NAME

ESPS Data File Header

SYNOPSIS

#include <esps/esps.h>

DESCRIPTION

All ESPS data files have a header that consists of two parts. The first part contains header items that are common to all ESPS data files. This part of the header should contain all information required by ESPS programs that can operate on any type of data file. After this section of the header comes the type specific part. This part of the header contains items are are appropriate for specific types of ESPS data files. They are declared in a union, so that the correct one can be referenced by ESPS programs, depending on the type of the file, as indicated by the type code in the header. Additional types may be added as required.

The file type specific header items are described in the manual pages for those file types. For example, to understand the header structure of the ESPS sampled data file, it is necessary to refer to this manual page and *SD*(5−ESPS).

In addition, there is support for *generic* header items that can be created at run-time. These are held in a data structure in the file header.

The header has the following layout as defined by *<esps/header.h>*

```
/*
 * Structure of a ESPS data file header
 */

struct header {

/*  fields common to all ESPS headers  */

   struct fixsiz {
   short        type;                   /* file type */
   int          check;                  /* check field  */
   char         date[DATESIZE];         /* file creation date */
   char         hdvers[VERSIONSIZE];    /* header version */
   char         prog[PROGSIZE];         /* program name */
   char         vers[VERSIONSIZE];      /* prog version */
   char         progdate[DATESIZE];     /* prog compile date */
   long         ndrec;                  /* number of data records */
   short        tag;                    /* YES if data has tag */
   long         ndouble;                /* number of doubles */
   long         nfloat;                 /* number of floats */
   long         nlong;                  /* number of longs */
   long         nshort;                 /* number of shorts */
   long         nchar;                  /* number of chars */
   long         fixsiz;                 /* fixed header size */
   long         hsize;                  /* total header size */
   char         user[USERSIZ];          /* user that created file */
   short        edr;                    /* YES if file in EDR, NO for NATIVE */
   short        machine_code;           /* machine that produced file */
   short        spares[NSPARES];        /* spares */
   } common;

   struct varsize {
   char         *source[MAX_SOURCES];   /* pointers to src
                                           file names */
   struct header    *srchead[MAX_SOURCES]; /* pointers to src
```

```
                                        headers */
    char            *typtxt;            /* text field */
    char            *comment;           /* comment field */
    char            *refer;             /* reference file for tags */
    struct header   *refhd;             /* pointer to special reference header */
    short           nnames;             /* number of source file names */
    short           nheads;             /* number of source file
                                        headers */
    struct gen_hd   *gentab[GENTABSIZ]; /* generic item
                                        symbol table */
    short           ngen;               /* number of generic items */
    struct header   *refhd;             /* reference header */
    char            *current_path;        (** directory path */
    } variable;

/* Type specific portion of the header */

    union {
            struct sd_header *sd;       /* sampled data files */
            struct pit_header *pit;     /* pitch files */
            struct ana_header *ana;     /* analysis files */
            struct spec_header *spec;   /* spectral files */
            struct ros_header *ros;     /* rosetta speech frame files */
            struct filt_header *filt;   /* filter files */
            struct scbk_header *scbk;   /* scaler codebook file */
            struct fea_header *fea;     /* feature file */
    } hd;
};
```

Z domain functions are represented by the following structure;

```
struct zfunc {
  short         nsiz;     /* length of numerator polynomial */
  short         dsiz;     /* length of denominator polynomial */
  float         *zeros;   /* pointer to numerator polynomial */
  float         *poles;   /* pointer to denominator polynomial */
};
```

The gentab symbol table is a hash table of pointers to the following structure. Conflicts in the hash function are resolved by chaining nodes together.

```
struct gen_hd {
  char *name;           /* symbol name */
  unsigned int size;    /* size of item */
  short type;           /* type of symbol */
  char *d_ptr;          /* pointer to data */
  char **codes;         /* codes for CODED data type */
  struct geh_hd *next;  /* next in chain */
};
```

The following items are all in the fixed portion of the pitch file header.

type This field is set to the type code for the ESPS file. File type codes are defined in <*esps/ftypes.h*>, which is included automatically by means of <*esps/esps.h*>. Type code FT_FEA indicates an ESPS FEA file. Other types currently in use include FT_SD (old-style ESPS sampled data files), FT_SPEC (old-style ESPS spectral record files), FT_FILT (filter files), and FT_SCBK (scalar

codebook files). An ASCII array of strings (char *file_type[]) for these defined constants, suitable for use with *lin_search* (3−ESPS, is defined in the ESPS library. The definition is included automatically by means of <*esps/esps.h*>. This field should not be altered by a application programs. It is set by the header access programs.

check This field is used by the header access routines as a check that this is really a ESPS file header. It stores the special value HD_CHECK_VAL. This field should not be altered by application programs.

date The date/time that this data file was created. The format is that returned by *ctime*(3). This field should not be altered by application programs. It is set by the header access programs.

hdvers This field contains the version number of <*esps/header.h*> in use when this header was created. It is set by the header access programs and should not be altered by application programs. For embedded headers, this will be the version of the current header structure (relative to the binary of the program involved) and not necessarily the version when the original data file was created.

prog The name of the program that created this data file.

vers The version of the program that created this data file.

progdate
The date/time that the program that created this file was compiled. The format of the date/time is the same as *date*.

ndrec When a disk file is read by *read_header*, this field is set to the number of data records in the file. If a pipe is being read, then this field is set to -1. Programs should not set this field, and they must not rely on the value of this field if the input is to be a pipe. Normally, ESPS programs should be written to work with pipes so ndrec should not be used. If it is used, the restriction that pipes cannot be used must be documented on the manual page. If the number of records must be known before processing and and you still want to accept data from a pipe, an alternative is to write the pipe data to a temporary file. Note that *psps* (1−ESPS) does display the value of ndrec; thus, when *psps* is run on a disk file, the number of records is displayed correctly.

tag This field is YES if the data records are preceded by a tag. Otherwise this field is equal to NO. If a tag is present, it is not counted in the length of data record computation. If this field is YES, *variable.refer* should name the file to which tags refer.

The type of the data in the file is given by the next five header items. If a data structure contains more than a single data type, then they always will occur in the following order by type; doubles, floats, longs, shorts, and chars. The following header items give the number of records by each type. A general purpose application program should be able to determine the record structure from this information. See *write_header*(3−ESPSu) and *set_sd_type*(3−ESPSu). The position tag field, if present, is not included here.

ndouble
The number of double precision data types in the data record.

nfloat The number of floating point data types.

nlong The number of long data types.

nshort The number of short data types.

nchar The number of char data types.

fixsiz Size of the fixed portion of the header in 32-bit words. This field should not be written into by application programs. This field is filled in by *write_header*(3−ESPSu) and should not be written into by application programs.

hsize Total header size in 32-bit words. May be used to locate the first data point. This field is filled in by *write_header*(3) and should not be written into by application programs.

user The name (first 8 characters) of the username (login name) when this ESPS file was created. This field is filled in by *write_header*(3−ESPSu) and should not be written into by application programs.

spares There are NSPARES short words available for future use.

source A pointer to a source file name stored as a null-terminated string. The order of the source files (if there is more than one) is significant and is assumed to be order that the files were processed. There may be up to MAX_SOURCES source file names. ost programs do not deal directly with this field, but use *add_source_file*(3−ESPSu).

srchead A pointer to another ESPS header. The order is assumed to correspond to the order of *source* as described above. Most programs do not deal directly with this field, but use *add_source_file*(3−ESPSu).

current_path
Path to the directory from which the program was executed. This field is filled in by *write_header*(3−ESPSu) and should not be written into by application programs.

typtxt Pointer to a NULL terminated string. The maximum length of this string is MAX_STRING characters. This field is intended to be used as a plot label or title. For a reasonably short sampled data file, this field will often contain the text of the sampled speech. See *savestring*(3−ESPSu).

comment
This is also a pointer to a NULL terminated string of maximum length MAX_STRING. This string is intended to be used for a running commentary as the file is taken through various processing stages. The NULL terminated string may consist of NEW_LINE terminated substrings. See *add_comment*(3−ESPSu) and *comment*(1−ESPS).

refer A NULL terminated string of the name of a reference file relevant to a tag that might be in the data record. See *savestring*(3−ESPSu). For SD files, *hd.sd−>src_sf* is the sampling frequency of this reference file (which typically is a distinguished source file). For non-SD files, the reference file.

refhd A pointer to a special ESPS header. Unlike the header pointers *srchead*, *refhd* is not intended for use in keeping histories. Rather, it is used to retain reference information when needed by particular programs. (For example, it is needed in FEA file processing to maintain information about field derivations − see *fea_stat*(1−ESPS).

nnames The number of source file names in *source*.

nheads The number of source file headers in *srchead*.

FILES

/usr/include/esps/header.h

SEE ALSO

FILT(5−ESPS), SD(5−ESPS), SCBK(5−ESPS), FEA(5−ESPS), read_header(3−ESPSu), write_header(3−ESPSu), savestring(3−ESPSu), add_comment(3−ESPSu), add_genhd(3−ESPSu), get_genhd(3−ESPSu), genhd_list(3−ESPSu), genhd_type(3−ESPSu), comment(1−ESPS)

FUTURE CHANGES

AUTHOR

Joe Buck, Alan Parker, John Shore