Name

header - WFDB header file format

Description

For each database record, a header file specifies the names of the associated signal files and their attributes. Programs compiled with the WFDB library (*-lwfdb*) can read header files created by *newheader* (see **wfdb**(3)). Header files contain line- and field-oriented ASCII text. ASCII linefeed characters separate lines (which may not contain more than 255 characters each, including the linefeed), and spaces or tabs separate fields (except as noted below). Beginning with WFDB library version 6.1, an ASCII carriage return character may precede each linefeed. Fields not specifically designated below as optional must be present.

Header files contain at a minimum a *record line*, which specifies the record name, the number of segments, and the number of signals. Header files for ordinary records (those that contain one segment) also contain a *signal specification line* for each signal. Header files for multi-segment records (supported by WFDB library version 9.1 and later versions) contain a *segment specification line* for each segment; see the section on multi-segment records below for details.

Comment lines may appear anywhere in a header file. The first printing character in a comment line must be '#'. Comment lines that follow the last signal specification line are treated specially (see *Info strings*, below). All other comment lines are ignored by WFDB library functions that read header files.

Record line

The first non-empty, non-comment line is the *record line*. It contains information applicable to all signals in the record. Its fields are, from left to right:

record name
: A string of characters that identify the record. The record name may include letters, digits and underscores ('_') only.

number of segments [optional]
: This field, if present, is *not* separated by whitespace from the record name field; rather, it follows a '/', which serves as a field separator. If the field is present, it indicates that the record is a multi-segment record containing the specified number of segments, and that the header file contains segment specification lines rather than signal specification lines. The number of segments must be greater than zero. A value of 1 in this field is legal, though unlikely to be useful.

number of signals
: Note that this is not necessarily equal to the number of signal files, since two or more signals can share a signal file. This number must not be negative; a value of zero is legal, however.

sampling frequency (in samples per second per signal) [optional]
: This number can be expressed in any format legal for **scanf**(3) input of floating point numbers (thus '360', '360.', '360.0', and '3.6e2' are all legal and equivalent). The sampling frequency must be greater than zero; if it is missing, a value of 250 (**DEFREQ**, defined in <*wfdb/wfdb.h*>) is assumed.

counter frequency (in ticks per second) [optional]
: This field (a floating-point number, in the same format as the sampling frequency) can be present only if the sampling frequency is also present. It is *not* separated by whitespace from the sampling frequency field; rather, it follows a '/', which serves as a field separator. The sampling and counter frequencies are used by *strtim* to convert strings beginning with 'c' into sample intervals. Typically, the counter frequency may be derived from an analog tape counter, or from page numbers in a chart recording. If the counter frequency is absent or not positive, it is assumed to be equal to the sampling frequency. WFDB library versions 5.1 and earlier ignore the counter frequency field.

base counter value [optional]

This field can be present only if the counter frequency is also present. It is *not* separated by whitespace from the counter frequency field; rather, it is surrounded by parentheses, which delimit it. The base counter value is a floating-point number that specifies the counter value corresponding to sample 0. If absent, the base counter value is taken to be zero. WFDB library versions 5.1 and earlier ignore the base counter value field.

number of samples per signal [optional]
: This field can be present only if the sampling frequency is also present. If it is zero or missing, the number of samples is unspecified and checksum verification of the signals is disabled.

base time [optional]
: This field can be present only if the number of samples is also present. It gives the time of day that corresponds to the beginning of the record, in HH:MM:SS format (using a 24-hour clock; thus 13:05:00, or 13:5:0, represent 1:05 pm). If this field is absent, the time-conversion functions assume a value of 0:0:0, corresponding to midnight.

base date [optional]
: This field can be present only if the base time is also present. It contains the date that corresponds to the beginning of the record, in DD/MM/YYYY format (e.g., 25/4/1989 is 25 April 1989).

Signal specification lines

Each non-empty, non-comment line following the record line in a single-segment record contains specifications for one signal, beginning with signal 0. Header files must contain valid signal specification lines for at least as many signals as were indicated in the record line (the first non-empty, non-comment line in the file). Any extra signal specification lines are not read by WFDB library functions. From left to right in each line, the fields are:

file name
: The name of the file in which samples of the signal are kept. The environment variable **WFDB** (the database path) lists the directories in which signal files (as well as WFDB header and annotation files) are found; normally **WFDB** should include an initial empty component, so that signal files can be kept in any directory if they are designated by absolute path names in the header file. If the file name specifies that the signal file is to be found in a directory that is not already in **WFDB**, that directory is appended to the end of **WFDB** (by functions that read header files in WFDB library version 6.2 and later versions). Although the record name is usually part of the signal file name, this convention is not a requirement (see, e.g., examples 3, 4, and 5 below). Note that several signals can share the same file (i.e., they can belong to the same signal group); all entries for signals that share a given file must be consecutive, however. The file name '-' refers to the standard input or output. The sum of the lengths of the file name and description fields (see below) is limited to 80 characters.

format
: This field is an integer that specifies the storage format of the signal. All signals in a given group are stored in the same format. The most common formats are format 8 (eight-bit first differences) and format 16 (sixteen-bit amplitudes); see **signal**(5) (or <*wfdb/wfdb.h*>) for a list of other supported formats. The following three optional fields, if present, are bound to the format field (i.e., not separated from it by whitespace); they may be considered as format modifiers, since they further describe the encoding of samples within the signal file.

samples per frame [optional]
: If present, this field follows an 'x' that serves as a field separator. Normally, all signals in a given record are sampled at the (base) sampling frequency as specified in the record line; in this case, the number of samples per frame is 1 for all signals, and this field is conventionally omitted. If the signal was sampled at some integer multiple, *n*, of the base sampling frequency, however, each frame (set of samples returned by *getframe*) contains *n* samples of the signal, and the value specified in this field is also *n*. (Note that non-integer multiples of the base sampling frequency are not supported.) WFDB library versions 8.3 and earlier ignore this field if it is present, and cannot properly read signal files that contain more than one sample per signal per frame.

skew [optional]
: If present, this field follows a ':' that serves as a field separator. Ideally, within a given record, samples of different signals with the same sample number are simultaneous (within one sampling interval). If this is not the case (as, for example, when a multitrack analog tape recording is digitized and the azimuth of the playback head does not match that of the recording head), the skew between signals can

sometimes determined (for example, by locating recorded waveform features with known time relationships, such as calibration signals). If this has been done, the skew field may be inserted into the header file to indicate the (positive) number of samples of the signal that are considered to *precede* sample 0. These samples, if any, are included in the checksum, but cannot be returned by *getvec* or *getframe* (thus the checksum need not be changed if the skew field is inserted or modified). WFDB library versions 9.1 and earlier ignore this field if it is present; later versions correctly deskew signals in accordance with the contents of this field.

byte offset [optional]

If present, this field follows a '+' that serves as a field separator. Normally, signal files include only sample data. If a signal file includes a preamble, however, this field specifies the offset in bytes from the beginning of the signal file to sample 0 (i.e., the length of the preamble). Data within the preamble is not included in the signal checksum. Note that the byte offset must be the same for all signals within a given group (use the skew field to correct for intersignal skew). This feature is provided only to simplify the task of reading signal files not generated using the WFDB library; the WFDB library does not support any means of writing such files, and byte offsets must be inserted into header files manually. WFDB library versions 4.4 and earlier ignore byte offsets; these versions return any preamble data as samples.

ADC gain (ADC units per physical unit) [optional]

This field is a floating-point number that specifies the difference in sample values that would be observed if a step of one physical unit occurred in the original analog signal. For ECGs, the gain is usually roughly equal to the R-wave amplitude in a lead that is roughly parallel to the mean cardiac electrical axis. If the gain is zero or missing, this indicates that the signal amplitude is uncalibrated; in such cases, a value of 200 (**DEFGAIN**, defined in <*wfdb/wfdb.h*>) ADC units per physical unit may be assumed.

baseline (ADC units) [optional]

This field can be present only if the ADC gain is also present. It is *not* separated by whitespace from the ADC gain field; rather, it is surrounded by parentheses, which delimit it. The baseline is an integer that specifies the sample value corresponding to 0 physical units. If absent, the baseline is taken to be equal to the ADC zero. Note that the baseline need not be a value within the ADC range; for example, if the ADC input range corresponds to 200-300 degrees Kelvin, the baseline is the (extended precision) value that would map to 0 degrees Kelvin. WFDB library versions 5.0 and earlier ignore baseline fields.

units [optional]

This field can be present only if the ADC gain is also present. It follows the baseline field if that field is present, or the gain field if the baseline field is absent. It is not separated by whitespace from the previous field; rather, it follows a '/', which serves as a field separator. The units field is a character string without embedded whitespace that specifies the type of physical unit. If the units field is absent, the physical unit may be assumed to be one millivolt. WFDB library versions 4.7 and earlier ignore units fields.

ADC resolution (bits) [optional]

This field can be present only if the ADC gain is also present. It specifies the resolution of the analog-to-digital converter used to digitize the signal. Typical ADCs have resolutions between 8 and 16 bits. If this field is missing or zero, the default value is 12 bits for amplitude-format signals, or 10 bits for difference-format signals (unless a lower value is specified by the *format* field).

ADC zero [optional]

This field can be present only if the ADC resolution is also present. It is an integer that represents the amplitude (sample value) that would be observed if the analog signal present at the ADC inputs had a level that fell exactly in the middle of the input range of the ADC. For a bipolar ADC, this value is usually zero, but a unipolar (offset binary) ADC usually produces a non-zero value in the middle of its range. Together with the ADC resolution, the contents of this field can be used to determine the range of possible sample values. If this field is missing, a value of zero is assumed.

initial value [optional]

This field can be present only if the ADC zero is also present. It specifies the value of sample 0 in the signal, but is used only if the signal is stored in difference format. If this field is missing, a value equal to the ADC zero is assumed.

checksum [optional]

This field can be present only if the initial value is also present. It is a 16-bit signed checksum of all *samples* in the signal. (Thus the checksum is independent of the storage format.) If the entire record is read without skipping samples, and the header's record line specifies the correct number of samples per signal, this field is compared against a computed checksum to verify that the signal file has not been corrupted. A value of zero may be used as a field placeholder if the number of samples is unspecified.

block size [optional]
: This field can be present only if the checksum is present. This field is an integer and is usually zero. If the signal is stored in a file that must be read in blocks of a specific size, however, this field specifies the block size in bytes. (On UNIX systems, this is the case only for character special files, corresponding to certain tape and raw disk files. If necessary, the block size may be given as a negative number to indicate that the associated file lacks I/O driver support for **fseek**(3) operations.) All signals belonging to the same signal group have the same block size.

description [optional]
: This field can be present only if the block size is present. Any text between the block size field and the end of the line is taken to be a description of the signal. When creating new records, follow the style used to document the signals in existing header files. Unlike the other fields in the header file, the description may include embedded spaces; note that whitespace between the block size and description fields is not considered to be part of the description, however. If the description is missing, the WFDB library functions that read header files supply a description of the form 'record *rec*, signal *n*' (shortened to 'signal *n*' by many WFDB applications).

Info strings

Comment lines that follow the last signal specification line in a header file can be read and written by the WFDB library functions *getinfo* and *putinfo*; the contents of these lines (excluding the initial '#' comment character) are referred to as 'info strings'. There must be no whitespace preceding the initial '#' in any line that is to be recognized by *getinfo*.

Multi-segment records

Each non-empty, non-comment line following the record line in the top-level header file of a multi-segment record contains specifications for one segment, beginning with segment 0. (Info strings cannot be used in the top-level header file of a multi-segment record.) Top-level header files must contain valid segment specification lines for at least as many segments as were indicated in the record line. Any extra segment specification lines are not read by WFDB library functions.

A *segment* is simply an ordinary (single-segment) record, with its own header and signal files. By including segments in a multi-segment record, the signals within them can be read by WFDB applications as if they were continuous signals, beginning with those in segment 0 and continuing with those in segment 1, with no need for the applications to do anything special to move from one segment to another. The only restrictions are that segments cannot themselves contain other segments (they *must* be single-segment records), the sampling frequencies must not change from segment to segment, and the number of samples per signal must be defined for each segment in the record line of the segment's own header file.

Two types of multi-segment records are defined. In a *fixed-layout* record, the arrangement of signals is constant across all segments, and the signal gain, baseline, units, ADC resolution and zero, and description match for corresponding signals in all segments (these recommendations are not enforced by the WFDB library, but existing applications are likely to behave unpredictably if they are not followed). Note, however, that it is not necessary to use the same signal storage format in all segments, and significant space savings may be possible in some cases by selecting an optimal format for each segment. Each segment of a fixed-layout record is an ordinary record containing one or more samples.

In a *variable-layout* record, the arrangement of signals may vary, signals may be absent in some segments, and the gains and baselines may change between segments. A variable-layout record can be identified by the presence of a *layout segment*, which must be segment 0 and must have a length of 0 samples. The layout segment has no associated signal files; its header file specifies the desired arrangement of signals and their gains and baselines. Signal file names in a layout segment header are recorded as '~'. When read using WFDB library version 10.3.17 or later, the signals of a variable layout record are rearranged, shifted, and rescaled as needed in order to present the signals in the arrangement and with the gains and baselines specified in the layout segment header.

Segment specification lines

Each segment specification line contains the following fields, separated by whitespace:

record name
: A string of characters identifying the single-segment record that comprises the segment. As in the record line, the record name may include letters, digits, and underscores ('_') only.

number of samples per signal
: This number must match the number specified in the header file for the single-segment record that comprises the segment.

Variable-layout records may contain *null segments*, which can be identified if the record name given in the segment specification line is '~'. The number of samples per signal indicates the length of the null segment; when read, these samples have the value WFDB_INVALID_DATA (defined in <wfdb/wfdb.h>). Null segments do not have associated header or signal files.

Examples:

Example 1 (MIT DB record 100):

```
100 2 360 650000 0:0:0 0/0/0
100.dat 212 200 11 1024 995 -22131 0 MLII

100.dat 212 200 11 1024 1011 20052 0 V5

# 69 M 1085 1629 x1
# Aldomet, Inderal
```

This header specifies 2 signals each sampled at 360 Hz, each 650000 samples (slightly over 30 minutes) long. The starting time and date were not recorded; in the example, the defaults are shown, but they might be omitted without changing the meaning of the header file. Each signal is stored in 12-bit bit-packed format (2 samples per 3 bytes; see **signal**(5) for details), and one file contains both signals. Since the filename given (*100.dat*) does not include path information, WFDB library-based programs will find the signal file only if it is located in one of the directories specified by the **WFDB** environment variable. The gain for each signal was the (default) 200 ADC units per millivolt (the default physical unit), and the ADC had 11-bit resolution and an offset such that its output was 1024 ADC units given an input exactly in the middle of its range. The baseline is not given explicitly, but may be assumed to be equal to the ADC zero value of 1024. The first samples acquired had values of 995 and 1011 (i.e., both signals began slightly below 0 VDC). The checksums of the 650000 samples are -22131 and 20052, and I/O may be performed in blocks of any desired size (since the block size fields are zero). The signal descriptions specify which leads were used (MLII: modified lead II). Finally, the last two lines contain 'info strings'. (In this example, the first info string specifies the sex and age of the subject and data about the recording, and the second lists the subject's medications. The contents and format of info strings vary between databases; it is not wise to rely on the presence of specific data in info strings, since their use in header files is optional.)

Example 2 (AHA DB record 7001):

```
7001 2 250 525000
/db1/data0/d0.7001 8 100 10 0 -53 -1279 0 ECG signal 0
/db1/data1/d1.7001 8 100 10 0 -69 15626 0 ECG signal 1
```

This header illustrates how on-line AHA DB records were formerly kept at MIT. Note that the sampling frequency and ADC specifications differ from the previous example. In this example, each signal is kept in its own signal file, specified by its absolute pathname. As shown here, AHA DB records may be kept in 8-bit first difference format, but the sampling rate requires that the signals be scaled down (from 12-bit to 10-bit ADC resolution) to stay within the slew rate limits imposed by the format. Note that signal checksums (-1279 and 15626 in this example) are derived from the reconstructed sample values, and not from the first differences; thus they should not change if the signals are reformatted.

Example 3 (Local record 8l):

```
8l 16
data0 8
data1 8
...

data15 8
```

This example illustrates how relative pathnames can be used for user-created records. If *data** files in the proper format are created in any of the directories named by the **WFDB** environment variable, they become the signal files for record 8l.

Example 4 (Piped record 16x4):

```
# Piped record 16x4. Use this record to read or write 4 signals
# using the standard I/O.
16x4 4
- 16
- 16
- 16
- 16
```

This example illustrates several features not seen in the earlier examples. The special file name ‘-’ means that samples will be read from the standard input or written to the standard output when using this record. All four signals are associated with the same file. The signals are kept in 16-bit amplitude format. The example includes two comment lines, which are ignored by the WFDB library functions that read header files.

Example 5 ("ahatape" header file):

```
# Use this record on a UNIX system to read directly
# from a 9-track AHA DB distribution tape with
# 4096-byte blocks. The tape must be positioned
# to the beginning of the ECG data file before
# using this record.

ahatape 2 250
/dev/nrmt0 16 0 12 0 0 0 4096
/dev/nrmt0 16 0 12 0 0 0 4096
```

As in the previous example, both signals are associated with the same file; in this case, the file is */dev/nrmt0*, the non-rewinding raw 9-track tape drive (on some systems, the name of this device may differ). The block size must be specified in this case, since I/O to or from a raw device (character special file) is not buffered by the operating system and must be performed in the units appropriate to the device (in this case, the tape block size). AHA DB tapes written at 1600 bpi contain 4096 bytes per block (i.e., 1024 two-byte samples from each of the two signals).

Example 6 ("multi" header file):

```
multi/3 2 360 45000
100s 21600
null 1800
100s 21600
```

This header file is a sample of a multi-segment record. The first line contains the record name ("multi"), the number of segments (there are 3), the number of signals (2; this must be the same in each segment), the sampling frequency (360), and the total length of the record in sample intervals (45000; this must be the sum of the segment lengths).

The second line contains the record name ("100s") of the first segment of the record, and its length in sample intervals (21600). The third and fourth lines contain the record names and lengths of the remaining segments. The remaining lines are comments.

Note that a segment may appear more than once in a multi-segment record, as in this sample, and that storage formats may vary between segments (the second segment is a "null" record, containing format 0 "signals", and the others are written in format 8).

This record may be read by any WFDB application built using WFDB library version 9.1 or later; the application need not be aware that this is a multi-segment record. Earlier versions of the WFDB library do not support multi-segment records (or format 0 signals).

Old format

Versions 2.3 through 4.6 of the WFDB library included support for reading header files written in an obsolete format. This support has been removed from WFDB library version 5.0. Obsolete-format header files can be brought up-to-date using *revise* (in the *convert* directory of the WFDB software distribution).

See Also

annot(5) , **signal**(5) , **wfdbcal**(5)
WFDB Programmer's Guide

Author

George B. Moody (george@mit.edu)

Table of Contents

Up: WFDB Applications Guide
