Latency Tests Reference Manual

3.4.0

Generated by Doxygen 1.4.5

Fri Nov 18 07:38:54 2005

Contents

Chapter 1

Latency Tests Directory Hierarchy

1.1 Latency Tests Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 2

Latency Tests Data Structure Index

2.1 Latency Tests Data Structures

Here are the data structures with brief descriptions:

Chapter 3

Latency Tests File Index

3.1 Latency Tests File List

Here is a list of all files with brief descriptions:

Chapter 4

Latency Tests Directory Documentation

4.1 cfg/ Directory Reference

Files

- file **config.h**

4.2 lib/ Directory Reference

Files

- file **misc.c**
- file **misc.h**

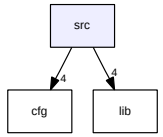
4.3 scripts/ Directory Reference

Files

- file **ascii2docbook.c**
- file **freqcurve.c**

  Analyze data to generate histogram values.

4.4 src/ Directory Reference



Files

- file **drone.c**

  Latency Drone - runs on the target, copies data from infifo to outfifo.

- file **latency-client.c**

  Latency Client - runs on the device to be tested.

- file **latency-server.c**

  Latency Server - runs on the desktop machine.

- file **serialblast.c**
- file **udpblast.c**

  Send a UDP packet.

Chapter 5

Latency Tests Data Structure Documentation

5.1 yy_buffer_state Struct Reference

Data Fields

- FILE * **yy_input_file**
- char * **yy_ch_buf**
- char * **yy_buf_pos**
- **yy_size_t yy_buf_size**
- int **yy_n_chars**
- int **yy_is_our_buffer**
- int **yy_is_interactive**
- int **yy_at_bol**
- int **yy_fill_buffer**
- int **yy_buffer_status**

5.1.1 Detailed Description

Definition at line 144 of file ascii2docbook.c.

5.1.2 Field Documentation

5.1.2.1 int yy_buffer_state::yy_at_bol

Definition at line 178 of file ascii2docbook.c.

5.1.2.2 char* yy_buffer_state::yy_buf_pos

Definition at line 149 of file ascii2docbook.c.

5.1.2.3 yy_size_t yy_buffer_state::yy_buf_size

Definition at line 154 of file ascii2docbook.c.

5.1.2.4 int yy_buffer_state::yy_buffer_status

Definition at line 185 of file ascii2docbook.c.

5.1.2.5 char* yy_buffer_state::yy_ch_buf

Definition at line 148 of file ascii2docbook.c.

5.1.2.6 int yy_buffer_state::yy_fill_buffer

Definition at line 183 of file ascii2docbook.c.

5.1.2.7 FILE* yy_buffer_state::yy_input_file

Definition at line 146 of file ascii2docbook.c.

5.1.2.8 int yy_buffer_state::yy_is_interactive

Definition at line 172 of file ascii2docbook.c.

5.1.2.9 int yy_buffer_state::yy_is_our_buffer

Definition at line 165 of file ascii2docbook.c.

5.1.2.10 int yy_buffer_state::yy_n_chars

Definition at line 159 of file ascii2docbook.c.

The documentation for this struct was generated from the following file:

- **ascii2docbook.c**

Chapter 6

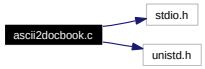
Latency Tests File Documentation

6.1 ascii2docbook.c File Reference

`#include <stdio.h>`

`#include <unistd.h>`

Include dependency graph for ascii2docbook.c:



Data Structures

- struct **yy_buffer_state**

Defines

- #define **FLEX_SCANNER**
- #define **YY_FLEX_MAJOR_VERSION** 2
- #define **YY_FLEX_MINOR_VERSION** 5
- #define **yyconst**
- #define **YY_PROTO**(proto) ()
- #define **YY_NULL** 0
- #define **YY_SC_TO_UI**(c) ((unsigned int) (unsigned char) c)
- #define **BEGIN** **yy_start** = 1 + 2 *
- #define **YY_START** ((**yy_start** - 1) / 2)
- #define **YYSTATE** YY_START
- #define **YY_STATE_EOF**(state) (YY_END_OF_BUFFER + state + 1)
- #define **YY_NEW_FILE** yyrestart(**yyin**)
- #define **YY_END_OF_BUFFER_CHAR** 0
- #define **YY_BUF_SIZE** 16384
- #define **EOB_ACT_CONTINUE_SCAN** 0
- #define **EOB_ACT_END_OF_FILE** 1

- #define **EOB_ACT_LAST_MATCH** 2
- #define **yyless**(n)
- #define **unput**(c) yyunput(c, yytext_ptr)
- #define **YY_BUFFER_NEW** 0
- #define **YY_BUFFER_NORMAL** 1
- #define **YY_BUFFER_EOF_PENDING** 2
- #define **YY_CURRENT_BUFFER** **yy_current_buffer**
- #define **YY_FLUSH_BUFFER** yy_flush_buffer(**yy_current_buffer**)
- #define **yy_new_buffer** yy_create_buffer
- #define **yy_set_interactive**(is_interactive)
- #define **yy_set_bol**(at_bol)
- #define **YY_AT_BOL**() (**yy_current_buffer** → yy_at_bol)
- #define **yytext_ptr** **yytext**
- #define **YY_DO_BEFORE_ACTION**
- #define **YY_NUM_RULES** 4
- #define **YY_END_OF_BUFFER** 5
- #define **REJECT** reject_used_but_not_detected
- #define **yymore**() yymore_used_but_not_detected
- #define **YY_MORE_ADJ** 0
- #define **YY_RESTORE_YY_MORE_OFFSET**
- #define **INITIAL** 0
- #define **YY_NO_PUSH_STATE** 1
- #define **YY_NO_POP_STATE** 1
- #define **YY_NO_TOP_STATE** 1
- #define **YY_READ_BUF_SIZE** 8192
- #define **ECHO** (void) fwrite(**yytext**, **yyleng**, 1, **yyout**)
- #define **YY_INPUT**(**buf**, result, max_size)
- #define **yyterminate**() return YY_NULL
- #define **YY_START_STACK_INCR** 25
- #define **YY_FATAL_ERROR**(msg) yy_fatal_error(msg)
- #define **YY_DECL** int yylex YY_PROTO((void))
- #define **YY_BREAK** break;
- #define **YY_RULE_SETUP** YY_USER_ACTION
- #define **YY_EXIT_FAILURE** 2
- #define **yyless**(n)

Typedefs

- typedef **yy_buffer_state** * **YY_BUFFER_STATE**
- typedef int **yyleng**
- typedef FILE * **yyin** = (FILE *) 0
- typedef FILE * **yyout** = (FILE *) 0
- typedef unsigned int **yy_size_t**
- typedef unsigned char **YY_CHAR**
- typedef int **yy_state_type**

Functions

- void yyrestart **YY_PROTO** ((FILE *input_file))
- void yy_switch_to_buffer **YY_PROTO** ((**YY_BUFFER_STATE** new_buffer))
- void yy_load_buffer_state **YY_PROTO** ((void))
- **YY_BUFFER_STATE** yy_create_buffer **YY_PROTO** ((FILE *__file__, int __size__))
- void yy_delete_buffer **YY_PROTO** ((**YY_BUFFER_STATE** b))
- void yy_init_buffer **YY_PROTO** ((**YY_BUFFER_STATE** b, FILE *__file__))
- **YY_BUFFER_STATE** yy_scan_buffer **YY_PROTO** ((char *base, **yy_size_t** __size__))
- **YY_BUFFER_STATE** yy_scan_string **YY_PROTO** ((yyconst char *yy_str))
- **YY_BUFFER_STATE** yy_scan_bytes **YY_PROTO** ((yyconst char *bytes, int __len__))
- static void *yy_flex_alloc **YY_PROTO** ((**yy_size_t**))
- static void *yy_flex_realloc **YY_PROTO** ((void *, **yy_size_t**))
- static void yy_flex_free **YY_PROTO** ((void *))
- static **yy_state_type** yy_try_NUL_trans **YY_PROTO** ((**yy_state_type** current_state))
- static void yy_fatal_error **YY_PROTO** ((yyconst char msg[]))
- static void yyunput **YY_PROTO** ((int c, char *buf_ptr))

Variables

- static **YY_BUFFER_STATE yy_current_buffer** = 0
- static char **yy_hold_char**
- static int **yy_n_chars**
- static char * **yy_c_buf_p** = (char *) 0
- static int **yy_init** = 1
- static int **yy_start** = 0
- static int **yy_did_buffer_switch_on_eof**
- char * **yytext**
- static yyconst short int **yy_accept** [9]
- static yyconst int **yy_ec** [256]
- static yyconst int **yy_meta** [5]
- static yyconst short int **yy_base** [9]
- static yyconst short int **yy_def** [9]
- static yyconst short int **yy_nxt** [11]
- static yyconst short int **yy_chk** [11]
- static **yy_state_type yy_last_accepting_state**
- static char * **yy_last_accepting_cpos**
- register char * **yy_bp**
- int **size**
- FILE * **file**
- int **len**

6.1.1 Define Documentation

6.1.1.1 #define BEGIN yy_start = 1 + 2 *

Definition at line 79 of file ascii2docbook.c.

6.1.1.2 #define ECHO (void) fwrite(yytext, yyleng, 1, yyout)

Definition at line 440 of file ascii2docbook.c.

6.1.1.3 #define EOB_ACT_CONTINUE_SCAN 0

Definition at line 104 of file ascii2docbook.c.

6.1.1.4 #define EOB_ACT_END_OF_FILE 1

Definition at line 105 of file ascii2docbook.c.

6.1.1.5 #define EOB_ACT_LAST_MATCH 2

Definition at line 106 of file ascii2docbook.c.

6.1.1.6 #define FLEX_SCANNER

Definition at line 7 of file ascii2docbook.c.

6.1.1.7 #define INITIAL 0

Definition at line 360 of file ascii2docbook.c.

6.1.1.8 #define REJECT reject_used_but_not_detected

Definition at line 355 of file ascii2docbook.c.

6.1.1.9 #define unput(c) yyunput(c, yytext_ptr)

Definition at line 135 of file ascii2docbook.c.

6.1.1.10 #define YY_AT_BOL() (yy_current_buffer → yy_at_bol)

Definition at line 262 of file ascii2docbook.c.

6.1.1.11 #define YY_BREAK break;

Definition at line 499 of file ascii2docbook.c.

6.1.1.12 #define YY_BUF_SIZE 16384

Definition at line 97 of file ascii2docbook.c.

6.1.1.13 #define YY_BUFFER_EOF_PENDING 2

Definition at line 198 of file ascii2docbook.c.

6.1.1.14 #define YY_BUFFER_NEW 0

Definition at line 186 of file ascii2docbook.c.

6.1.1.15 #define YY_BUFFER_NORMAL 1

Definition at line 187 of file ascii2docbook.c.

6.1.1.16 #define YY_CURRENT_BUFFER yy_current_buffer

Definition at line 207 of file ascii2docbook.c.

6.1.1.17 #define YY_DECL int yylex YY_PROTO((void))

Definition at line 487 of file ascii2docbook.c.

6.1.1.18 #define YY_DO_BEFORE_ACTION

Value:

```
yytext_ptr = yy_bp; \
        yyleng = (int) (yy_cp - yy_bp); \
        yy_hold_char = *yy_cp; \
        *yy_cp = '\0'; \
        yy_c_buf_p = yy_cp;
```

Definition at line 278 of file ascii2docbook.c.

6.1.1.19 #define YY_END_OF_BUFFER 5

Definition at line 286 of file ascii2docbook.c.

6.1.1.20 #define YY_END_OF_BUFFER_CHAR 0

Definition at line 94 of file ascii2docbook.c.

6.1.1.21 #define YY_EXIT_FAILURE 2

6.1.1.22 #define YY_FATAL_ERROR(msg) yy_fatal_error(msg)

Definition at line 480 of file ascii2docbook.c.

6.1.1.23 #define YY_FLEX_MAJOR_VERSION 2

Definition at line 8 of file ascii2docbook.c.

6.1.1.24 #define YY_FLEX_MINOR_VERSION 5

Definition at line 9 of file ascii2docbook.c.

6.1.1.25 #define YY_FLUSH_BUFFER yy_flush_buffer(yy_current_buffer)

Definition at line 236 of file ascii2docbook.c.

6.1.1.26 #define YY_INPUT(buf, result, max_size)

Value:

```
if ( yy_current_buffer->yy_is_interactive ) \
                { \
                int c = '*', n; \
                for ( n = 0; n < max_size && \
                             (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
                        buf[n] = (char) c; \
                if ( c == '\n' ) \
                        buf[n++] = (char) c; \
                if ( c == EOF && ferror( yyin ) ) \
                        YY_FATAL_ERROR( "input in flex scanner failed" ); \
                result = n; \
                } \
        else if ( ((result = fread( buf, 1, max_size, yyin )) == 0) \
                  && ferror( yyin ) ) \
                YY_FATAL_ERROR( "input in flex scanner failed" );
```

Definition at line 447 of file ascii2docbook.c.

6.1.1.27 #define YY_MORE_ADJ 0

Definition at line 357 of file ascii2docbook.c.

6.1.1.28 #define yy_new_buffer yy_create_buffer

Definition at line 246 of file ascii2docbook.c.

6.1.1.29 #define YY_NEW_FILE yyrestart(yyin)

Definition at line 92 of file ascii2docbook.c.

6.1.1.30 #define YY_NO_POP_STATE 1

Definition at line 410 of file ascii2docbook.c.

6.1.1.31 #define YY_NO_PUSH_STATE 1

Definition at line 409 of file ascii2docbook.c.

6.1.1.32 #define YY_NO_TOP_STATE 1

Definition at line 411 of file ascii2docbook.c.

6.1.1.33 #define YY_NULL 0

Definition at line 66 of file ascii2docbook.c.

6.1.1.34 #define YY_NUM_RULES 4

Definition at line 285 of file ascii2docbook.c.

6.1.1.35 #define YY_PROTO(proto) ()

Definition at line 62 of file ascii2docbook.c.

6.1.1.36 #define YY_READ_BUF_SIZE 8192

Definition at line 431 of file ascii2docbook.c.

6.1.1.37 #define YY_RESTORE_YY_MORE_OFFSET

Definition at line 358 of file ascii2docbook.c.

6.1.1.38 #define YY_RULE_SETUP YY_USER_ACTION

Definition at line 502 of file ascii2docbook.c.

6.1.1.39 #define YY_SC_TO_UI(c) ((unsigned int) (unsigned char) c)

Definition at line 73 of file ascii2docbook.c.

6.1.1.40 #define yy_set_bol(at_bol)

Value:

```
{ \
        if ( ! yy_current_buffer ) \
               yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
        yy_current_buffer->yy_at_bol = at_bol; \
        }
```

Definition at line 255 of file ascii2docbook.c.

6.1.1.41 #define yy_set_interactive(is_interactive)

Value:

```
{ \
        if ( ! yy_current_buffer ) \
               yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
        yy_current_buffer->yy_is_interactive = is_interactive; \
        }
```

Definition at line 248 of file ascii2docbook.c.

6.1.1.42 #define YY_START ((yy_start - 1) / 2)

Definition at line 85 of file ascii2docbook.c.

6.1.1.43 #define YY_START_STACK_INCR 25

Definition at line 475 of file ascii2docbook.c.

6.1.1.44 #define YY_STATE_EOF(state) (YY_END_OF_BUFFER + state + 1)

Definition at line 89 of file ascii2docbook.c.

6.1.1.45 #define yyconst

Definition at line 55 of file ascii2docbook.c.

6.1.1.46 #define yyless(n)

Value:

```
do \
                { \
                /* Undo effects of setting up yytext. */ \
                yytext[yyleng] = yy_hold_char; \
                yy_c_buf_p = yytext + n; \
                yy_hold_char = *yy_c_buf_p; \
                *yy_c_buf_p = '\0'; \
                yyleng = n; \
                } \
        while ( 0 )
```

Definition at line 124 of file ascii2docbook.c.

6.1.1.47 #define yyless(n)

Value:

```
do \
                { \
                /* Undo effects of setting up yytext. */ \
                *yy_cp = yy_hold_char; \
                YY_RESTORE_YY_MORE_OFFSET \
                yy_c_buf_p = yy_cp = yy_bp + n - YY_MORE_ADJ; \
                YY_DO_BEFORE_ACTION; /* set up yytext again */ \
                } \
        while ( 0 )
```

Definition at line 124 of file ascii2docbook.c.

6.1.1.48 #define yymore() yymore_used_but_not_detected

Definition at line 356 of file ascii2docbook.c.

6.1.1.49 #define YYSTATE YY_START

Definition at line 86 of file ascii2docbook.c.

6.1.1.50 #define yyterminate() return YY_NULL

Definition at line 470 of file ascii2docbook.c.

6.1.1.51 #define yytext_ptr yytext

Definition at line 268 of file ascii2docbook.c.

6.1.2 Typedef Documentation

6.1.2.1 typedef struct yy_buffer_state* YY_BUFFER_STATE

Definition at line 99 of file ascii2docbook.c.

6.1.2.2 typedef unsigned char YY_CHAR

Definition at line 264 of file ascii2docbook.c.

6.1.2.3 typedef unsigned int yy_size_t

Definition at line 141 of file ascii2docbook.c.

6.1.2.4 typedef int yy_state_type

Definition at line 266 of file ascii2docbook.c.

6.1.2.5 FILE * yyin = (FILE *) 0

Definition at line 265 of file ascii2docbook.c.

6.1.2.6 int yyleng

Definition at line 216 of file ascii2docbook.c.

6.1.2.7 FILE * yyout = (FILE *) 0

Definition at line 265 of file ascii2docbook.c.

6.1.3 Function Documentation

6.1.3.1 static void yyunput YY_PROTO ((int c, char *buf_ptr)) [static]

6.1.3.2 static void yy_fatal_error YY_PROTO ((yyconst char msg[])) [static]

6.1.3.3 static yy_state_type yy_try_NUL_trans YY_PROTO ((yy_state_type current_state)) [static]

6.1.3.4 static void yy_flex_free YY_PROTO ((void *)) [static]

6.1.3.5 static void* yy_flex_realloc YY_PROTO ((void *, yy_size_t)) [static]

6.1.3.6 static void* yy_flex_alloc YY_PROTO ((yy_size_t)) [static]

6.1.3.7 YY_BUFFER_STATE yy_scan_bytes YY_PROTO ((yyconst char *bytes, int len))

6.1.3.8 YY_BUFFER_STATE yy_scan_string YY_PROTO ((yyconst char *yy_str))

6.1.3.9 YY_BUFFER_STATE yy_scan_buffer YY_PROTO ((char *base, yy_size_t size))

6.1.3.10 void yy_init_buffer YY_PROTO ((YY_BUFFER_STATE b, FILE *file))

6.1.3.11 void yy_flush_buffer YY_PROTO ((YY_BUFFER_STATE b))

6.1.3.12 YY_BUFFER_STATE yy_create_buffer YY_PROTO ((FILE *file, int size))

6.1.3.13 static int input YY_PROTO ((void))

6.1.3.14 void yy_switch_to_buffer YY_PROTO ((YY_BUFFER_STATE new_buffer))

6.1.3.15 void yyrestart YY_PROTO ((FILE *input_file))

6.1.4 Variable Documentation

6.1.4.1 FILE* file

Definition at line 1158 of file ascii2docbook.c.

6.1.4.2 int len

Definition at line 1275 of file ascii2docbook.c.

Referenced by main().

6.1.4.3 yy_size_t size

Definition at line 1069 of file ascii2docbook.c.

6.1.4.4 yyconst short int yy_accept[9] [static]

Initial value:

```
{   0,
    0,    0,    5,    3,    4,    2,    1,    0
}
```

Definition at line 287 of file ascii2docbook.c.

6.1.4.5 yyconst short int yy_base[9] [static]

Initial value:

```
{   0,
    0,    0,    5,    6,    6,    6,    6,    6
}
```

Definition at line 329 of file ascii2docbook.c.

6.1.4.6 register char* yy_bp

Definition at line 921 of file ascii2docbook.c.

6.1.4.7 char* yy_c_buf_p = (char *) 0 [static]

Definition at line 219 of file ascii2docbook.c.

6.1.4.8 yyconst short int yy_chk[11] [static]

Initial value:

```
{   0,
    1,    1,    1,    1,    3,    8,    8,    8,    8,    8
}
```

Definition at line 344 of file ascii2docbook.c.

6.1.4.9 YY_BUFFER_STATE yy_current_buffer = 0 [static]

Definition at line 201 of file ascii2docbook.c.

6.1.4.10 yyconst short int yy_def[9] [static]

Initial value:

```
{   0,
    8,    1,    8,    8,    8,    8,    8,    0
}
```

Definition at line 334 of file ascii2docbook.c.

6.1.4.11 int yy_did_buffer_switch_on_eof [static]

Definition at line 226 of file ascii2docbook.c.

6.1.4.12 yyconst int yy_ec[256] [static]

Definition at line 292 of file ascii2docbook.c.

6.1.4.13 char yy_hold_char [static]

Definition at line 211 of file ascii2docbook.c.

6.1.4.14 int yy_init = 1 [static]

Definition at line 220 of file ascii2docbook.c.

6.1.4.15 char* yy_last_accepting_cpos [static]

Definition at line 350 of file ascii2docbook.c.

6.1.4.16 yy_state_type yy_last_accepting_state [static]

Definition at line 349 of file ascii2docbook.c.

6.1.4.17 yyconst int yy_meta[5] [static]

Initial value:

```
{   0,
    1,    1,    1,    1
}
```

Definition at line 324 of file ascii2docbook.c.

6.1.4.18 int yy_n_chars [static]

Definition at line 213 of file ascii2docbook.c.

6.1.4.19 yyconst short int yy_nxt[11] [static]

Initial value:

```
{   0,
    4,    5,    6,    7,    8,    3,    8,    8,    8,    8
}
```

Definition at line 339 of file ascii2docbook.c.

6.1.4.20 int yy_start = 0 [static]

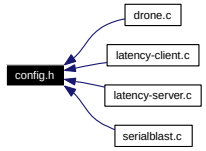
Definition at line 221 of file ascii2docbook.c.

6.1.4.21 char * yytext

Definition at line 359 of file ascii2docbook.c.

6.2 config.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define **PACKAGE** "latencytests"
- #define **PACKAGE_BUGREPORT** "twoerner.k@gmail.com"
- #define **PACKAGE_NAME** "Latency Tests"
- #define **PACKAGE_STRING** "Latency Tests 3.4.0"
- #define **PACKAGE_TARNAME** "latencytests"
- #define **PACKAGE_VERSION** "3.4.0"
- #define **VERSION** "3.4.0"
- #define **YYTEXT_POINTER** 1

6.2.1 Define Documentation

6.2.1.1 #define PACKAGE "latencytests"

Definition at line 5 of file config.h.

6.2.1.2 #define PACKAGE_BUGREPORT "twoerner.k@gmail.com"

Definition at line 8 of file config.h.

Referenced by usage().

6.2.1.3 #define PACKAGE_NAME "Latency Tests"

Definition at line 11 of file config.h.

6.2.1.4 #define PACKAGE_STRING "Latency Tests 3.4.0"

Definition at line 14 of file config.h.

Referenced by usage().

6.2.1.5 #define PACKAGE_TARNAME "latencytests"

Definition at line 17 of file config.h.

6.2.1.6 #define PACKAGE_VERSION "3.4.0"

Definition at line 20 of file config.h.

6.2.1.7 #define VERSION "3.4.0"

Definition at line 23 of file config.h.

6.2.1.8 #define YYTEXT_POINTER 1

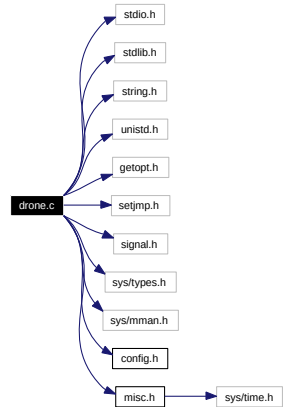
Definition at line 27 of file config.h.

6.3 drone.c File Reference

Latency Drone - runs on the target, copies data from infifo to outfifo.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <getopt.h>
#include <setjmp.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/mman.h>
#include "config.h"
#include "misc.h"
```

Include dependency graph for drone.c:



Functions

- static void **usage** (const char *)
- static void **signal_handler** (int)
- int **main** (int argc, char *argv[])

Variables

- static jmp_buf **env_G**
- static char **buf** [BUFSIZE]

6.3.1 Detailed Description

Latency Drone - runs on the target, copies data from infifo to outfifo.

If it's really dumb it'll open and close the fifo descriptors once for each read/write cycle.

Definition in file **drone.c**.

6.3.2 Function Documentation

6.3.2.1 int main (int *argc*, char * *argv*[])

Definition at line 40 of file drone.c.

References buf, BUFSIZE, env_G, open_fifo(), signal_handler(), STAT_OK, and usage().

6.3.2.2 static void signal_handler (int) [static]

Definition at line 141 of file drone.c.

References env_G.

Referenced by main().

6.3.2.3 static void usage (const char *) [static]

Definition at line 134 of file drone.c.

6.3.3 Variable Documentation

6.3.3.1 char buf[BUFSIZE] [static]

Definition at line 33 of file drone.c.

Referenced by generate_report(), and main().

6.3.3.2 jmp_buf env_G [static]

Definition at line 27 of file drone.c.

Referenced by main(), and signal_handler().

6.4 freqcurve.c File Reference

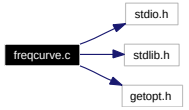
Analyze data to generate histogram values.

`#include <stdio.h>`

`#include <stdlib.h>`

`#include <getopt.h>`

Include dependency graph for freqcurve.c:



Defines

- #define **BUFSIZE** 256

Functions

- void **usage** (char *)
  Display program usage information.
- int **main** (int argc, char *argv[])

6.4.1 Detailed Description

Analyze data to generate histogram values.

This program generates the data necessary for creating a frequency curve from the provided data.

The raw input data needs to be provided in a file, one value per line. The data values are assumed to be floats written in ASCII.

Creating a frequency graph from raw data consists of defining a number of equally spaced ranges (or boxes) and counting the number of raw data points that fall in each of the ranges. At the end of processing we can say that X number of data points fell within the range from a to b, Y number of points fell within the range from b to c, et cetera. Therefore, in addition to providing the name of the file to process, this program needs to know how many ranges (or boxes) and the width of each box (or spacing). The first box is always assumed to start at zero.

The number of boxes is assumed to be specified as a natural number. The spacing is assumed to be specified as a float value.

So, for example, one could ask for 10 boxes with a spacing of 1 which would analyze all the data and count how many values fall between 0 and 1, 1 and 2, ... up to 9 to 10. Conversely, for a higher granularity one could ask for 20 boxes with a spacing of 0.5. This would still could all the values between 0 and 10 but use twice the number of boxes.

If you simply want to see the count of items in each box simply run this program without options, or perhaps specifying the "-v" option (to make the results clearer). In order to generate a *.png graph of this data (by using GNU plotutils' graph program) the box information needs to be

provided in "x y" format (without the quotes). In that case run the program with only the "-g" option to create the right numbers for graphing.

Definition in file **freqcurve.c**.

6.4.2 Define Documentation

6.4.2.1 #define BUFSIZE 256

Definition at line 48 of file freqcurve.c.

6.4.3 Function Documentation

6.4.3.1 int main (int *argc*, char * *argv*[])

Definition at line 53 of file freqcurve.c.

References buf, BUFSIZE, and usage().

6.4.3.2 static void usage (char * *name_p*)

Display program usage information.

Parameters:

← ***name_p*** Program name as a string to print as part of usage information.

This function displays the usage information for this application and then returns without exiting. It is up to the caller whether to exit normally or abnormally (if at all).

Definition at line 161 of file freqcurve.c.

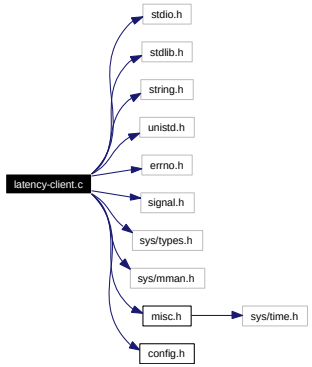
Referenced by cmdline_args(), and main().

6.5 latency-client.c File Reference

Latency Client - runs on the device to be tested.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/mman.h>
#include "misc.h"
#include "config.h"
```

Include dependency graph for latency-client.c:



Defines

- #define **DRONE_PGM** "drone"

Functions

- static void **usage** (const char *)
- static void **signal_handler** (int)
- int **main** (int argc, char *argv[])

Variables

- static char **fromSerial** [BUFSIZE]

- static char **fromFifo** [BUFSIZE]
- static char **leftFifoName** [16]
- static char **rightFifoName** [16]
- static int * **children_pG**
- static int **serialFd_G**
- static int **writeFifoFd_G**
- static int **readFifoFd_G**
- static int **drones_G**

6.5.1 Detailed Description

Latency Client - runs on the device to be tested.

This file defines the latency client. The latency client opens a serial device and sets up zero or more drones. It then waits for something to come along on the serial port. When something does, it reads it and passes it along to the first drone. It then waits for the last drone to pass the data back at which point it writes it back out to the serial port. If zero drones are specified then it simply copies the data back out to the serial port directly.

Definition in file **latency-client.c**.

6.5.2 Define Documentation

6.5.2.1 #define DRONE_PGM "drone"

Definition at line 33 of file latency-client.c.

Referenced by main().

6.5.3 Function Documentation

6.5.3.1 int main (int *argc*, char * *argv*[])

Definition at line 53 of file latency-client.c.

References BUFSIZE, children_pG, DRONE_PGM, drones_G, fromFifo, fromSerial, leftFifoName, open_fifo(), readFifoFd_G, rightFifoName, serial_conf(), serialFd_G, signal_handler(), STAT_OK, usage(), and writeFifoFd_G.

6.5.3.2 static void signal_handler (int) [static]

Definition at line 157 of file latency-client.c.

References children_pG, drones_G, leftFifoName, readFifoFd_G, serialFd_G, and writeFifoFd_G.

6.5.3.3 static void usage (const char *) [static]

Definition at line 145 of file latency-client.c.

References PACKAGE_BUGREPORT, and PACKAGE_STRING.

6.5.4 Variable Documentation

6.5.4.1 int* children_pG [static]

Definition at line 44 of file latency-client.c.

Referenced by main(), and signal_handler().

6.5.4.2 int drones_G [static]

Definition at line 47 of file latency-client.c.

Referenced by main(), and signal_handler().

6.5.4.3 char fromFifo[BUFSIZE] [static]

Definition at line 42 of file latency-client.c.

Referenced by main().

6.5.4.4 char fromSerial[BUFSIZE] [static]

Definition at line 41 of file latency-client.c.

Referenced by main().

6.5.4.5 char leftFifoName[16] [static]

Definition at line 43 of file latency-client.c.

Referenced by main(), and signal_handler().

6.5.4.6 int readFifoFd_G [static]

Definition at line 46 of file latency-client.c.

Referenced by main(), and signal_handler().

6.5.4.7 char rightFifoName[16] [static]

Definition at line 43 of file latency-client.c.

Referenced by main().

6.5.4.8 int serialFd_G [static]

Definition at line 45 of file latency-client.c.

Referenced by clean_up(), cmdline_args(), main(), and signal_handler().

6.5.4.9 int writeFifoFd_G [static]

Definition at line 46 of file latency-client.c.

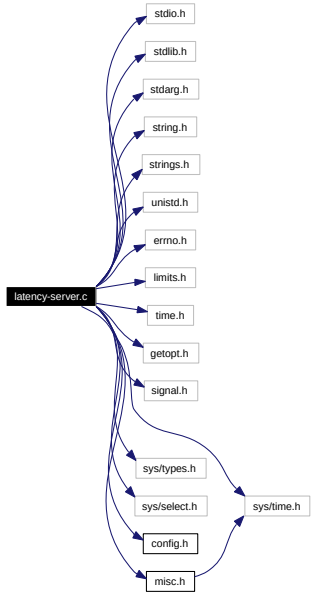
Referenced by main(), and signal_handler().

6.6 latency-server.c File Reference

Latency Server - runs on the desktop machine.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <strings.h>
#include <unistd.h>
#include <errno.h>
#include <limits.h>
#include <time.h>
#include <getopt.h>
#include <signal.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/select.h>
#include "config.h"
#include "misc.h"
```

Include dependency graph for latency-server.c:



Defines

- #define **SDEV_LEN** 64

Functions

- static void **cmdline_args** (int argc, char *argv[])
  Process command-line arguments.
- static void **generate_report** (void)
  Prints report to stdout.
- static void **clean_up** (void)
  Cleanup function.
- static void **usage** (char *name_p)
  Display program usage information.
- static void **how_to_use** (void)
- static void **output_msg** (const char *fmt_p,...)
  Printf() + logging.
- static double **convert_cpu_ticks_2_us** (unsigned long long)

- static void **ctrlc_handler** (int sig)
  Handle Ctrl-C.
- static void **ctrlz_handler** (int sig)
  Handle Ctrl-Z.
- static void **timer_handler** (int sig)
  Handle timer.
- int **main** (int argc, char *argv[], char *env[])

Variables

- static int **verbose_G** = 0
- static unsigned long long **iterations_G** = 0
- static unsigned long long **fixedIterations_G** = 0
- static int **serialFd_G**
- static int **fifoFd_G**
- static double **runAvg_G** = 0
- static unsigned long long **cpuTicksMax_G** = 0
- static unsigned long long **cpuTicksMin_G** = ULONG_MAX
- static unsigned long **pause_G** = 0
- static unsigned long **reportIterations_G** = 1
- static unsigned int **wantTimedReports_G** = 0
- static FILE * **logFile_pG** = NULL
- static unsigned **khz_G** = 1
- static float **loadavg_G**
- static float **uptime_G**
- static float **idletime_G**
- static long **ggp_G**
- static int **showGgp_G** = 0
- static volatile unsigned long long **duration_G**
- static char **serialDev_G** [SDEV_LEN]

6.6.1 Detailed Description

Latency Server - runs on the desktop machine.

This file defines the latency server. The latency server tests and measures the performance of a client machine over a serial connection. It takes a block of text which it reads on a local fifo, sends it to the client over a serial link, and receives data over the same serial link from the client. The target machine needs to be running a special version of the serial driver. The only thing different about this serial driver is that right before sending the data back it transmits (without curly braces): **{duration} {cpu clock KHz} {loadavg} {uptime} {idletime}**

It repeats this test any number of times (even infinite).

Statistics are gathered throughout program execution. There are a number of ways to generate a report:

- The application terminates itself when a fixed number of iterations are specified by the user and this number is reached.

- The application receives SIGINT (Ctrl-C on the controlling terminal) causing it to generate a report and then terminate.
- The application receives SIGTSTP (Ctrl-Z).

The format of the report depends on the verbosity level requested by the user.

Definition in file **latency-server.c**.

6.6.2 Define Documentation

6.6.2.1 #define SDEV_LEN 64

The name of the serial device.

Definition at line 129 of file latency-server.c.

Referenced by cmdline_args().

6.6.3 Function Documentation

6.6.3.1 static void clean_up (void) [static]

Cleanup function.

Closes, deallocates, cleans up any resources that need to be released.

Definition at line 384 of file latency-server.c.

References fifoFd_G, logFile_pG, and serialFd_G.

Referenced by ctrlc_handler(), and main().

6.6.3.2 static void cmdline_args (int *argc*, char * *argv*[]) [static]

Process command-line arguments.

Update global variables which modify how the program runs based on user command-line choices. Each invocation requires the name of the serial device and the fifo to be specified, setup and configure these two devices.

Definition at line 428 of file latency-server.c.

References fifoFd_G, fixedIterations_G, logFile_pG, open_fifo(), output_msg(), pause_G, reportIterations_G, SDEV_LEN, serial_conf(), serialDev_G, serialFd_G, showGgp_G, usage(), verbose_G, and wantTimedReports_G.

Referenced by main().

6.6.3.3 static double convert_cpu_ticks_2_us (unsigned long *long*) [static]

Definition at line 581 of file latency-server.c.

References khz_G.

Referenced by generate_report(), and main().

6.6.3.4 static void ctrlc_handler (int *sig*) [static]

Handle Ctrl-C.

Signal handler when SIGINT is received (Ctrl-C).

Definition at line 595 of file latency-server.c.

References clean_up(), and generate_report().

Referenced by main().

6.6.3.5 static void ctrlz_handler (int *sig*) [static]

Handle Ctrl-Z.

Signal handler for when SIGTSTP is received (Ctrl-Z).

Definition at line 608 of file latency-server.c.

References generate_report().

Referenced by main().

6.6.3.6 static void generate_report (void) [static]

Prints report to stdout.

This function generates and prints out a report to stdout. The amount of information in the report is dictated by the verbosity setting. A timestamp is included with each report.

Definition at line 327 of file latency-server.c.

References buf, convert_cpu_ticks_2_us(), cpuTicksMax_G, cpuTicksMin_G, duration_G, ggp_G, idletime_G, iterations_G, loadavg_G, output_msg(), runAvg_G, showGgp_G, uptime_G, and verbose_G.

Referenced by ctrlc_handler(), ctrlz_handler(), main(), and timer_handler().

6.6.3.7 static void how_to_use (void) [static]

Definition at line 542 of file latency-server.c.

References output_msg().

Referenced by main().

6.6.3.8 int main (int *argc*, char * *argv*[], char * *env*[])

Definition at line 136 of file latency-server.c.

References BUFSIZE, clean_up(), cmdline_args(), convert_cpu_ticks_2_us(), cpuTicksMax_G, cpuTicksMin_G, ctrlc_handler(), ctrlz_handler(), duration_G, fifoFd_G, fixedIterations_G, generate_report(), ggp_G, how_to_use(), idletime_G, iterations_G, khz_G, len, loadavg_G, MAX, MIN, output_msg(), pause_G, read_until(), reportIterations_G, runAvg_G, serial_conf(), serialDev_G, serialFd_G, showGgp_G, STAT_OK, STAT_TIMEOUT, timer_handler(), uptime_G, verbose_G, wantTimedReports_G, and write_n().

6.6.3.9 static void output_msg (const char * *fmt_p*, ...) [static]

Printf() + logging.

I've added this function to consolodate all the output through one vector, that way if logging is enabled is not a concern to some part of the program simply wanting to output some information. If logging is enabled all messages will be printed to the output and duplicated to the log file, otherwise all messages are simply just printed to stdout like normal.

Definition at line 402 of file latency-server.c.

References logFile_pG.

Referenced by cmdline_args(), generate_report(), how_to_use(), and main().

6.6.3.10 static void timer_handler (int *sig*) [static]

Handle timer.

The user is able to optionally specify that every N seconds a report be be generated. This function is what is called when the timer goes off.

Definition at line 620 of file latency-server.c.

References generate_report(), and wantTimedReports_G.

Referenced by main().

6.6.3.11 static void usage (char * *name_p*) [static]

Display program usage information.

Parameters:

← ***name_p*** Program name as a string to print as part of usage information.

This function displays the usage information for this application and then returns without exiting. It is up to the caller whether to exit normally or abnormally (if at all).

6.6.4 Variable Documentation

6.6.4.1 unsigned long long cpuTicksMax_G = 0 [static]

Stores the longest and shortest round-trip time. [in target CPU ticks]

Definition at line 90 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.2 unsigned long long cpuTicksMin_G = ULONG_MAX [static]

Definition at line 90 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.3 volatile unsigned long long duration_G [static]

The duration of the most recent round-trip.

Definition at line 126 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.4 int fifoFd_G [static]

Definition at line 84 of file latency-server.c.

Referenced by clean_up(), cmdline_args(), and main().

6.6.4.5 unsigned long long fixedIterations_G = 0 [static]

If a user has specified the number of iterations it is stored here, otherwise 0 to indicate an infinite run.

Definition at line 81 of file latency-server.c.

Referenced by cmdline_args(), and main().

6.6.4.6 long ggp_G [static]

The rcu ggp value.

Definition at line 120 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.7 float idletime_G [static]

The amount of idle time (in seconds) of the target. From /proc/uptime.

Definition at line 117 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.8 unsigned long long iterations_G = 0 [static]

Counts how many iterations have actually been performed.

Definition at line 77 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.9 unsigned khz_G = 1 [static]

The clock rate of the target.

Definition at line 108 of file latency-server.c.

Referenced by convert_cpu_ticks_2_us(), and main().

6.6.4.10 float loadavg_G [static]

The loadavg of the target.

Definition at line 111 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.11 FILE* logFile_pG = NULL [static]

If the user requests output to a log file this is the *FILE, otherwise NULL

Definition at line 105 of file latency-server.c.

Referenced by clean_up(), cmdline_args(), and output_msg().

6.6.4.12 unsigned long pause_G = 0 [static]

If the user requests an inter-iteration pause it is stored here, otherwise 0 to indicate no pause.

Definition at line 94 of file latency-server.c.

Referenced by cmdline_args(), and main().

6.6.4.13 unsigned long reportIterations_G = 1 [static]

If the user wants reports every N interations, N is stored here, otherwise 0 indicates no iteration reporting is requested. The default is every transaction is reported.

Definition at line 99 of file latency-server.c.

Referenced by cmdline_args(), and main().

6.6.4.14 double runAvg_G = 0 [static]

Stores a running counter of the average round-trip time.

Definition at line 87 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.15 char serialDev_G[SDEV_LEN] [static]

Definition at line 130 of file latency-server.c.

Referenced by cmdline_args(), and main().

6.6.4.16 int serialFd_G [static]

File descriptors.

Definition at line 84 of file latency-server.c.

6.6.4.17 int showGgp_G = 0 [static]

Whether or not to show the ggp value.

Definition at line 123 of file latency-server.c.

Referenced by cmdline_args(), generate_report(), and main().

6.6.4.18 float uptime_G [static]

The uptime (in seconds) of the target. From /proc/uptime.

Definition at line 114 of file latency-server.c.

Referenced by generate_report(), and main().

6.6.4.19 int verbose_G = 0 [static]

Tracks the user's requested verbosity level.

Definition at line 74 of file latency-server.c.

Referenced by cmdline_args(), generate_report(), and main().

6.6.4.20 unsigned int wantTimedReports_G = 0 [static]

Generate a report every N seconds, zero indicates no reports requested.

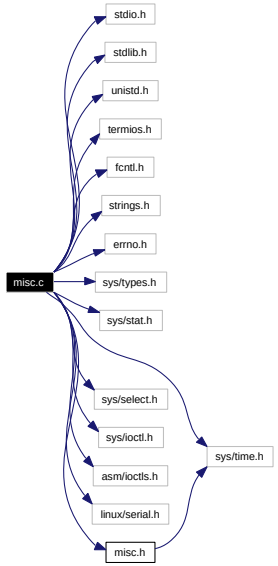
Definition at line 102 of file latency-server.c.

Referenced by cmdline_args(), main(), and timer_handler().

6.7 misc.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <termios.h>
#include <fcntl.h>
#include <strings.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/select.h>
#include <sys/ioctl.h>
#include <asm/ioctls.h>
#include <linux/serial.h>
#include "misc.h"
```

Include dependency graph for misc.c:



Functions

- int **serial_conf** (char *dev_p)
  Configure a serial port in a standard way.
- **Status_t read_until** (int fd, char *dst_p, size_t count, char ch, struct timeval *timeout_p)
  Read(2) from a file descriptor until a specific character is encountered, with timeout.
- **Status_t write_n** (int fd, const char *src_p, size_t count, struct timeval *timeout_p)
  Write(2) all N characters to a file descriptor within a specified timeout.
- **Status_t open_fifo** (char *name_p, int *fdOut_p)
  Open a named fifo.

6.7.1 Function Documentation

6.7.1.1 Status_t open_fifo (char * *name_p*, int * *fdOut_p*)

Open a named fifo.

Parameters:
- ← ***name_p*** The filename of the fifo to open.
- → ***fdOut_p*** A pointer to a location to store the opened file descriptor.

Returns:
Status_t(p. 50):
- **STAT_OK**(p. 50) Everything went well and *fdOut_p contains a valid file descriptor for this fifo.
- **STAT_NULL**(p. 50) fdOut_p is NULL, no place to store the opened file descriptor.
- **STAT_SEE_C_ERRNO**(p. 50) A problem occurred creating or opening the fifo.

Definition at line 274 of file misc.c.

References STAT_NULL, STAT_OK, and STAT_SEE_C_ERRNO.

Referenced by cmdline_args(), and main().

6.7.1.2 Status_t read_until (int *fd*, char * *dst_p*, size_t *count*, char *ch*, struct timeval * *timeout_p*)

Read(2) from a file descriptor until a specific character is encountered, with timeout.

Parameters:
- ← ***fd*** The file descriptor to read from.
- → ***dst_p*** A pointer to the location to put the read bytes.
- ← ***count*** The total number of bytes in *dst_p. Note if you are reading a string you must reserve space for the terminating NULL by passing in the size of *dst_p minus 1.
- ← ***ch*** The character to look for.

← ***timeout_p*** The maximum amount of time to wait, use NULL to represent an infinite timeout.

Returns:
A **Status_t**(p.50) value:

- **STAT_OK**(p.50) everything went well.
- **STAT_TIMEOUT**(p.50) the timeout expired before the character was found.
- **STAT_FULL**(p.50) the end of the buffer was encountered before the character was read.
- **STAT_SEE_C_ERRNO**(p.50) some system call failed (select(2), read(2)).
- **STAT_EOF**(p.50) the end-of-file has been reached and the character was not found.
- **STAT_NOT_OPEN**(p.50) the fd is invalid.
- **STAT_NULL**(p.50) the destination pointer is NULL.

Note:
The performance of this function is quite poor since it has to read(2) each character one by one. This is because there is no buffering I/O on file descriptors and therefore no way to push any characters back. I didn't want to get into a scheme of passing back the "extra" characters.
If the file-descriptor was opened NON-BLOCKING this function will similarly not block, and visa versa.

Definition at line 124 of file misc.c.

References STAT_EOF, STAT_FULL, STAT_NOT_OPEN, STAT_NULL, STAT_OK, STAT_SEE_C_ERRNO, and STAT_TIMEOUT.

Referenced by main().

6.7.1.3 int serial_conf (char * *dev_p*)

Configure a serial port in a standard way.

Parameters:
← ***dev_p*** A string specifying the device to configure (e.g. "/dev/ttyS0").

Returns:
If everything goes well, returns the file descriptor of the newly configured serial device. Otherwise a -1 indicates failure.

Knowing that I want to configure all serial devices in the same way with the same serial attributes I created one function that when passed the name of the device to configure, would open and configure the device in a known way and return the file descriptor of the newly opened and configured serial device.

Definition at line 37 of file misc.c.

Referenced by cmdline_args(), and main().

6.7.1.4 Status_t write_n (int *fd*, const char * *src_p*, size_t *count*, struct timeval * *timeout_p*)

Write(2) all N characters to a file descriptor within a specified timeout.

Parameters:
← ***fd*** The file descriptor to read from.
← ***src_p*** Points to the string of data to send.
← ***count*** The number of bytes to transfer. Note that this function treats all data the same; it makes no distinction for strings or binary data. Therefore if you want to send a string plus its terminating NULL you must remember to adjust the count accordingly.
← ***timeout_p*** A pointer to the maximum amount of time to wait for the write to complete. Passing a NULL implies an infinite wait.

Returns:
A value of **Status_t**(p.50):

- **STAT_OK**(p.50) everything went well.
- **STAT_NOT_OPEN**(p.50) the file descriptor isn't valid.
- **STAT_NULL**(p.50) the src_p is NULL.
- **STAT_TIMEOUT**(p.50) the write(2) timed out.
- **STAT_SEE_C_ERRNO**(p.50) problems with either write(2) or select(2).

Note:
If the file-descriptor was opened NON-BLOCKING this function will similarly not block, and visa versa.

Definition at line 212 of file misc.c.

References STAT_NOT_OPEN, STAT_NULL, STAT_OK, STAT_SEE_C_ERRNO, and STAT_TIMEOUT.

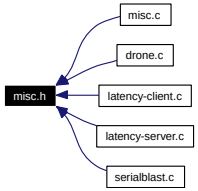
Referenced by main().

6.8 misc.h File Reference

`#include <sys/time.h>`

Include dependency graph for misc.h:

This graph shows which files directly or indirectly include this file:



Defines

- #define **BUFSIZE** 256
- #define **MAX**(a, b) ((a) > (b) ? (a) : (b))
- #define **MIN**(a, b) ((a) < (b) ? (a) : (b))

Enumerations

- enum **Status_t** {
  STAT_OK, **STAT_TIMEOUT**, **STAT_FULL**, **STAT_SEE_C_ERRNO**,
  STAT_EOF, **STAT_NOT_OPEN**, **STAT_NULL** }

Functions

- int **serial_conf** (char *)
  Configure a serial port in a standard way.
- **Status_t read_until** (int fd, char *dst_p, size_t count, char ch, struct timeval *timeout_p)
  Read(2) from a file descriptor until a specific character is encountered, with timeout.
- **Status_t write_n** (int fd, const char *src_p, size_t count, struct timeval *timeout_p)
  Write(2) all N characters to a file descriptor within a specified timeout.
- **Status_t open_fifo** (char *name_p, int *fdOut_p)
  Open a named fifo.

6.8.1 Define Documentation

6.8.1.1 #define BUFSIZE 256

Definition at line 11 of file misc.h.

Referenced by main().

6.8.1.2 #define MAX(a, b) ((a) > (b) ? (a) : (b))

Definition at line 12 of file misc.h.

Referenced by main().

6.8.1.3 #define MIN(a, b) ((a) < (b) ? (a) : (b))

Definition at line 13 of file misc.h.

Referenced by main().

6.8.2 Enumeration Type Documentation

6.8.2.1 enum Status_t

Enumerator:

STAT_OK
STAT_TIMEOUT
STAT_FULL
STAT_SEE_C_ERRNO
STAT_EOF
STAT_NOT_OPEN
STAT_NULL

Definition at line 15 of file misc.h.

6.8.3 Function Documentation

6.8.3.1 Status_t open_fifo (char * *name_p*, int * *fdOut_p*)

Open a named fifo.

Parameters:

← ***name_p*** The filename of the fifo to open.

→ ***fdOut_p*** A pointer to a location to store the opened file descriptor.

Returns:

Status_t(p. 50):

- **STAT_OK**(p. 50) Everything went well and *fdOut_p contains a valid file descriptor for this fifo.

- **STAT_NULL**(p.50) fdOut_p is NULL, no place to store the opened file descriptor.
- **STAT_SEE_C_ERRNO**(p.50) A problem occurred creating or opening the fifo.

Definition at line 274 of file misc.c.

References STAT_NULL, STAT_OK, and STAT_SEE_C_ERRNO.

Referenced by cmdline_args(), and main().

6.8.3.2 Status_t read_until (int *fd*, char * *dst_p*, size_t *count*, char *ch*, struct timeval * *timeout_p*)

Read(2) from a file descriptor until a specific character is encountered, with timeout.

Parameters:

← ***fd*** The file descriptor to read from.

→ ***dst_p*** A pointer to the location to put the read bytes.

← ***count*** The total number of bytes in *dst_p. Note if you are reading a string you must reserve space for the terminating NULL by passing in the size of *dst_p minus 1.

← ***ch*** The character to look for.

← ***timeout_p*** The maximum amount of time to wait, use NULL to represent an infinite timeout.

Returns:

A **Status_t**(p.50) value:

- **STAT_OK**(p.50) everything went well.
- **STAT_TIMEOUT**(p.50) the timeout expired before the character was found.
- **STAT_FULL**(p.50) the end of the buffer was encountered before the character was read.
- **STAT_SEE_C_ERRNO**(p.50) some system call failed (select(2), read(2)).
- **STAT_EOF**(p.50) the end-of-file has been reached and the character was not found.
- **STAT_NOT_OPEN**(p.50) the fd is invalid.
- **STAT_NULL**(p.50) the destination pointer is NULL.

Note:

The performance of this function is quite poor since it has to read(2) each character one by one. This is because there is no buffering I/O on file descriptors and therefore no way to push any characters back. I didn't want to get into a scheme of passing back the "extra" characters.
If the file-descriptor was opened NON-BLOCKING this function will similarly not block, and visa versa.

Definition at line 124 of file misc.c.

References STAT_EOF, STAT_FULL, STAT_NOT_OPEN, STAT_NULL, STAT_OK, STAT_SEE_C_ERRNO, and STAT_TIMEOUT.

Referenced by main().

6.8.3.3 int serial_conf (char * *dev_p*)

Configure a serial port in a standard way.

Parameters:

← ***dev_p*** A string specifying the device to configure (e.g. "/dev/ttyS0").

Returns:

If everything goes well, returns the file descriptor of the newly configured serial device. Otherwise a **-1** indicates failure.

Knowing that I want to configure all serial devices in the same way with the same serial attributes I created one function that when passed the name of the device to configure, would open and configure the device in a known way and return the file descriptor of the newly opened and configured serial device.

Definition at line 37 of file misc.c.

Referenced by cmdline_args(), and main().

6.8.3.4 Status_t write_n (int *fd*, const char * *src_p*, size_t *count*, struct timeval * *timeout_p*)

Write(2) all N characters to a file descriptor within a specified timeout.

Parameters:

← ***fd*** The file descriptor to read from.

← ***src_p*** Points to the string of data to send.

← ***count*** The number of bytes to transfer. Note that this function treats all data the same; it makes no distinction for strings or binary data. Therefore if you want to send a string plus its terminating NULL you must remember to adjust the count accordingly.

← ***timeout_p*** A pointer to the maximum amount of time to wait for the write to complete. Passing a NULL implies an infinite wait.

Returns:

A value of **Status_t**(p.50):

- **STAT_OK**(p.50) everything went well.
- **STAT_NOT_OPEN**(p.50) the file descriptor isn't valid.
- **STAT_NULL**(p.50) the src_p is NULL.
- **STAT_TIMEOUT**(p.50) the write(2) timed out.
- **STAT_SEE_C_ERRNO**(p.50) problems with either write(2) or select(2).

Note:

If the file-descriptor was opened NON-BLOCKING this function will similarly not block, and visa versa.

Definition at line 212 of file misc.c.

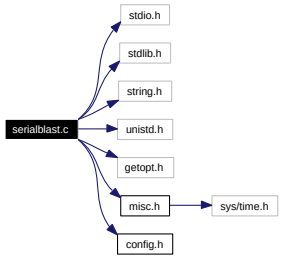
References STAT_NOT_OPEN, STAT_NULL, STAT_OK, STAT_SEE_C_ERRNO, and STAT_TIMEOUT.

Referenced by main().

6.9 serialblast.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <getopt.h>
#include "misc.h"
#include "config.h"
```

Include dependency graph for serialblast.c:



Functions

- static void **cmdline_args** (int argc, char *argv[])
- static void **usage** (char *name_p)
- int **main** (int argc, char *argv[])

Variables

- static unsigned long **pause_G** = 500000
- static int **textNeedsFree_G** = 0
- static char * **textToSend_pG** = "00000000011111111112222222222333333333344444444455555555556666666
- static char * **serialDev_pG** = NULL

6.9.1 Function Documentation

6.9.1.1 static void cmdline_args (int *argc*, char * *argv*[]) [static]

Definition at line 57 of file serialblast.c.

References pause_G, serialDev_pG, textNeedsFree_G, textToSend_pG, and usage().

6.9.1.2 int main (int *argc*, char * *argv*[])

Definition at line 24 of file serialblast.c.

References cmdline_args(), len, pause_G, serial_conf(), serialDev_pG, textNeedsFree_G, textToSend_pG, and write_n().

6.9.1.3 static void usage (char * *name_p*) [static]

6.9.2 Variable Documentation

6.9.2.1 unsigned long pause_G = 500000 [static]

Definition at line 18 of file serialblast.c.

6.9.2.2 char* serialDev_pG = NULL [static]

Definition at line 21 of file serialblast.c.

Referenced by cmdline_args(), and main().

6.9.2.3 int textNeedsFree_G = 0 [static]

Definition at line 19 of file serialblast.c.

Referenced by cmdline_args(), and main().

6.9.2.4 char* textToSend_pG = "00000000011111111112222222222333333333344444444455555555556666666 [static]

Definition at line 20 of file serialblast.c.

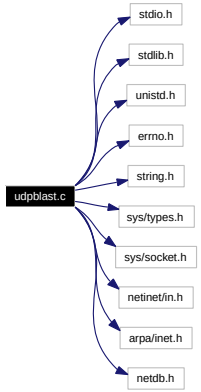
Referenced by cmdline_args(), and main().

6.10 udpblast.c File Reference

Send a UDP packet.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
```

Include dependency graph for udpblast.c:



Defines

- #define **PORT_NUM** 5002

Functions

- int **main** (int argc, char *argv[])

6.10.1 Detailed Description

Send a UDP packet.

On the command-line specify the hostname to address the packet to and provide a string that will form the message to send. This message is sent 100 times per second.

Definition in file **udpblast.c**.

6.10.2 Define Documentation

6.10.2.1 #define PORT_NUM 5002

Definition at line 25 of file udpblast.c.

Referenced by main().

6.10.3 Function Documentation

6.10.3.1 int main (int *argc*, char * *argv*[])

Definition at line 28 of file udpblast.c.

References PORT_NUM.

Index