h5jam - h5unjam

h5jam - h5unjam

h5jam: Adds user block to front of an HDF5 file, to create a new concatenated file

h5unjam: Splits user block and HDF5 file into two files: user block data and HDF5 data

Syntax:

```
h5jam -u user_block -i in_file.h5 [-o out_file.h5] [--clobber]
h5jam -h

h5unjam -i in_file.h5 [-u user_block | --delete] [-o out_file.h5]
h5unjam -h
```

Description:

`h5jam`:
`h5jam` concatenates a `user_block` file and an HDF5 file to create an HDF5 file with a user block.

If `out_file.h5` is given, a new file is created with the `user_block` followed by the contents of `in_file.h5`. In this case, `infile.h5` is unchanged.

If `out_file.h5` is not specified, the `user_block` is added to `in_file.h5`.

If `in_file.h5` already has a user block, the contents of `user_block` will be added to the end of the existing user block, and the file shifted to the next boundary. If `--clobber` is set, any existing user block will be overwritten.

A user block can contain either binary or text data.

The minimum size of a user block is 512 bytes. As needed, the user block can be any power of 2 greater than that: 1024 bytes, 2048 bytes, etc. The user block in the output file is padded so that the HDF5 header begins on the first appropriate boundary. For example, if only 8 bytes of data are inserted for the user block, the HDF5 header will be found at byte 512; if 1100 bytes of data are inserted for the user block, the HDF5 header will be found at byte 2048.
`h5unjam`:
`h5unjam` splits an HDF5 file, writing the user block to a file or to stdout and the HDF5 file to an HDF5 file with a header at byte zero (`0`, i.e., with no user block).

If `out_file.h5` is given, a new file is created with the contents of `in_file.h5` without the user block. In this case, `infile.h5`is unchanged.

If `out_file.h5` is not specified, the `user_block` is removed and `in_file.h5` is rewritten, starting at byte 0.

If `user_block` is set, the user block will be written to `user_block`. If `user_block` is not set, the user block, if any, will be written to stdout. If `--delete` is selected, the user block will not be written.

The last portion of a returned user block may contain padding or undefined data (see discussion below: “`h5jam` and `h5unjam`not necessarily transitive”). It is the user’s or the user application’s responsibility to handle this correctly.

Exit Status:

0	Succeeded.
> 0	An error occurred.

Example:

Create new file, `newfile.h5`, with the text in file `mytext.txt` as the user block for the HDF5 file `file.h5`.

```
h5jam -u mytext.txt -i file.h5 -o newfile.h5
```

Add text in file `mytext.txt` to front of HDF5 dataset, `file.h5`.

```
h5jam -u mytext.txt -i file.h5
```

Overwrite the user block, if any, in `file.h5` with the contents of `mytext.txt`.

```
h5jam -u mytext.txt -i file.h5 --clobber
```

For an HDF5 file, `with_ub.h5`, with a user block, extract the user block to `user_block.txt` and the HDF5 portion of the file to `wo_ub.h5`.

```
h5unjam -i with_ub.h5 -u user_block.txt -o wo_ub.h5
```

Caveats:

These tools copy all the data sequentially in the file(s) to new offsets. For a large file, this copy will take a long time.

The most efficient way to create a user block is to create the file with a user block (see `H5Pset_userblock`), and write the user block data into that space from a program.

The user block is completely opaque to the HDF5 library and to the `h5jam` and `h5unjam` tools. The user block is read or written in a single block as a string of bytes; it can contain text or any kind of binary data; and it is up to the user to know what the user block content means and how to process it.

When the user block is extracted, its entire contents are written as a single block of output, including any padding or uninitialized data.

This tool moves the HDF5 portion of the file through byte copies; i.e., it does not read or interpret the HDF5 objects.

`h5jam` and `h5unjam` not necessarily transitive:
Note that `h5jam` and `h5unjam` are not necessarily transitive operations. Any amount of data can be inserted into a user block, but an HDF5 user block itself has specific size requirements. The minimum size is 512 bytes; beyond that, the user block can be 512 bytes times any positive power of 2. That is, a user block’s size will be one of the following: 512 bytes, 1024 bytes, 2048 bytes, 4096 bytes, et cetera.

If `h5jam` is used to insert a 700 byte file into the user block, `h5jam` will create a user block of 1024 bytesa and insert the user’s file as the first 700 bytes of that block. The remaining 324 bytes will be undefined. If the remaining bytes must have a particular fill value, for instance, the user must modify the input file by padding it to exactly 1024 bytes with the required fill value before inserting it with `h5jam`.

When `h5unjam` is asked to return the above user block, it will be returned with the padding in the last 324 bytes if the user defined it or with undefined data in the last 324 bytes if the user took no action to insert the padding.

If the file must be cleaned up for use, it is the user’s or the user application’s responsibility.

If a community of users employs user block data that must be cleaned up after the use of `h5unjam`, the community should establish a protocol for that process so that every community member knows what is required. The community may prefer to create and provide a tool to perform standard cleanup. A simple protocol might be for a user community to declare that the first *N* bytes of the user block will always contain the length or size of the valid user block content, much as a Pascal string starts with the length of the string data. Also see the HDF5 source code for examples of examining or reading the user block without modifying the file in any way. The relevant source files are the test programs `tools/h5jam/tellub.c` and `tools/h5jam/getub.c`.

--- Last Modified: August 28, 2019 | 08:38 AM