NAME

ts_read, ts_read_raw, ts_read_mt, ts_read_raw_mt − read tslib touch samples

SYNOPSIS

#include <tslib.h>

int ts_read(struct tsdev ****dev*, struct ts_sample ****samp***, int** *nr***);**

int ts_read_raw(struct tsdev ****dev*, struct ts_sample ****samp***, int** *nr***);**

int ts_read_mt(struct tsdev ****dev*, struct ts_sample_mt *****samp***, int** *slots***, int** *nr***);**

int ts_read_raw_mt(struct tsdev ****dev*, struct ts_sample_mt *****samp***, int** *slots***, int** *nr***);**

DESCRIPTION

ts_read() reads **nr** input samples with tslib's filters applied. **struct ts_sample** is define as follows:

```
struct ts_sample {
    int             x;
    int             y;
    unsigned int    pressure;
    struct timeval  tv;
};
```

ts_read_mt() reads **nr** * **slots** input samples with tslib's filters applied. **struct ts_sample_mt** is defined as follows:

```
struct ts_sample_mt {
    /* most recent ABS_MT_* event codes.
     * see linux/input.h for descriptions */
    int             x;
    int             y;
    unsigned int    pressure;
    int             slot;
    int             tracking_id;

    int             tool_type;
    int             tool_x;
    int             tool_y;
    unsigned int    touch_major;
    unsigned int    width_major;
    unsigned int    touch_minor;
    unsigned int    width_minor;
    int             orientation;
    int             distance;
    int             blob_id;

    struct timeval  tv;

    /* BTN_TOUCH state */
    short           pen_down;

    /* the TSLIB_MT_VALID bit is set in valid if this sample
     * contains new data;
         * valid is set to 0 otherwise */
    short           valid;
```

};

The user has to provide the amount of memory described in **nr** and **slots** to hold them.

ts_read_raw() and **ts_read_raw_mt**() do the same thing without tslib’s filters applied.

RETURN VALUE

The number of actually read samples is returned. Especially when opened in non-blocking mode, see **ts_setup()** , that can be less than requested in the call. On failure, a negative error number is returned.

EXAMPLE

The following program continuously reads tslib multitouch input samples and prints slot and position values to stdout as the touch screen is touched.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

#include <tslib.h>

#define READ_SAMPLES 1
#define MAX_SLOTS 5

int main(int argc, char **argv)
{
        struct tsdev *ts;
        struct ts_sample_mt **samp_mt = NULL;
        int i, j;
        int ret;

        ts = ts_setup(NULL, 0);
        if (!ts)
                return −1;

        samp_mt = malloc(READ_SAMPLES * sizeof(struct ts_sample_mt **));
        if (!samp_mt)
                return −1;

        for (i = 0; i < READ_SAMPLES; i++) {
                samp_mt[i] = calloc(MAX_SLOTS, sizeof(struct ts_sample_mt));
                if (!samp_mt[i])
                        return −1;
        }

        while(1) {
                ret = ts_read_mt(ts, samp_mt, MAX_SLOTS, READ_SAMPLES);
                for (i = 0; i < ret; i++) {
                        printf("sample nr %d0, i);
                        for (j = 0; i < MAX_SLOTS; j++) {
                                if (!(samp_mt[i][j].valid & TSLIB_MT_VALID))
                                        continue;

                                printf("slot %d: X:%d Y: %d0,
                                       samp_mt[i][j].slot,
                                samp_mt[i][j].x,
                                samp_mt[i][j].y);
                        }
                    }
                }
            }
```

SEE ALSO

ts_setup(3), **ts_config**(3), **ts_open**(3), **ts_close**(3), **ts.conf**(5)