting and partion now or..
Select the number of cylinders for this partition or return for all:100
Select the partition type 0.) FAT16(0x04), 1.) FAT12(0x01): 0

Defining partition number : 2
This many cylinders remain: 320
This many sectors remain: 20160
Type X or x to stop selecting and partion now or..
Select the number of cylinders for this partition or return for all:100
Select the partition type 0.) FAT16(0x04), 1.) FAT12(0x01): 0

```
Defining partition number : 3
This many cylinders remain: 220
This many sectors remain: 13860
An extended partition will be created if you do not select all..
Type X or x to stop selecting and partion now or..
Select the number of cylinders for this partition or return for all:100
Select the partition type 0.) FAT16(0x04), 1.) FAT12(0x01): 0

Defining partition number : 4
This many cylinders remain: 120
This many sectors remain: 7560
Type X or x to stop selecting and partion now or..
Select the number of cylinders for this partition or return for all:
Select the partition type 0.) FAT16(0x04), 1.) FAT12(0x01): 0
Elapsed time       : 48984 Milliseconds
CMD>
```

DUMPMBR *Print contents of a partition table.*

Example:

```
CMD> DUMPMBR
Enter the drive to read A:, B: etc C:
Partition #  ---->0
Boot ==  80 , Type ==  4 , Size == 18838 , Start == 63
SHead == 1 , Packed Cyl =  13f , Sector   =  63 , Cylinder =  1
EHead ==    1 , Packed Cyl =  2c7f , Sector   = 63 , Cylinder = 300
Partition #  ---->1
Boot ==  80 , Type ==  4 , Size == 13798 , Start == 18963
SHead == 1 , Packed Cyl =  2d7f , Sector   =  63 , Cylinder =  301
EHead ==    1 , Packed Cyl =  8bf , Sector   = 63 , Cylinder = 520
Signature aa55
CMD>
```

DUMPBPB *Print contents of a volume's bpb.*

Example:

```
CMD> DUMPBPB
Enter the drive to read A:, B: etc C:
   DOS Boot sig (0xe9)  : e9
   OEM Name            :MSWIN4.1
   Bytes per sector    :512
   Sectors per cluster  :2
   Reserved sectors     :1
   Number of fats       :2
   Num Root            :512
   Total Sector(16)    :18838
   Media Description    :f8
   Total Sector(32)    :0
   Sector per track    :63
   Heads              :1
   NumHide             :63
   DOS 4 Ext Sig        :29
   DOS 7 Ext Sig        :0
     MSDOS-4.0 EPB detected
```

```
    Sector per FAT     :37
    Drive              :2
    Binary Volume      :12345678
  Text Volume          :VOLUMELABEL
  Filesys Type         :FAT16
  Signature (aa55 ?)   :aa55
CMD>
```

CHKDSK *Perform a check disk procedure on a drive.*
Example: CHKDSK C: 0 - check drive C: don't write lost chains.
CHKDSK C: 1 - check drive C: write lost chains to .CHK files

EJECT *Simulate a device removal event*. The next access will re-mount the drive.
Example: EJECT C:

RESET *Reinitialize RTFS*. This command aborts all file opens in addition to mounting and releasing all buffers. If dynamic memory is in use it frees all dynamic memory. It then restarts Rtfs. The sequence that is invoked is **pc_ertfs_shutdown()** followed by **pc_ertfs_run()**.

Example: RESET

CD *Set or display working directory.* This command sets the default directory if an argument is supplied, otherwise it displays the current working directory.
Example: CD - Display working directory
Example: CD \usr\data - Change working directory

STAT *Print a file's properties.* This command calls the stat library routine and prints the results.
Example: STAT A:FILE.DAT

GETATTR *Print a file's attributes.* This command calls the **pc_get_attributes()** library routine and prints the results.
Example: GETATTR FILE.DAT

SETATTR *Change a file's attributes.* This command calls the **pc_set_attributes()** library routine to change a file's attributes.
Example: SETATTR FILE:DAT RDONLY*
The following values may be valid types for the attribute argument, RDONLY, HIDDEN, SYSTEM, ANORMAL

DIR *Print a directory listing.*
Example: DIR *.*

RDIR *Print a directory listing, reverse scan.*
Example: RDIR *.*

ENUMDIR *Print a recursive directory listing using* **pc_enumerate().**
Example: ENUMDIR \users\mydirectory my_*.*
Show directories(Y/N) Y
Show files(Y/N) */
Show volume labels (Y/N) N
Show .. (Y/N) N
Show . (Y/N) N

This sequence will recursively display all files and sub-directories with the prefix my_ in the subdirectory named \users\mydirectory. The "." And ".." entries for subdirectories will not be displayed.

MKDIR *Create a directory.* This command creates a directory.
Example: MKDIR \USR\NEWDIR

RMDIR *Remove a subdirectory.*
Example: RMDIR C:\TEMPDIR

DELTREE *Delete a directory and its descendants.*
Example: DELTREE C:\TEMP

RENAME *Rename a file.* This command will rename a file.
Example: RENAME C:\TES\JOSUF.TXT JOSEPH.TXT

DELETE *Delete a file.* This command deletes a file.
Example: DELETE A:\use\ASCII\budget.txt

CHSIZE *Truncate or extend a file.*
Example: CHSIZE A:DATAFILE 4096 - Change DATAFILE's size to 4096 bytes

FILLFILE *Create a file and fill it with a pattern.* This command creates a file and repeatedly fills it with a pattern. It is useful when you wish to create some files for experimenting with on an otherwise empty volume.
Example: Create and fill the file file.dat with the pattern "THIS IS A TEST" 1000 times.
FILLFILE FILE.DAT "THIS IS A TEST" 1000

FILLHUGEFILE *Create a file and fill it with a numeric pattern.* This command creates a file and fills it with a numeric pattern. You can specify gigabytes and bytes separately to create very large files. A metadata only method is also provided to create the file without actually writing the data blocks.

FILLHUGEFILE Filename DOMETADATAONLY buffersizebytes GIGABYTES BYTES

Create a file with length equal to (GIGABYTES*1073741824)+BYTES
Example: FILLHUGEFILE myfile.dat 0 32768 1 0
Writes a pattern in 32 K chunks into a file 1073741824 bytes long
Example: FILLHUGEFILE myfile.dat 1 131072 1 1000
Extends a file in 128 K increments to 1073742824 bytes long without actually

writing a pattern to the file.

Example: FILLHUGEFILE myfile.dat 0 1 0 1000
Write a pattern 1 bytes at a time into a file 1000 bytes long

---

READHUGEFILE *Read a file and optionally test it against a numeric pattern.* This command reads a file and compares it to a numeric pattern. A metadata only method is also provided to read the file without actually reading the data blocks.

READHUGEFILE Filename DOMETADATAONLY(1/0) DOCOMPARE(1/0) buffersizebytes

Example: READHUGEFILE myfile.dat 0 1 32768
Read a pattern in 32 K chunks into a file 1073741824 bytes long, check pattern
Example: READHUGEFILE myfile.dat 1 0 131072
Read a file in 128 K increments without transferring data or comparing
Example: READHUGEFILE myfile.dat 0 1 1
Read a pattern 1 byte at a time into a file and compare to a pattern

---

COPY *Copy a file to another location.* This command copies the source file to the destination.
Example: COPY A:FILE.DAT B:FILE.DAT

DIFF *Compare two files.* This command compares two files and prints whether or not they are the same.
Example: DIFF A:FILE1.DAT B:FILE1.DAT

FS (see the **fs** command reference for the extended shell)

CAT *Display contents of a file*. This command displays the contents of a file to the console.
Example: CAT A:\use\ASCII\budget.txt

Extended Command Shell Reference

Invoke the **ESHELL** command from the basic shell to access commands for extended IO, circular files, FAT64, asynchronous operations and access to ProPlus test procedures.

Command Reference

For help on commands that take arguments, type the command with no arguments.

The following commands are available.

- Drive and System Operations
  - **clear**
  - **remount**
  - **async**
  - **complete**
  - **setdrive**
  - **diskflush**
- Failsafe Operations
  - **fs enter D:** - Begin journalling
  - **fs abort D:** - Abort the current mount without flushing
  - **fs exit D:** - Flush Journal file and synchronize FAT volume, stop journalling
  - **fs jcommit D:** - Flush Journal file but don't synchronize. Stop journalling and abort. Leaves a restorable Journal file.
  - **fs jflush D:** - Flush Journal file but don't synchronize and continue journalling. Creates a flushed frame.
  - **fs commit D:** - Flush Journal file and synchronize FAT volume
  - **fs info D:** - Display information about current Failsafe Session
  - **fs clear D:** - Reset Journal IO statistics
  - **fs restore D:** - Restore the volume from the current Journal file
- File Operations -
  - **open** - Open a file with extended options
  - **close** **-** Close a linear or circular file
  - **flush** **-** Flush a linear or circular file
  - **sethint** - Force preallocation or set cluster allocation hint
  - **write** **-** Write or simulate write to a linear or circular file
  - **read** **-** Read or simulate read of a linear or circular file
  - **seek** **-** Demonstrate seek performance
  - **fstat** **-** Print extended file stats of an open linear or circular file
  - **chsize** **-** Expand or contract a linear or circular file.

- Circular File Operations
  - **copen** - Open a circular file.

    - **extract** - Extract a section of a circular file to a linear file
- Miscelaneous Operations
    - **delete** - Perform high speed asynchronous delete
    - **settime** - Change a directory entry's time and date fields.
    - **setsize** - Change a directory entry's size field.
    - **setcluster** - Change a directory entry's start cluster field.
    - **test**
        - **test efile D:** - Test Extendeded File API
        - **test extract D:** - Test **pc_efext_extract()**, swap and remove
        - **test cfile D:** - Test Circular files
        - **test async D:** - Test Async operation and Failsafe
        - **test transaction D**: - Test transaction files
        - **test failsafe D**: - Test Failsafe
    - **quit** **-** Return to the basic shell

All Basic shell commands are also available from the extended shell. Basic shell commands are all uppercase while extended shell commands are all lower case. To execute a Basic command from the extended shell simply type the basic upper case command with its arguments. For basic **HELP** type upper case "**HELP**".

The following pages contain usage instructions for the Individual command descriptions:

---

clear D:

D:	Drive id.

Reset all drive access statistics to zero for D:. After executing **clear**, the access statistics reported for the next operation will reflect only those accesses needed to complete the next operation.

---

remount D:

D:	Drive id.

Flush the drive, close it and then re-open it and rescan the FAT.

Note: If asynchronous mode is enabled (see **async***) the command is performed asynchronously and must be completed by the method assigned in the* **async** *command.*

---

async Y <bg|inline|manual> delay

Enable asynchronous operations and set the asynchronous completion policy.

bg – If **bg** is specified then asynchronous operations are completed in the background by a thread that is spawned for that purpose.

*Note: bg mode requires a separate task to operate. The subroutine named **spawn_async_continue()** in efishellrd.c is responsible for spawning a thread. This is supported only for the Linux and Windows targets. To uses **bg** mode in other operating environments, you must modify this function*

inline – If **inline** is specified then immediately after asynchronous calls are started the command shell repeatedly calls **pc_async_continue()** until all async processing completes.

manual – If **manual** is specified then the command **complete** must be called to manually invoke a loop that repeatedly calls **pc_async_continue()** until all async processing completes.

delay - milliseconds between calls to **pc_async_continue()**. This argument is required only when **bg** mode is selected.

---

async N

Turn off asynchronous mode and use synchronous versions operations.

---

complete

Call this command to complete pending asynchronous operations. This command must be called to complete asynchronous operations when the **async** command enabled asynchronous operations but specified **manual** mode.

---

setdrive D:

Select the default drive id for future **ESHELL** commands to operate on.

D:	Drive id.

---

diskflush D:

Flush a drive's directory and small file FAT caches. If asynchronous mode is enabled (see **async**) then loop and complete the delete asynchronously.

D:	Drive id.

fs command [N]

Execute a Failsafe related operation.

Where **command** is:

abort	Abort the current mount without flushing. This command may be used to abort an active Failsafe Journaling session or to abort the current mount, even if Journaling is not active.
exit	Flush the Journal file, synchronize the volume and stop journaling.
enter	Begin journaling.
jcommit	Flush the Journal file but do not synchronize the volume. Then stop journaling and abort. This command may be used to create a Journal file, that when restored will synchronize the volume with the application view immediately prior to when "**fs jcommit**" was invoked.
jflush	Flush the Journal file but do not synchronize the volume and continue journaling.
commit	Flush the Journal file, synchronize the volume and continue journaling.
info	Display information about current Failsafe Session or the current Failsafe file if journaling is not currently enabled
clear	Reset Journal IO statistics
restore	Restore the volume from the current Journal File

open filename [preallocsize (clusters)] [option option ..]

Open or re-open a file.

*Note: If asynchronous mode is enabled (see **async**) the command is performed asynchronously and must be completed by the method assigned in the **async** command.*

Filename	File name to open
preallocsize	Optional parameter, if specified write calls allocate this many clusters when they need to expand the file. This helps to reduce fragmentation. When the file is closed unused pre-allocated clusters are freed.
[options]	One or more of the following options may be used.
FIRST_FIT	Always allocate new clusters from the beginning of the FAT region instead of the default, which is to allocate from the next cluster after the last cluster in this file.
FORCE_FIRST	Force cluster allocation to allocate the first free cluster, even if that means clusters that span multiple clusters are broken up into multiple writes with seeks. Otherwise by default for writes that span multiple clusters, they are allocated in a contiguous group if possible, and if that is not possible, then they are broken up into multiple writes with seeks
CONTIGUOUS	For writes that span multiple clusters, allocate clusters in a contiguous group. If that is not possible, then fail.

KEEP_PREALLOC	If **preallocsize** is specified, when the file is closed, instead of releasing preallocated clusters, make them part of the file and set the file size to include them.
FILE_64	Open the file as a FAT64 file instead of a regular FAT file. *Note: This is only necessary upon the original open of a file, to indicate that the file is FAT64. Re-opens of these files are not required.*
LOAD_AS_NEEDED	Load cluster chains as they are accessed, not when the file is opened.
REMAP_FILE	The file being opened will be used as an argument to **pc_cfilio_extract()**. Cannot be written to with the write command.
TEMP_FILE	Tell Rtfs to delete the file when it is closed, this also has the effect of never actually writing the cluster chains to the FAT so the close happens much faster than a normal close
TRANSACTION_FILE	Open the file as a transaction file. *Note: Failsafe Journaling must be enabled or the open will fail.*
BUFFERED	Open the file in buffered mode
TRUNCATE	If it is a file reopen, truncate the file. *Note: this option is invalid if in asynchronous mode.*

When **open** succeeds it prints a file index number that may be used as an argument to the other commands that operate on open files

---

close fileindex

Flush a file's directory entry and cluster chain to disk. And release the file.

Note: If asynchronous mode is enabled (see **async***) the command is performed asynchronously and must be completed by the method assigned in the* **async** *command.*

Fileindex	The file index printed by the **open** or **copen** command

---

flush fileindex

Flush a file's directory entry and cluster chain to disk.

Note: If asynchronous mode is enabled (see **async** *) the command is performed asynchronously and must be completed by the method assigned in the* **async** *command.*

Fileindex	The file index printed by the **open** or **copen** command

---

sethint fileindex clusternumber nclusters

Pre-allocate clusters or set a hint for where the next clusters for the file should be allocated from.

Note: After the file is written to, **SHOWFILEEXTENTS** *can be used to verify that the specified clusters were used.*

Fileindex	The file index printed by the **open** or **copen** command
clusternumber	Hint of where to allocate next clusters from
nclusters	If >0 pre-allocate this many clusters

---

write fileindex resetfp xferdata totalsize writesize

Write bytes to a file.

fileindex	The file index printed by the **open** or **copen** command
resetfp	'y' or 'Y' to seek to the beginning of the file before writing 'n' or 'N' to append from the current file pointer
xferdata	'y' or 'Y' to transfer data to the file (the contents of the data is just random data returned from a **malloc()**. 'n' or 'N' to pass a NULL pointer to the write call. When a NULL pointer is passed the routine behaves exactly the same except that data is not written to the data blocks of the file.
totalsize	Total number of bytes to write to the file. The argument is in (GB,KB,B) format[i].
writesize	Number of bytes to write per write call. The write command will call **pc_efilio_write()** or **pc_cfilio_write()** as many times as needed until **totalsize** bytes are written. The argument is in (GB,KB,B) format[*].

*Note: If **xferdata** is 'Y' or 'y' then the write command calls **malloc()** to allocate a buffer **writesize** bytes long to use for data (the buffer is freed when the command finishes). So very large **writesize** values will result in very large **malloc()**s.*
*If **xferdata** is 'N' or 'n' then no buffer is allocated so gigabyte values and more are acceptable.*

---

read fileindex resetfp xferdata totalsize readsize

Read bytes from a file.

fileindex	The file index printed by the **open** or **copen** command
resetfp	'y' or 'Y' to seek to the beginning of the file before reading. 'n' or 'N' to read from the current file pointer
xferdata	'y' or 'Y' to transfer data from the file 'n' or 'N' to pass a NULL pointer to the read call. When a NULL pointer is passed the routine behaves exactly the same except that the data is not read from the data blocks of the file.
totalsize	Total number of bytes to read from the file. The argument is in (GB,KB,B) format[*].

readsize	Number of bytes to read per call. The **read** command will call **pc_efilio_read()** or **pc_cfilio_read()** as many times as needed until **totalsize** bytes are read. The argument is in (GB,KB,B) format*.

*Note: if **xferdata** is 'Y' or 'y' then the read command call **malloc()** to allocate a buffer **readsize** bytes long to use for data (the buffer is freed when the command finishes). So very large **readsize** values will result in very large **malloc()**s.*
*If **xferdata** is 'N' or 'n' then no buffer is allocated so gigabyte values and more are acceptable.*

---

seek fileindex nseeks doread(Y/N) doxfer(Y/N)

Perform a specified number of seeks of random lengths on a file.

fileindex	The file index printed by the **open** and **copen** command
nseeks	Number of seeks to perform on the file. The argument is in (GB,KB,B) format*.
doread	'y' or 'Y' to perform a one block read operation after every seek. 'n' or 'N' to just perform the seeks.
doxfer	If **doread** is 'y' or 'Y' then this instructs the command whether it should perform data transfers during the read. If **doxfer** is 'y' or 'Y' this will mean the underlying routine will be passed a valid pointer. If **doxfer** is 'n' or 'N' then a NULL pointer is passed to the read call. When a NULL pointer is passed to the routine, it behaves exactly the same except that data is not read from the data blocks of the file.

*Note: If **doread** is 'y' and **doxfer** is 'y' then for each seek performed by this command the underlying device is forced to read a single block, which forces a seek on the device itself.*

Example:

Perform 100000 seeks on a file

```
seek 0 1000000 y n
```

Perform 100000 seeks on a file and do call read but do not force a call to the device driver.

```
seek 0 1000000 y n
```

Perform 1000 seeks on a file and do call read and do force a call to the device driver(a head seek).

```
seek 0 1000 y y
```

fstat fileindex

Print stat information on an open file.

fileindex	The file index printed by the **open** or **copen** command.

chsize fileindex newsize

Expand or contract an open file.

fileindex	The file index printed by the **open** or **copen** command
newsize	New size of the file in bytes. The argument is in (GB,KB,B) format*.

copen filename wrap point preallocsize [option option ..]

Open a circular file.

filename	File name to open
Wrap point	Required parameter. The byte offset in the file for when the file pointer completes it's read and write operations, and seeks back to this offset. The argument is in (GB,KB,B) format*.
preallocsize	Optional parameter. If specified write calls allocate this many clusters when they need to expand the file. This helps reduce fragmentation. When the file is closed unused pre-allocated clusters are freed. **Do not use, currently not supported.**
[options]	One or more of the following options may be used.
FILE_64	Open the file as a FAT64 file instead of a regular FAT file.
CONTIGUOUS	For writes that span multiple clusters, allocate clusters in a contiguous group. If that is not possible, then fail.
TEMP_FILE	Tell Rtfs to delete the file when it is closed, this also has the effect of never actually writing the cluster chains to the FAT so the close happens much faster than a normal close
BUFFERED	Open the file in buffered mode

Do not use other options

When **copen** succeeds it prints a file index number that may be use as an argument to the other commands that operate on open files.

extract circfile linfile offset length

Extract a portion of a circular file to a linear extract file.

circfile	A file index that was printed by the command **copen**
linfile	A file index that was printed by the command open, with **REMAP_FILE** specified.
offset	The stream offset in the circular file to extract from. The argument is in

	(GB,KB,B) format*.
length	The length in bytes to extract. The argument is in (GB,KB,B) format*.

delete filename

Delete a file from the disk.

Note: If asynchronous mode is enabled (see **async***) the command is performed asynchronously and must be completed by the method assigned in the* **async** *command.*

Note: Asynchronous delete takes advantage of user buffering and performs faster than synchronous delete.

filename	Name of file to delete

settime filename

Changes the last access time and date of a file. Rather than prompt for time input, this command demonstrates changing the file's timestamp by changing the time to: April, 1, 2007. 11:32:42 AM

Note: This function is designed solely for the purpose of demonstrating the capabilities of the API.

filename	Name of file to change

setsize filename newsize

Manually change the size field in a directory entry. This edits the directory entry directly and does not free or add clusters from the file chain. It can be used to create files with incorrect sizes according to check disk.

Filename	Name of file to change
newsize	New file size in bytes

setcluster filename newcluster

Manually change the first cluster field in a directory entry. This edits the directory entry directly and does not free the current file chain. It can be used to create lost chains and to move cluster chains from one file to another.

Filename	Name of file to change
newcluster	New value for first cluster in file

test testname D:

Run an RtfsProPlus regression test.

Note: Rtfs Pro Plus tests are not available in release 6.1a, they use internal obsolete methods to reconfigure disk parameters on the the fly and have not been upgraded to support the disk configuration method introduced in this release.

Note: All tests may not function properly unless the following conditions are met.

Constants in RTFSCONF.H	Necessary Value
INCLUDE_DEBUG_TEST_CODE	1
INCLUDE_DEBUG_RUNTIME_STATS	1
MAX_SEGMENTS_64	at least 8

Before the tests are executed the volume should be freshly formatted.

D:	Drive id.	
testname	The test to execute:	
	efile	Test efilio and diskio interface. Test
	extract	**pc_efext_extract()**, **swap** and **remove**
	cfile	Test cfilio interface
	async	Test asynchronous operations and aspects of Failsafe
	transaction	Test transaction file support
	failsafe	Test Failsafe operations

quit

Exit the extended shell and return to the basic shell.