Next Up Previous Contents FITSIO Home
Next: 4.7 Utility Routines **Up:** 4 CFITSIO Routines
Previous: 4.5 Table I/O Routines **Contents**

4.6 Header Keyword I/O Routines

The following routines read and write header keywords in the current HDU.

```
int fits_get_hdrspace(fitsfile *fptr, int *keysexist, int
*morekeys,
                int *status)
```

Return the number of existing keywords (not counting the mandatory END keyword) and the amount of empty space currently available for more keywords. The `morekeys` parameter may be set to NULL if it's value is not needed.

```
int fits_read_record(fitsfile *fptr, int keynum, char *record,
int *status)
int fits_read_card(fitsfile *fptr, char *keyname, char
*record, int *status)
int fits_read_key(fitsfile *fptr, int datatype, char *keyname,
            void *value, char *comment, int *status)

int fits_find_nextkey(fitsfile *fptr, char **inclist, int ninc,
            char **exclist, int nexc, char *card, int
```

(+)

```
*status)

int fits_read_key_unit(fitsfile *fptr, char *keyname, char
*unit,
                int *status)
```

These routines all read a header record in the current HDU. The first routine reads keyword number `keynum` (where the first keyword is at position 1). This routine is most commonly used when sequentially reading every record in the header from beginning to end. The 2nd and 3rd routines read the named keyword and return either the whole record, or the keyword value and comment string. In each case any non-significant trailing blank characters in the strings are truncated.

Wild card characters (*, ?, and #) may be used when specifying the name of the keyword to be read, in which case the first matching keyword is returned.

The `datatype` parameter specifies the C datatype of the returned keyword value and can have one of the following symbolic constant values: `TSTRING`, `TLOGICAL` (== int), `TBYTE`, `TSHORT`, `TUSHORT`, `TINT`, `TUINT`, `TLONG`, `TULONG`, `TFLOAT`, `TDOUBLE`, `TCOMPLEX`, and `TDBLCOMPLEX`. Data type conversion will be performed for numeric values if the intrinsic FITS keyword value does not have the same datatype. The `comment` parameter may be set equal to NULL if the comment string is not needed.

The 4th routine provides an easy way to find all the keywords in the header that match one of the name templates in `inclist` and do not match any of the name templates in `exclist`. `ninc` and `nexc` are the number of template strings in `inclist` and `exclist`,

(+)

respectively. Wild cards (*, ?, and #) may be used in the templates to match multiple keywords. Each time this routine is called it returns the next matching 80-byte keyword record. It returns status = `KEY_NO_EXIST` if there are no more matches.

The 5th routine returns the keyword value units string, if any. The units are recorded at the beginning of the keyword comment field enclosed in square brackets.

```
int fits_write_key(fitsfile *fptr, int datatype, char
*keyname,
      void *value, char *comment, int *status)
int fits_update_key(fitsfile *fptr, int datatype, char
*keyname,
      void *value, char *comment, int *status)
int fits_write_record(fitsfile *fptr, char *card, int *status)

int fits_modify_comment(fitsfile *fptr, char *keyname,
char *comment,
      int *status)
int fits_write_key_unit(fitsfile *fptr, char *keyname, char
*unit,
      int *status)
```

Write or modify a keyword in the header of the current HDU. The first routine appends the new keyword to the end of the header, whereas the second routine will update the value and comment fields of the keyword if it already exists, otherwise it behaves like the first routine and appends the new keyword. Note that `value` gives the address to the value and not the value itself. The `datatype` parameter specifies the C datatype of the keyword value and may have any of the

(+)

values listed in the description of the keyword reading routines, above. A NULL may be entered for the comment parameter, in which case the keyword comment field will be unmodified or left blank.

The third routine is more primitive and simply writes the 80-character `card` record to the header. It is the programmer's responsibility in this case to ensure that the record conforms to all the FITS format requirements for a header record.

The fourth routine modifies the comment string in an existing keyword, and the last routine writes or updates the keyword units string for an existing keyword. (The units are recorded at the beginning of the keyword comment field enclosed in square brackets).

---

int fits_write_comment(fitsfile *fptr, char *comment, int *status)
int fits_write_history(fitsfile *fptr, char *history, int *status)
int fits_write_date(fitsfile *fptr, int *status)

Write a `COMMENT`, `HISTORY`, or `DATE` keyword to the current header. The `COMMENT` keyword is typically used to write a comment about the file or the data. The `HISTORY` keyword is typically used to provide information about the history of the processing procedures that have been applied to the data. The `comment` or `history` string will be continued over multiple keywords if it is more than 70 characters long.

The `DATE` keyword is used to record the date and time that the FITS file was created. Note that this file creation date is usually different from the date of the

(+)

observation which obtained the data in the FITS file. The `DATE` keyword value is a character string in 'yyyy-mm-ddThh:mm:ss' format. If a `DATE` keyword already exists in the header, then this routine will update the value with the current system date.

---

int fits_delete_record(fitsfile *fptr, int keynum, int *status)
int fits_delete_key(fitsfile *fptr, char *keyname, int *status)

Delete a keyword record. The first routine deletes a keyword at a specified position (the first keyword is at position 1, not 0), whereas the second routine deletes the named keyword.

---

int fits_copy_header(fitsfile *infptr, fitsfile *outfptr, int *status)

Copy all the header keywords from the current HDU associated with infptr to the current HDU associated with outfptr. If the current output HDU is not empty, then a new HDU will be appended to the output file. The output HDU will then have the identical structure as the input HDU, but will contain no data.

Next Up Previous Contents FITSIO Home
Next: 4.7 Utility Routines **Up:** 4 CFITSIO Routines
Previous: 4.5 Table I/O Routines **Contents**

(+)