

log_ctc41

Generated by Doxygen 1.8.17

1 File Index	1
1.1 File List	1
2 File Documentation	3
2.1 log.c File Reference	3
2.1.1 Function Documentation	4
2.1.1.1 closePrinter()	4
2.1.1.2 doneLEXstartSYN()	4
2.1.1.3 doneSYNstartTAB()	5
2.1.1.4 doneTABstartGEN()	5
2.1.1.5 fflushc()	5
2.1.1.6 initializePrinter()	5
2.1.1.7 pc()	6
2.1.1.8 pce()	6
2.1.1.9 pp()	7
2.1.1.10 splitFileName()	7
2.1.2 Variable Documentation	8
2.1.2.1 currentState	8
2.1.2.2 fileER_	8
2.1.2.3 fileGEN	8
2.1.2.4 fileLEX	8
2.1.2.5 filesOpened	9
2.1.2.6 fileSYN	9
2.1.2.7 fileTAB	9
2.2 log.h File Reference	9
2.2.1 Typedef Documentation	10
2.2.1.1 FileDestination	10
2.2.2 Enumeration Type Documentation	10
2.2.2.1 fileDestination	10
2.2.3 Function Documentation	11
2.2.3.1 closePrinter()	11
2.2.3.2 doneLEXstartSYN()	11
2.2.3.3 doneSYNstartTAB()	11
2.2.3.4 doneTABstartGEN()	12
2.2.3.5 fflushc()	12
2.2.3.6 initializePrinter()	12
2.2.3.7 pc()	13
2.2.3.8 pce()	13
2.2.3.9 pp()	13
Index	15

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

log.c	3
log.h	9

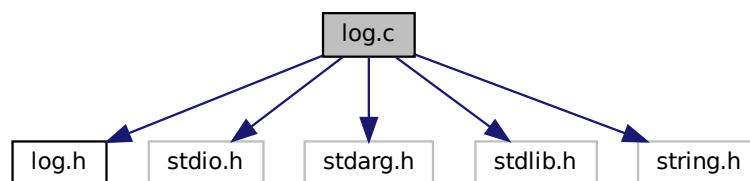
Chapter 2

File Documentation

2.1 log.c File Reference

```
#include "log.h"
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
```

Include dependency graph for log.c:



Functions

- void [splitFileName](#) (const char *fullFileName, char *path, char *fileName, char *extension)
aux func: split fullFileName (with full path) into path/fileName/extension
- void [initializePrinter](#) (const char *path, const char *baseName, [FileDestination](#) files2open)
open the files specified by files2open in the directory specified by path, with the basename specified
- void [closePrinter](#) ()
closes all opened files
- void [doneLEXstartSYN](#) ()
sets the curent compilation stage to SYN (syntatic analysis)
- void [doneSYNstartTAB](#) ()
sets the curent compilation stage to TAB (symbol table)
- void [doneTABstartGEN](#) ()
sets the curent compilation stage to GEN (code generation)

- void `fflushc` ()
flushes all opened files.
- void `pc` (const char *format,...)
prints in CURRENT output file AND stdout
- void `pce` (const char *format,...)
prints in CURRENT output file AND stdout AND error file (3-way)
- void `pp` (`FileDestination` destination, const char *format,...)
prints in both chosen output file(s) AND stdout

Variables

- FILE * `fileER_`
error output
- FILE * `fileLEX`
lexical analysis output
- FILE * `fileSYN`
syntatic analysis output
- FILE * `fileTAB`
symbol table output
- FILE * `fileGEN`
generated code output
- `FileDestination` `filesOpened`
sets which files will be opened. e.g. if you will only implement up to symbol table generation, do not open the file to output the generated code.
- `FileDestination` `currentState`
marks the current stage of the compilation, used for pc and pce functions

2.1.1 Function Documentation

2.1.1.1 `closePrinter()`

```
void closePrinter ( )
```

closes all opened files

2.1.1.2 `doneLEXstartSYN()`

```
void doneLEXstartSYN ( )
```

sets the curent compilation stage to SYN (syntatic analysis)

2.1.1.3 doneSYNstartTAB()

```
void doneSYNstartTAB ( )
```

sets the current compilation stage to TAB (symbol table)

2.1.1.4 doneTABstartGEN()

```
void doneTABstartGEN ( )
```

sets the current compilation stage to GEN (code generation)

2.1.1.5 fflushc()

```
void fflushc ( )
```

flushes all opened files.

2.1.1.6 initializePrinter()

```
void initializePrinter (
    const char * path,
    const char * baseName,
    FileDestination files2open )
```

open the files specified by files2open in the directory specified by path, with the basename specified

- sets currentState to LEX - the first stage of compilation is always lexical analysis
- the supplied main code already calls this function adequately. Probably, students will not need to change it.

Parameters

<i>path</i>	directory for detailed output files
<i>basename</i>	the radical part of the file name
<i>files2open</i>	choose which files to open. If not all compilation stages will be run, you may choose to not open all files

example usage:

```
initializePrinter(detailpath, pgm, LOGALL); // open detailed output files
```

into the path given by `detailpath`, with the basename given by `pgm`, and `LO←GALL` means all files will be opened Here is the call graph for this function:



2.1.1.7 pc()

```
void pc (
    const char * format,
    ... )
```

prints in CURRENT output file AND stdout

- use this to **print usual output messages into the current compilation stage output**.
- this function will be used most of the time.

Parameters

<i>format</i>	variadic parameters, to be used as <code>fprintf</code>
---------------	---

example usage:

```
pc("my message n. %i is %s", var_int, var_pointerchar); // prints on stdout
AND the output corresponding to the current compilation stage.
```

2.1.1.8 pce()

```
void pce (
    const char * format,
    ... )
```

prints in CURRENT output file AND stdout AND error file (3-way)

use this to **print error messages into the current compilation stage output**.

Parameters

<i>format</i>	variadic parameters, to be used as <code>fprintf</code>
---------------	---

example usage:

```
pce("my message n. %i is %s",var_int,var_pointerchar); // prints on stdout
AND error AND the output corresponding to the current compilation stage.
```

2.1.1.9 pp()

```
void pp (
    FileDestination destination,
    const char * format,
    ... )
```

prints in both chosen output file(s) AND stdout

this function is intended for **hardcoded fine control over output files**.

- it is a *variadic function* which accepts a variable number of arguments. It repasses its arguments to fprintf, printing into the output files indicated by the destination argument.
- it does not care about the current state.
- It checks if the file was set to be opened before printing

Parameters

<i>destination</i>	bitflag setting which output files will be used
<i>format</i>	after the destination flag, this function should be used as fprintf.

example usage (check the possible flags on the [enum fileDestination](#)) :

```
pp(SYN, "my message n. %i is %s",var_int,var_pointerchar); // prints on
stdout and sintatic analysis output
```

```
pp(SYN | LEX, "my message n. %i is %s",var_int,var_pointerchar); // prints
on stdout, lexic and sintatic analysis outputs
```

```
pp(TAB | ERR, "my message n. %i is %s",var_int,var_pointerchar); // prints
on stdout, symbol table and error outputs
```

2.1.1.10 splitFileName()

```
void splitFileName (
    const char * fullFileName,
    char * path,
    char * fileName,
    char * extension )
```

aux func: split fullFileName (with full path) into path/fileName/extension

students usually will not need to use this function

Here is the caller graph for this function:



2.1.2 Variable Documentation

2.1.2.1 currentState

`FileDestination` currentState

marks the current stage of the compilation, used for pc and pce functions

2.1.2.2 fileER_

`FILE*` fileER_

error output

2.1.2.3 fileGEN

`FILE*` fileGEN

generated code output

2.1.2.4 fileLEX

`FILE*` fileLEX

lexical analysis output

2.1.2.5 filesOpened

`FileDestination` filesOpened

sets which files will be opened. e.g. if you will only implement up to symbol table generation, do not open the file to output the generated code.

2.1.2.6 fileSYN

`FILE*` fileSYN

syntactic analysis output

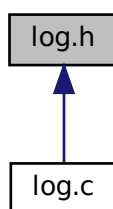
2.1.2.7 fileTAB

`FILE*` fileTAB

symbol table output

2.2 log.h File Reference

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



Typedefs

- typedef enum `fileDestination` FileDestination
bitmask to select output files

Enumerations

- enum `fileDestination` {
`ER_` = 0x1, `LEX` = 0x2, `LER` = 0x3, `SYN` = 0x4,
`SER` = 0x5, `TAB` = 0x8, `TER` = 0x9, `GEN` = 0x10,
`GER` = 0x11, `UP2SYN` = 0x7, `UP2TAB` = 0xF, `LOGALL` = 0x1F }
bitmask to select output files

Functions

- void `initializePrinter` (const char *path, const char *baseName, `FileDestination` files2open)
open the files specified by files2open in the directory specified by path, with the basename specified
- void `pp` (`FileDestination` destination, const char *format,...)
prints in both chosen output file(s) AND stdout
- void `doneLEXstartSYN` ()
sets the curent compilation stage to SYN (syntatic analysis)
- void `doneSYNstartTAB` ()
sets the curent compilation stage to TAB (symbol table)
- void `doneTABstartGEN` ()
sets the curent compilation stage to GEN (code generation)
- void `pc` (const char *format,...)
prints in CURRENT output file AND stdout
- void `pce` (const char *format,...)
prints in CURRENT output file AND stdout AND error file (3-way)
- void `fflushc` ()
flushes all opened files.
- void `closePrinter` ()
closes all opened files

2.2.1 Typedef Documentation

2.2.1.1 FileDestination

```
typedef enum fileDestination FileDestination
```

bitmask to select output files

2.2.2 Enumeration Type Documentation

2.2.2.1 fileDestination

```
enum fileDestination
```

bitmask to select output files

Enumerator

ER_	NOT STDERR!!! JUST A FILE TO STORE YOUR ERR MSGS!!
LEX	lexical
LER	LEX AND ERR.
SYN	syntatic analysis
SER	syntatic analysis AND error
TAB	symbol table
TER	symbol table AND error
GEN	code generation
GER	code generation AND error
UP2SYN	ER + LEX + SYN - all up to synthatic analysis AND ERROR.
UP2TAB	ER + LEX + SYN + TAB - all up to symbol table AND ERROR.
LOGALL	ER + LEX + SYN + TAB + GEN - everything, including code generation AND ERROR.

2.2.3 Function Documentation

2.2.3.1 closePrinter()

```
void closePrinter ( )
```

closes all opened files

2.2.3.2 doneLEXstartSYN()

```
void doneLEXstartSYN ( )
```

sets the curent compilation stage to SYN (syntatic analysis)

2.2.3.3 doneSYNstartTAB()

```
void doneSYNstartTAB ( )
```

sets the curent compilation stage to TAB (symbol table)

2.2.3.4 doneTABstartGEN()

```
void doneTABstartGEN ( )
```

sets the current compilation stage to GEN (code generation)

2.2.3.5 fflushc()

```
void fflushc ( )
```

flushes all opened files.

2.2.3.6 initializePrinter()

```
void initializePrinter (
    const char * path,
    const char * baseName,
    FileDestination files2open )
```

open the files specified by files2open in the directory specified by path, with the basename specified

- sets currentState to LEX - the first stage of compilation is always lexical analysis
- the supplied main code already calls this function adequately. Probably, students will not need to change it.

Parameters

<i>path</i>	directory for detailed output files
<i>basename</i>	the radical part of the file name
<i>files2open</i>	choose which files to open. If not all compilation stages will be run, you may choose to not open all files

example usage:

```
initializePrinter(detailpath, pgm, LOGALL); // open detailed output files
into the path given by detailpath, with the basename given by pgm, and LOGALL means all files will be opened
```

Here is the call graph for this function:



2.2.3.7 pc()

```
void pc (  
    const char * format,  
    ... )
```

prints in CURRENT output file AND stdout

- use this to **print usual output messages into the current compilation stage output**.
- this function will be used most of the time.

Parameters

<i>format</i>	variadic parameters, to be used as fprintf
---------------	--

example usage:

```
pc("my message n. %i is %s", var_int, var_pointerchar); // prints on stdout  
AND the output corresponding to the current compilation stage.
```

2.2.3.8 pce()

```
void pce (  
    const char * format,  
    ... )
```

prints in CURRENT output file AND stdout AND error file (3-way)

use this to **print error messages into the current compilation stage output**.

Parameters

<i>format</i>	variadic parameters, to be used as fprintf
---------------	--

example usage:

```
pce("my message n. %i is %s", var_int, var_pointerchar); // prints on stdout  
AND error AND the output corresponding to the current compilation stage.
```

2.2.3.9 pp()

```
void pp (  
    FileDestination destination,  
    const char * format,  
    ... )
```

prints in both chosen output file(s) AND stdout

this function is intended for **hardcoded fine control over output files**.

- it is a *variadic function* which accepts a variable number of arguments. It repasses its arguments to fprintf, printing into the output files indicated by the destination argument.
- it does not care about the current state.
- It checks if the file was set to be opened before printing

Parameters

<i>destination</i>	bitflag setting which output files will be used
<i>format</i>	after the destination flag, this function should be used as fprintf.

example usage (check the possible flags on the [enum fileDestination](#)) :

```
pp(SYN, "my message n. %i is %s", var_int, var_pointerchar); // prints on  
stdout and sintatic analysis output
```

```
pp(SYN | LEX, "my message n. %i is %s", var_int, var_pointerchar); // prints  
on stdout, lexic and sintatic analysis outputs
```

```
pp(TAB | ERR, "my message n. %i is %s", var_int, var_pointerchar); // prints  
on stdout, symbol table and error outputs
```

Index

closePrinter
 log.c, [4](#)
 log.h, [11](#)
currentState
 log.c, [8](#)

doneLEXstartSYN
 log.c, [4](#)
 log.h, [11](#)
doneSYNstartTAB
 log.c, [4](#)
 log.h, [11](#)
doneTABstartGEN
 log.c, [5](#)
 log.h, [11](#)

ER_
 log.h, [11](#)

fflushc
 log.c, [5](#)
 log.h, [12](#)
FileDestination
 log.h, [10](#)
fileDestination
 log.h, [10](#)
fileER_
 log.c, [8](#)
fileGEN
 log.c, [8](#)
fileLEX
 log.c, [8](#)
filesOpened
 log.c, [8](#)
fileSYN
 log.c, [9](#)
fileTAB
 log.c, [9](#)

GEN
 log.h, [11](#)
GER
 log.h, [11](#)

initializePrinter
 log.c, [5](#)
 log.h, [12](#)

LER
 log.h, [11](#)
LEX

 log.h, [11](#)
log.c, [3](#)
 closePrinter, [4](#)
 currentState, [8](#)
 doneLEXstartSYN, [4](#)
 doneSYNstartTAB, [4](#)
 doneTABstartGEN, [5](#)
 fflushc, [5](#)
 fileER_, [8](#)
 fileGEN, [8](#)
 fileLEX, [8](#)
 filesOpened, [8](#)
 fileSYN, [9](#)
 fileTAB, [9](#)
 initializePrinter, [5](#)
 pc, [6](#)
 pce, [6](#)
 pp, [7](#)
 splitFileName, [7](#)
log.h, [9](#)
 closePrinter, [11](#)
 doneLEXstartSYN, [11](#)
 doneSYNstartTAB, [11](#)
 doneTABstartGEN, [11](#)
 ER_, [11](#)
 fflushc, [12](#)
 FileDestination, [10](#)
 fileDestination, [10](#)
 GEN, [11](#)
 GER, [11](#)
 initializePrinter, [12](#)
 LER, [11](#)
 LEX, [11](#)
 LOGALL, [11](#)
 pc, [13](#)
 pce, [13](#)
 pp, [13](#)
 SER, [11](#)
 SYN, [11](#)
 TAB, [11](#)
 TER, [11](#)
 UP2SYN, [11](#)
 UP2TAB, [11](#)
LOGALL
 log.h, [11](#)

pc
 log.c, [6](#)
 log.h, [13](#)
pce

- log.c, [6](#)
 - log.h, [13](#)
- pp
 - log.c, [7](#)
 - log.h, [13](#)
- SER
 - log.h, [11](#)
- splitFileName
 - log.c, [7](#)
- SYN
 - log.h, [11](#)
- TAB
 - log.h, [11](#)
- TER
 - log.h, [11](#)
- UP2SYN
 - log.h, [11](#)
- UP2TAB
 - log.h, [11](#)