The Non-Casual Reference to

PoSRIC

\ˈpoż-ˈrik

an acronym for the

Portable **S**cripted **R**iPorFS **I**nterface in **C**

By Charles Thompson

Table Of Contents

Introduction

RiPorFS was created with the idea of a truly limitless and portable file system. To make that idea a reality, **PoSRIC** has been created so you can access such filesystems on virtually any computer with a command line and a standard C library.

PoSRIC was originally made to be a FAT substitute, but personally I have found more use in it as a long-term archiver, due to the fact that the tool is not OS- or architecture-dependent. This allows archives to be made of theoretically any size the host computer permits, and be able to access the data on much older and newer hardware.

Building

Since one of the main goals of **PoSRIC** is portability there are many, many ways to compile it's source into a usable executable. This section will cover some of the most common scenarios where you will be compiling the source code.

DOS

Compiling under DOS is really simple. Install Turbo C onto your computer and run the command `tc` inside the root **PoSRIC** directory. Press O+R+Enter, select the config file that corresponds to the Turbo C version, and press F9 to compile the program.

Linux

Linux compilation is also really easy. Make sure you have a GCC-like compiler installed along with make and run `make` inside the root **PoSRIC** directory. You can find the binary inside the local bin directory, and you can install **PoSRIC** by running `make install`.

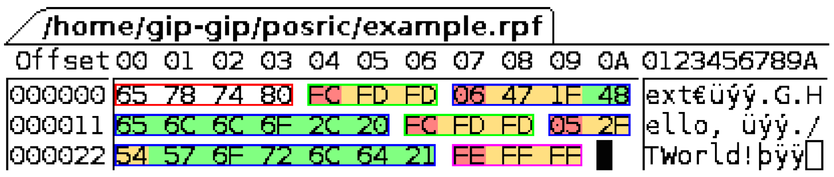
Commands

Command	Description
`#:foo is blue;`	Make a comment about foo being blue
`exit:;`	Exit posric with no errors
`giveUp:foo;`	Print “foo” and exit if the last command ended in error
`echo:foo;`	Print “foo” to the screen
`use:foo.rpf;`	Use the archive “foo.rpf”
`useName:foo`	Tells **PoSRIC** to apply future file-based commands to the file “foo”
`tmp:tmp.rpf;`	Use the file “tmp.rpf” as a temporary file
`format:;`	Format the archive being used
`list:;`	Lists all the files inside of the archive being used
`getFd:foo.bar;`	Gets file data from the in-archive file being used and stores it into “foo.bar”
`addFn:foo;`	Adds the file name “foo” to the archive, and gives an error if the file's detected
`addFd:foo.bar;`	Adds data from the file “foo.bar” to the in-archive file being used
`rmFile:foo;`	Removes the file “foo” from the archive, and gives an error if it doesn't exist

RiPorFS vIIβ Specs

The **Ri**dged **Por**table **F**ile **S**ystem is a storage protocol based off of(as you guessed) ridges. Ridges are 8-bit unsigned integers paired with a fletcher16 checksum of the integer and(if it's a data ridge) the data that follows. The ridges have a dual purpose; to describe data, and to provide data. Description ridges have the 8th bit set, and describe the data ridges that follow. Data ridges encode blocks of data that are 1-128 bytes long, and are combined to be interpreted in relation to the preceding descriptor ridge .

RiPorFS Layout



```
/home/gip-gip/posric/example.rpf
Offset 00 01 02 03 04 05 06 07 08 09 0A 0123456789A
000000 65 78 74 80 FC FD FD 06 47 1F 48 ext€üýý.G.H
000011 65 6C 6C 6F 2C 20 FC FD FD 05 2F ello, üýý./
000022 54 57 6F 72 6C 64 21 FE FF FF    TWorld!þÿÿ
```

Red border	Green border	Blue border	Pink border	Red fill	Tan fill	Green fill
signature	filename ridge	data ridge	end of fs ridge	ridge #	fletcher16 checksum	data

RiPorFS Ridge #s

Hex Value	Name	Description
FF	NULL Data	Used when the following data can be ignored
FE	End of FS Ridge	Used to signify the end of the filesystem
FD	File Data Ridge	Used when filedata is being read
FC	File Name Ridge	Used when a filename is being read. Encoded UTF-8
FB	Directory Name Ridge	Used when sorting files into a directory (named with the data being read). Encoded UTF-8
FA	Directory End Ridge	Used to end a directory
F9	Time Of Creation	Used to tell the time of creation of the following file/directory names. Data is a variable-length little-endian integer representing the time of creation in UNIX time.
F8	Time Of Modification	Used to tell the time of the last modification of the following file/directory names. Data is a variable-length little-endian integer representing the time of the last modification in UNIX time.
F7	File Owner	The name of the owner of the following files/directories.

		Encoded in UTF-8
F6	Last Writer	The name of the last person who modified the following files/directories. Encoded in UTF-8
F5	Permissions	The UNIX permissions for the following files/directories
F4	File Type	The MIME type of the following files
80	XML Metadata	Misc. XML metadata