Harry's Allocation Table

Misc – Workgroup: 0x10c Standards Committee
RFC X

Harry Jeffery

April 11, 2012

This draft provides the specification for the HAT(Harry's Allocation Table) filesystem. It is intended to provide a clear reference for anyone wishing to implement this filesystem.

1 Introduction

Currently there is no standard filesystem designed for use in a system built around the DCPU-16 architecture. This RFC provides a filesystem designed specifically for use on DCPU-16 systems.

HAT is designed to be simple and easy to implement while still providing all the features required of a filesystem.

2 Design Summary

HAT has a two layered design. The lower layer consists only of sectors. The lower layer is abstracted from the higher layer using "strips". The higher layer consists of inodes, links and raw file data.

The lower layer deals with the management of sectors and provides access to virtual strips of disk space.

The higher layer stores inodes, links and file data in these strips. Each strip (constructed in the lower layer using one or more sectors) contains an inode followed by either links or file data.

Each inode represents either a file or a directory. If the inode represents a file the strip contains the inode followed immediately by all the file data. If the inode represents a directory the strip contains the inode followed by a list of links.

Each link binds a file name to a file/directory. Each directory may contain several links to more directories or files, resulting in a tree growing from the root directory.

3 Data Structures

This section defines the data structures that comprise the HAT filesystem.

All sizes are given in words, which are considered to be 16 bits long.

3.1 Lower Layer

Section
header
sector map
sector 0
sector 1
sector 2
…

3.1.1 header

This structure contains the header information for HAT.

The total size of the header is 16 words.

Size	Name
1	version
1	num_sectors
2	sector_map_start
2	sector_start
1	sector_size
1	sectors_used

version The version field is a magic number that identifies both that the filesystem in use is HAT and the version of HAT.

The value of this field must be 0x4001. This magic number identifies the filesystem as version one of HAT.

num_sectors This field contains the total number of sectors in the filesystem.

sector_map_start This field contains the address of the start of the sector map.

sector_start This field contains the address of the first sector on disk.

sector_size This field contains the size of each sector in the filesystem.

This must be a power of 2, such as 128, 256 or 512.

sectors_used This field contains the number of sectors that are currently in use.

3.1.2 sector map

This section is a bitmap representing which sectors are in use. It can be used to quickly locate free sectors for new files.

Size	Name
ceil(num_sectors/16)	bitmap

bitmap This field is a bitmap that represents all the sectors in the filesystem. Each bit of the bitmap represents whether an sector is in use. When an sector is in use the corresponding bit is set to 1. When an sector is free, the corresponding bit is set to 0.

Any spare bits at the end of the bitmap must be set to 1.

3.1.3 sector

The sector structure is used to store blocks of raw data. Each sector may point to another sector. The chain of sectors that forms is then provided to the higher layer in the form of a strip that masks the non-contiguous nature of the sectors.

Size	Name
1	next_sector_index
sector_size - 1	sector_data

next_sector_index This field contains the index of the sector that follows this one. When there is no such sector this field's value is set to 0x0000.

sector_data This field simply contains raw data.

The size of this field is determined by the sector_size field in the header of the filesystem.

3.2 Higher Layer

3.2.1 inode

The inode structure is used to store metadata about files.

Size	Name
1	type
2	content_size
1	link_count

type This field indicates what type of inode it is. If this field is set to 0 then the inode is not in use and represents nothing. If this field is set to 1 then the inode represents a directory. If this field is set to 2 then the inode represents a file.

content_size This field contains the amount of data stored with this inode in words.

link_count This field contains the number of links there are that point to the strip containing this inode.

3.2.2 link

Size	Name
1	strip_start_sector
15	file_name

strip_start_sector This field contains the index of the start sector of the strip the inode being linked to is stored in.

file_name This field contains the file name to be associated with the inode that is being linked. Only alphanumeric characters, periods(.) and underscores(_) are allowed in the filename. The maximum length of the filename is 15 characters, any unused characters at the end of the filename must be set to 0x0000.