Progres

Gerado por Doxygen 1.9.4

Capítulo 1

Índice das estruturas de dados

1.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Capítulo 2

Índice dos arquivos

2.1 Lista de arquivos

Lista de todos os arquivos com uma breve descrição:

Capítulo 3

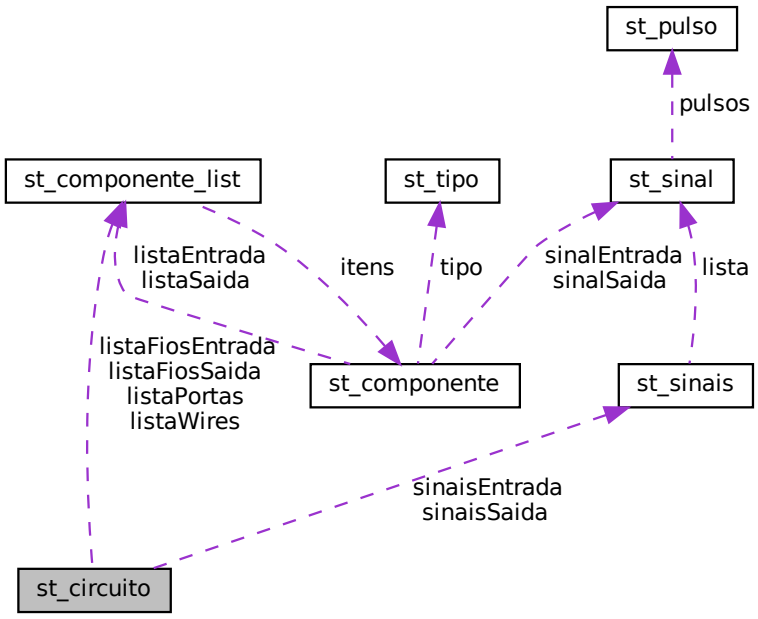
Documentação da estruturas de dados

3.1 Referência à estrutura st_circuito

Estrutura que representa um circuito, mais especificamente um 'module'.

`#include <estruturas.h>`

Diagrama de colaboração para st_circuito:



Campos de Dados

- ListaComponente * listaFiosEntrada
- Sinais * sinaisEntrada
- ListaComponente * listaFiosSaida
- Sinais * sinaisSaida
- ListaComponente * listaWires
- ListaComponente * listaPortas

3.1.1 Descrição detalhada

Estrutura que representa um circuito, mais especificamente um 'module'.

3.1.2 Documentação dos campos e atributos

3.1.2.1 listaFiosEntrada

```
ListaComponente* st_circuito::listaFiosEntrada
```

3.1.2.2 listaFiosSaida

```
ListaComponente* st_circuito::listaFiosSaida
```

3.1.2.3 listaPortas

```
ListaComponente* st_circuito::listaPortas
```

3.1.2.4 listaWires

```
ListaComponente* st_circuito::listaWires
```

3.1.2.5 sinaisEntrada

```
Sinais* st_circuito::sinaisEntrada
```

3.1.2.6 sinaisSaida

```
Sinais* st_circuito::sinaisSaida
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

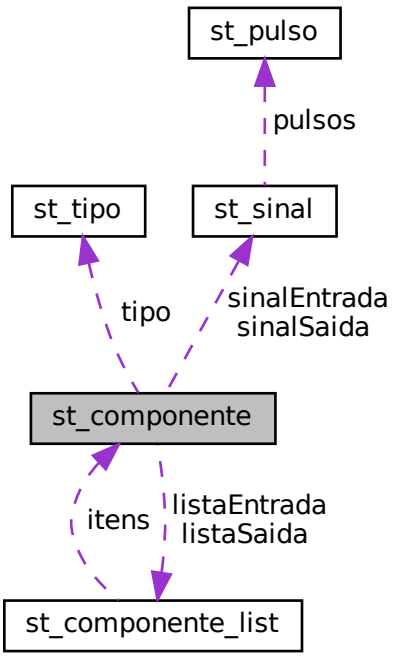
- estruturas.h

3.2 Referência à estrutura st_componente

Estrutura que representa um componente do circuito (uma porta lógica)

`#include <estruturas.h>`

Diagrama de colaboração para st_componente:



Campos de Dados

- char nome [16]
- t_tipo tipo
- ListaComponente * listaEntrada
- Sinal * sinalEntrada
- ListaComponente * listaSaida
- Sinal * sinalSaida
- ValorLogico valorDinamico

3.2.1 Descrição detalhada

Estrutura que representa um componente do circuito (uma porta lógica)

3.2.2 Documentação dos campos e atributos

3.2.2.1 listaEntrada

```
ListaComponente* st_componente::listaEntrada
```

3.2.2.2 listaSaida

```
ListaComponente* st_componente::listaSaida
```

3.2.2.3 nome

```
char st_componente::nome[16]
```

3.2.2.4 sinalEntrada

```
Sinal* st_componente::sinalEntrada
```

3.2.2.5 sinalSaida

```
Sinal* st_componente::sinalSaida
```

3.2.2.6 tipo

```
t_tipo st_componente::tipo
```

3.2.2.7 valorDinamico

```
ValorLogico st_componente::valorDinamico
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

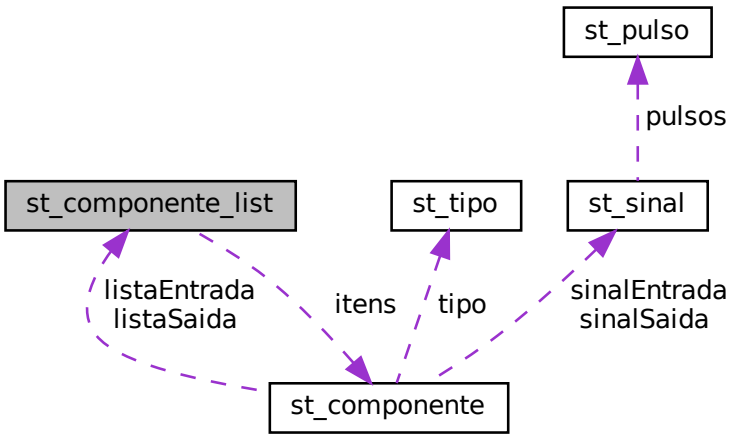
- estruturas.h

3.3 Referência à estrutura st_componente_list

Estrutura que representa uma lista de componentes. Na verdade ela guarda o total e um array de ponteiros para as portas.

`#include <estruturas.h>`

Diagrama de colaboração para st_componente_list:



Campos de Dados

- int tamanho
- Componente * itens

3.3.1 Descrição detalhada

Estrutura que representa uma lista de componentes. Na verdade ela guarda o total e um array de ponteiros para as portas.

3.3.2 Documentação dos campos e atributos

3.3.2.1 itens

```
Componente* st_componente_list::itens
```

3.3.2.2 tamanho

```
int st_componente_list::tamanho
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

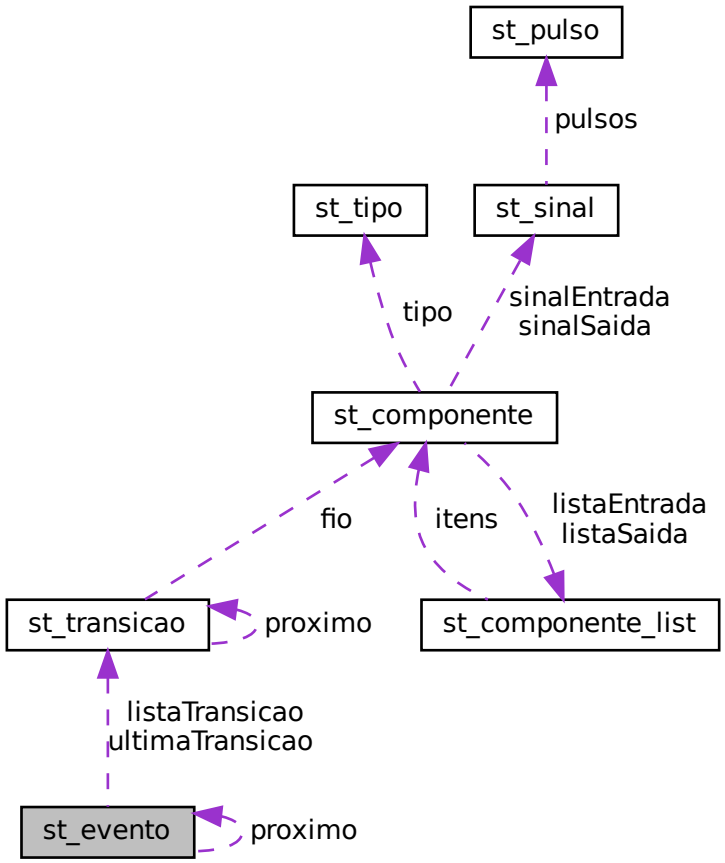
- estruturas.h

3.4 Referência à estrutura st_evento

Estrutura para um evento. Um lista é formada pelo encadeamento desses eventos. Como toda inserção nessa lista é ordenada por tempo, temos assim uma fila de eventos. Esta fila é referenciada pelo seu primeiro elemento, e temos que o último precede um NULL.

```
#include <eventos.h>
```

Diagrama de colaboração para st_evento:



Campos de Dados

- Tempo quando
- Transicao * listaTransicao
- Transicao * ultimaTransicao
- Evento * proximo

3.4.1 Descrição detalhada

Estrutura para um evento. Um lista é formada pelo encadeamento desses eventos. Como toda inserção nessa lista é ordenada por tempo, temos assim uma fila de eventos. Esta fila é referenciada pelo seu primeiro elemento, e temos que o último precede um NULL.

3.4.2 Documentação dos campos e atributos

3.4.2.1 listaTransicao

```
Transicao* st_evento::listaTransicao
```

3.4.2.2 proximo

```
Evento* st_evento::proximo
```

3.4.2.3 quando

```
Tempo st_evento::quando
```

3.4.2.4 ultimaTransicao

```
Transicao* st_evento::ultimaTransicao
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

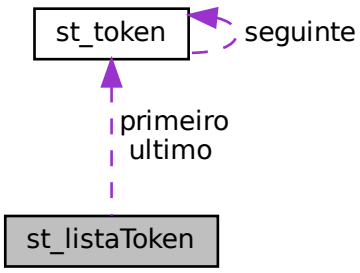
- eventos.h

3.5 Referência à estrutura st_listaToken

Tipo para uma lista encadeada de Tokens.

`#include <lex.h>`

Diagrama de colaboração para st_listaToken:



Campos de Dados

- Token * primeiro
- Token * ultimo
- int tamanho

3.5.1 Descrição detalhada

Tipo para uma lista encadeada de Tokens.

3.5.2 Documentação dos campos e atributos

3.5.2.1 primeiro

```
Token* st_listaToken::primeiro
```

3.5.2.2 tamanho

```
int st_listaToken::tamanho
```

3.5.2.3 ultimo

```
Token* st_listaToken::ultimo
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- lex.h

3.6 Referência à estrutura st_pulso

Um pulso de valor fixo e duração definida.

```
#include <sinais.h>
```

Campos de Dados

- ValorLogico valor
- Tempo tempo
- UnidTempo unidade

3.6.1 Descrição detalhada

Um pulso de valor fixo e duração definida.

3.6.2 Documentação dos campos e atributos

3.6.2.1 tempo

```
Tempo st_pulso::tempo
```

3.6.2.2 unidade

```
UnidTempo st_pulso::unidade
```

3.6.2.3 valor

`ValorLogico st_pulso::valor`

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

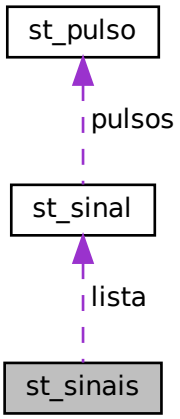
- sinais.h

3.7 Referência à estrutura st_sinais

Um conjunto de um ou mais sinais. Podem ser todos de entrada ou todos de saída.

`#include <sinais.h>`

Diagrama de colaboração para st_sinais:



Campos de Dados

- int quantidade
- Sinal * lista

3.7.1 Descrição detalhada

Um conjunto de um ou mais sinais. Podem ser todos de entrada ou todos de saída.

3.7.2 Documentação dos campos e atributos

3.7.2.1 lista

```
Sinal* st_sinais::lista
```

3.7.2.2 quantidade

```
int st_sinais::quantidade
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

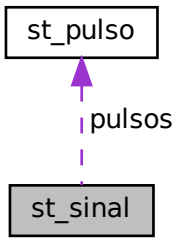
- sinais.h

3.8 Referência à estrutura st_sinal

Um sinal contém um array de pulsos com o último pulso nulo. Semelhantemente a uma string.

```
#include <sinais.h>
```

Diagrama de colaboração para st_sinal:



Campos de Dados

- char nome [MAX_NOME_SINAL]
- Pulso * pulsos
- Tempo duracaoTotal

3.8.1 Descrição detalhada

Um sinal contém um array de pulsos com o último pulso nulo. Semelhantemente a uma string.

3.8.2 Documentação dos campos e atributos

3.8.2.1 duracaoTotal

```
Tempo st_sinal::duracaoTotal
```

3.8.2.2 nome

```
char st_sinal::nome[MAX_NOME_SINAL]
```

3.8.2.3 pulsos

```
Pulso* st_sinal::pulsos
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- sinais.h

3.9 Referência à estrutura st_tipo

Estrutura que define a porta. Qual sua função lógica e seu delay.

```
#include <estruturas.h>
```

Campos de Dados

- t_operador operador
- Tempo atraso

3.9.1 Descrição detalhada

Estrutura que define a porta. Qual sua função lógica e seu delay.

3.9.2 Documentação dos campos e atributos

3.9.2.1 atraso

```
Tempo st_tipo::atraso
```

3.9.2.2 operador

```
t_operador st_tipo::operador
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

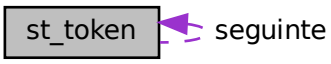
- estruturas.h

3.10 Referência à estrutura st_token

Tipo basico para o elemento que representa um token.

```
#include <lex.h>
```

Diagrama de colaboração para st_token:



Campos de Dados

- char valor [MAX_TOKEN_SIZE]
- int linha
- int coluna
- GrupoToken tipo
- struct st_token * seguinte

3.10.1 Descrição detalhada

Tipo basico para o elemento que representa um token.

3.10.2 Documentação dos campos e atributos

3.10.2.1 coluna

```
int st_token::coluna
```

3.10.2.2 linha

```
int st_token::linha
```

3.10.2.3 seguinte

```
struct st_token* st_token::seguinte
```

3.10.2.4 tipo

```
GrupoToken st_token::tipo
```

3.10.2.5 valor

```
char st_token::valor[MAX_TOKEN_SIZE]
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

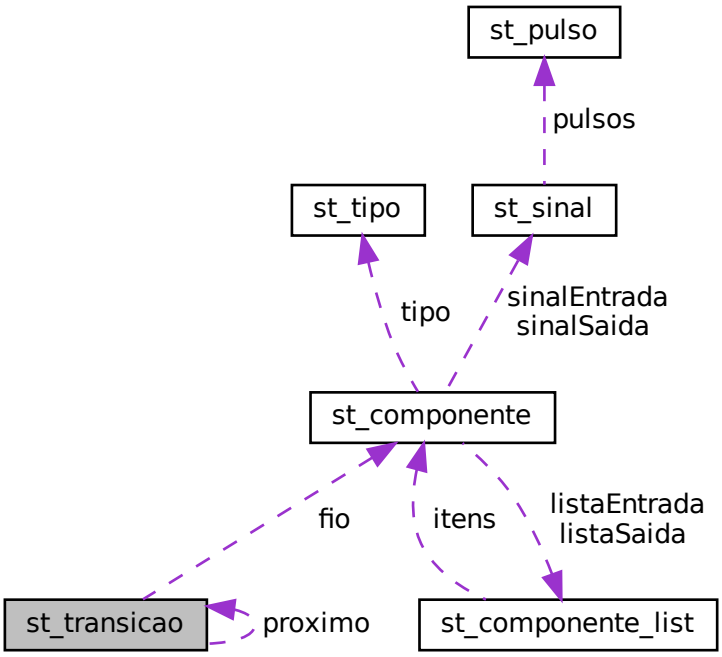
- lex.h

3.11 Referência à estrutura st_transicao

Estrutura que representa uma transição do valor lógico de um fio. Um lista é formada pelo encadeamento dessas transições. Esta lista é referenciada pelo seu primeiro elemento, e temos que o último precede um NULL.

`#include <eventos.h>`

Diagrama de colaboração para st_transicao:



Campos de Dados

- Componente fio
- ValorLogico novoValor
- Transicao * proximo

3.11.1 Descrição detalhada

Estrutura que representa uma transição do valor lógico de um fio. Um lista é formada pelo encadeamento dessas transições. Esta lista é referenciada pelo seu primeiro elemento, e temos que o último precede um NULL.

3.11.2 Documentação dos campos e atributos

3.11.2.1 fio

```
Componente st_transicao::fio
```

3.11.2.2 novoValor

```
ValorLogico st_transicao::novoValor
```

3.11.2.3 proximo

```
Transicao* st_transicao::proximo
```

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- eventos.h

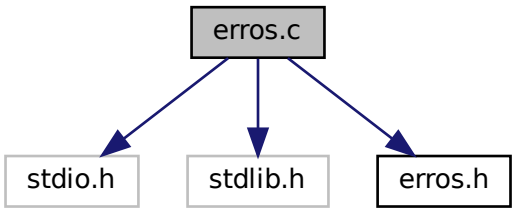
Capítulo 4

Documentação do arquivo

4.1 Referência ao arquivo erros.c

```
#include <stdio.h>
#include <stdlib.h>
#include "erros.h"
```

Diagrama de dependências de inclusão para erros.c:



Funções

- void ∗ exibeMsgErro (char ∗msg, int linha, int coluna, char ∗esperado, char ∗encontrado)

  Exibe na saida padrao, uma mensagem de erro relativa a analise lexica ou sintatica do arquivo fonte em questao.
- void erroFatalMemoria ()

  Exibe uma mensagem de erro por falta de memória e encerra o programa.

4.1.1 Documentação das funções

4.1.1.1 erroFatalMemoria()

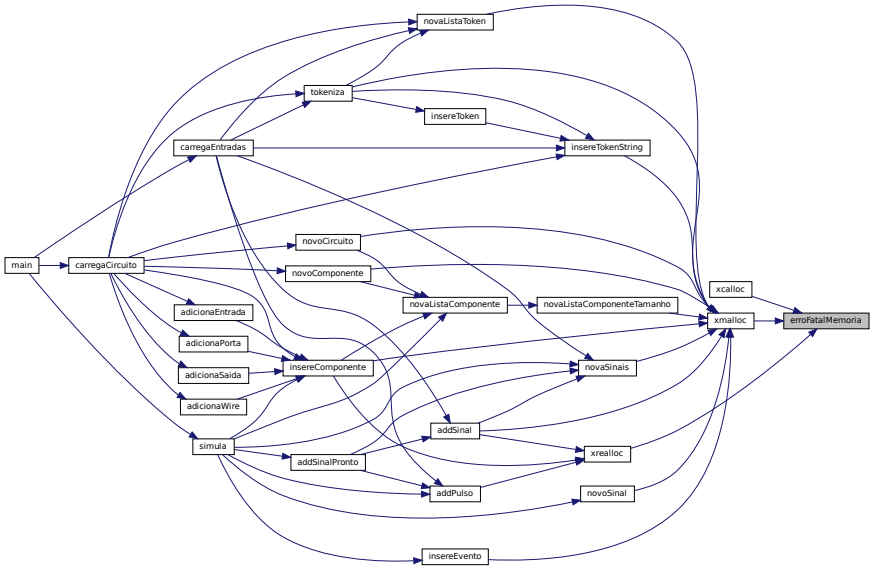
```
void erroFatalMemoria ( )
```

Exibe uma mensagem de erro por falta de memória e encerra o programa.

Retorna

Void.

Este é o diagrama das funções que utilizam esta função:



4.1.1.2 exibeMsgErro()

```
void * exibeMsgErro (
            char * msg,
            int linha,
            int coluna,
            char * esperado,
            char * encontrado )
```

Exibe na saida padrao, uma mensagem de erro relativa a analise lexica ou sintatica do arquivo fonte em questao.

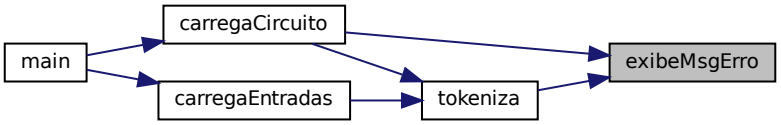
Parâmetros

msg	O texto da mensagem de erro a ser exibida.
linha	A linha onde ocorre o erro no fonte. -1 para omitir.
coluna	A coluna onde ocorre o erro no fonte. -1 para omitir.
esperado	Uma string com o que era esperado. NULL para omitir.
encontrado	Uma string com o que foi encontrado no lugar. NULL para omitir.

Retorna

Um NULL, sempre.

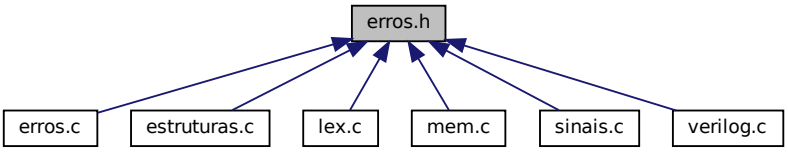
Este é o diagrama das funções que utilizam esta função:



4.2 Referência ao arquivo erros.h

Gerenciamento de mensagens de erros, ou casos de erro.

Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Funções

- void ∗ exibeMsgErro (char ∗msg, int linha, int coluna, char ∗esperado, char ∗encontrado)

  Exibe na saida padrao, uma mensagem de erro relativa a analise lexica ou sintatica do arquivo fonte em questao.
- void erroFatalMemoria ()

  Exibe uma mensagem de erro por falta de memória e encerra o programa.

4.2.1 Descrição detalhada

Gerenciamento de mensagens de erros, ou casos de erro.

4.2.2 Documentação das funções

4.2.2.1 erroFatalMemoria()

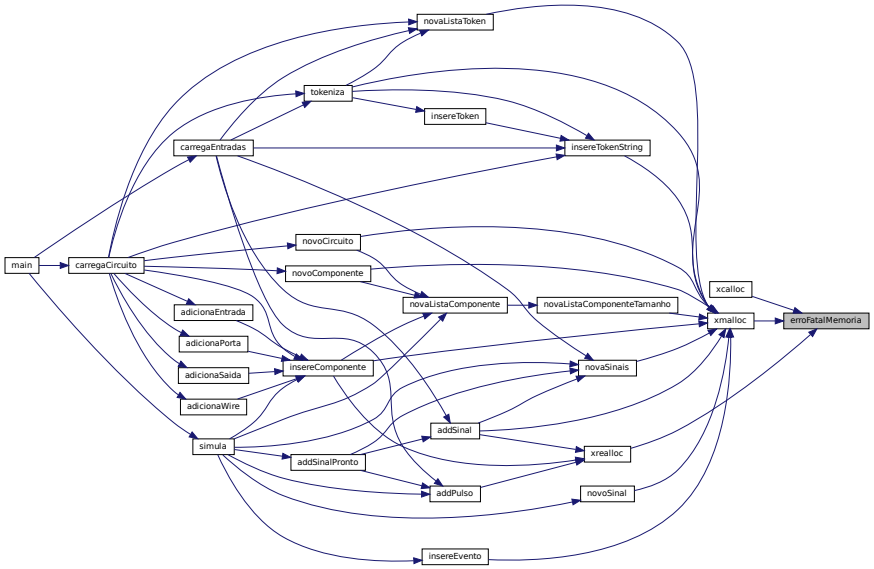
```
void erroFatalMemoria ( )
```

Exibe uma mensagem de erro por falta de memória e encerra o programa.

Retorna

Void.

Este é o diagrama das funções que utilizam esta função:



4.2.2.2 exibeMsgErro()

```
void * exibeMsgErro (
            char * msg,
            int linha,
            int coluna,
            char * esperado,
            char * encontrado )
```

Exibe na saida padrao, uma mensagem de erro relativa a analise lexica ou sintatica do arquivo fonte em questao.

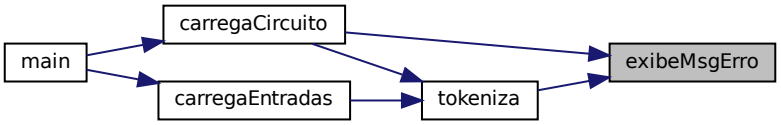
Parâmetros

msg	O texto da mensagem de erro a ser exibida.
linha	A linha onde ocorre o erro no fonte. -1 para omitir.
coluna	A coluna onde ocorre o erro no fonte. -1 para omitir.
esperado	Uma string com o que era esperado. NULL para omitir.
encontrado	Uma string com o que foi encontrado no lugar. NULL para omitir.

Retorna

Um NULL, sempre.

Este é o diagrama das funções que utilizam esta função:



4.3 erros.h

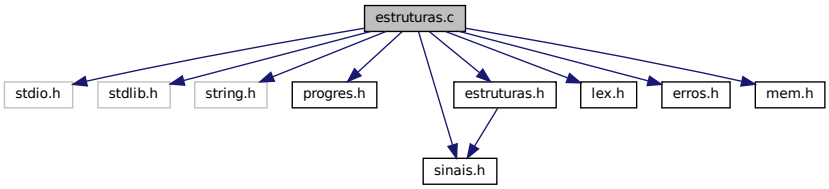
Ir para a documentação deste arquivo.

```
1
6 #ifndef ERROS_H
7
8 #define ERROS_H
9
19 void* exibeMsgErro(char* msg, int linha, int coluna, char* esperado, char *encontrado);
20
24 void erroFatalMemoria();
25
26 #endif // ERROS_H
```

4.4 Referência ao arquivo estruturas.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "progres.h"
#include "sinais.h"
#include "estruturas.h"
#include "lex.h"
#include "erros.h"
#include "mem.h"
```

Diagrama de dependências de inclusão para estruturas.c:



Funções

- t_circuito * novoCircuito ()

  Inicialização de uma estrutura de circuito.
- void adicionaEntrada (t_circuito *circ, Componente comp)

  Adiciona a entrada representada por comp à lista de fios de entrada do circuito.
- void adicionaSaida (t_circuito *circ, Componente comp)

  Adiciona a saída representada por comp à lista de fios de saída do circuito.
- void adicionaWire (t_circuito *circ, Componente comp)

  Adiciona o fio representada por comp à lista de fios (wires) do circuito.
- void adicionaPorta (t_circuito *circ, Componente comp)

  Adiciona a porta lógica representada por comp à lista de portas do circuito.
- ListaComponente * novaListaComponente ()

  Inicializa a estrutura de lista de componentes vazia.
- ListaComponente * novaListaComponenteTamanho (int tamanho)

  Inicializa a estrutura de lista de componentes com o tamanho indicado.
- void insereComponente (ListaComponente *ls, Componente cp)

  Insere o componente na lista de componentes.
- int contemComponente (ListaComponente *ls, Componente cp)

  Retorna verdadeiro se o componente indicado está contido na lista.
- Componente novoComponente (char *nome, t_operador porta)

  Inicialização de uma estrutura de componente.
- Componente getComponenteItemPorNome (ListaComponente *ls, char *nome)

  Retorna o componente da lista indica que possui o referido nome, se houver.
- Componente getPortaPorNome (t_circuito *circ, char *nome)

  Retorna a porta que tem o nome indicado, se houver na lista de portas do circuito.
- Componente getWirePorNome (t_circuito *circ, char *nome)

  Retorna o wire que tem o nome indicado, se houver.
- Componente getInputPorNome (t_circuito *circ, char *nome)

  Retorna a entrada que tem o nome indicado, se houver na lista de fios de entrada do circuito.
- Componente getOutputPorNome (t_circuito *circ, char *nome)

  Retorna a saída que tem o nome indicado, se houver na lista de fios de saída do circuito.

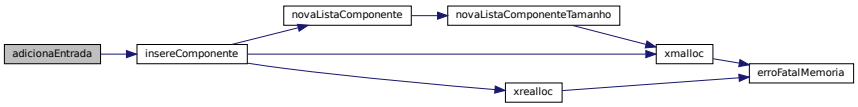
4.4.1 Documentação das funções

4.4.1.1 adicionaEntrada()

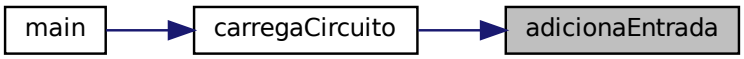
```
void adicionaEntrada (
            t_circuito * circ,
            Componente comp )
```

Adiciona a entrada representada por comp à lista de fios de entrada do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

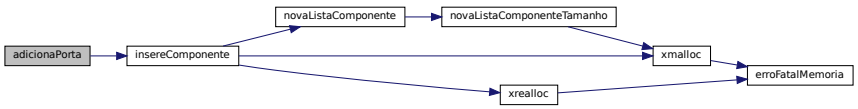


4.4.1.2 adicionaPorta()

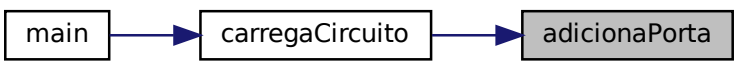
```
void adicionaPorta (
            t_circuito * circ,
            Componente comp )
```

Adiciona a porta lógica representada por comp à lista de portas do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

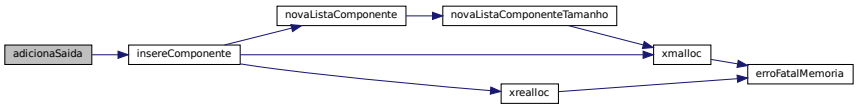


4.4.1.3 adicionaSaida()

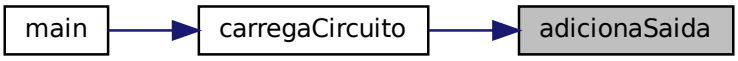
```
void adicionaSaida (
            t_circuito * circ,
            Componente comp )
```

Adiciona a saída representada por comp à lista de fios de saída do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

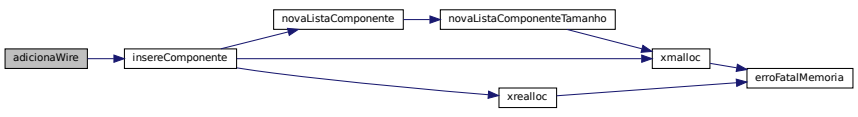


4.4.1.4 adicionaWire()

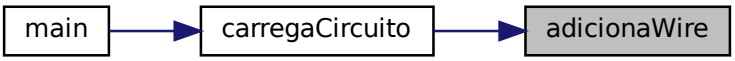
```
void adicionaWire (
            t_circuito * circ,
            Componente comp )
```

Adiciona o fio representada por comp à lista de fios (wires) do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.4.1.5 contemComponente()

```
int contemComponente (
            ListaComponente * ls,
            Componente cp )
```

Retorna verdadeiro se o componente indicado está contido na lista.

Este é o diagrama das funções que utilizam esta função:

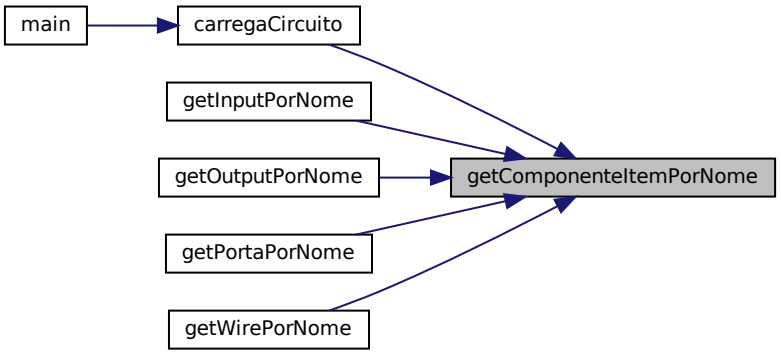
4.4.1.6 getComponenteItemPorNome()

```
Componente getComponenteItemPorNome (
            ListaComponente * ls,
            char * nome )
```

Retorna o componente da lista indica que possui o referido nome, se houver.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:

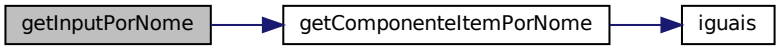


4.4.1.7 getInputPorNome()

```
Componente getInputPorNome (
            t_circuito * circ,
            char * nome )
```

Retorna a entrada que tem o nome indicado, se houver na lista de fios de entrada do circuito.

Grafo de chamadas desta função:

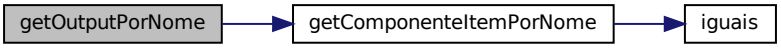


4.4.1.8 getOutputPorNome()

```
Componente getOutputPorNome (
            t_circuito * circ,
            char * nome )
```

Retorna a saída que tem o nome indicado, se houver na lista de fios de saída do circuito.

Grafo de chamadas desta função:

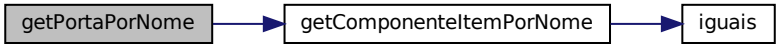


4.4.1.9 getPortaPorNome()

```
Componente getPortaPorNome (
            t_circuito * circ,
            char * nome )
```

Retorna a porta que tem o nome indicado, se houver na lista de portas do circuito.

Grafo de chamadas desta função:

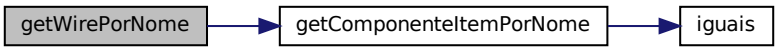


4.4.1.10 getWirePorNome()

```
Componente getWirePorNome (
            t_circuito * circ,
            char * nome )
```

Retorna o wire que tem o nome indicado, se houver.

Grafo de chamadas desta função:

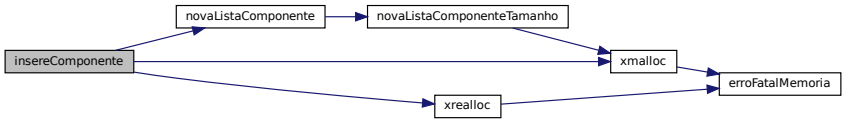


4.4.1.11 insereComponente()

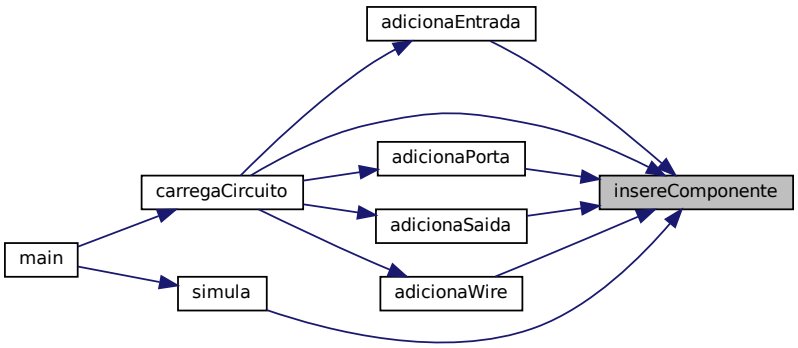
```
void insereComponente (
            ListaComponente * ls,
            Componente cp )
```

Insere o componente na lista de componentes.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

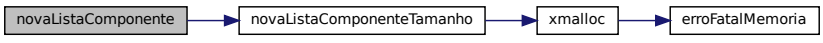


4.4.1.12 novaListaComponente()

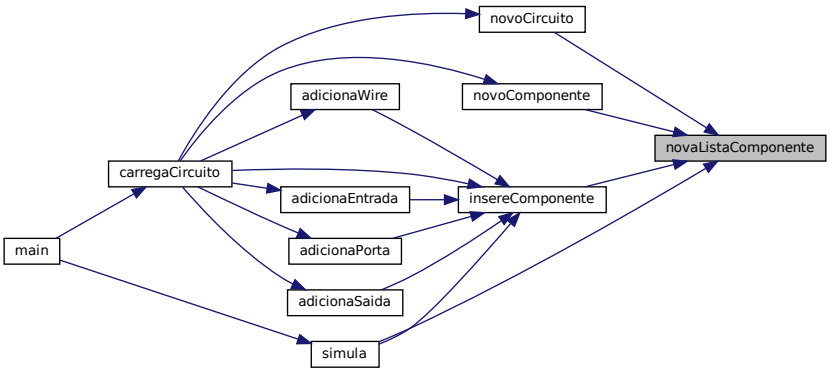
```
ListaComponente * novaListaComponente ( )
```

Inicializa a estrutura de lista de componentes vazia.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

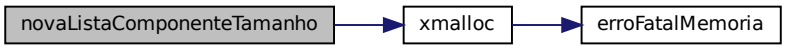


4.4.1.13 novaListaComponenteTamanho()

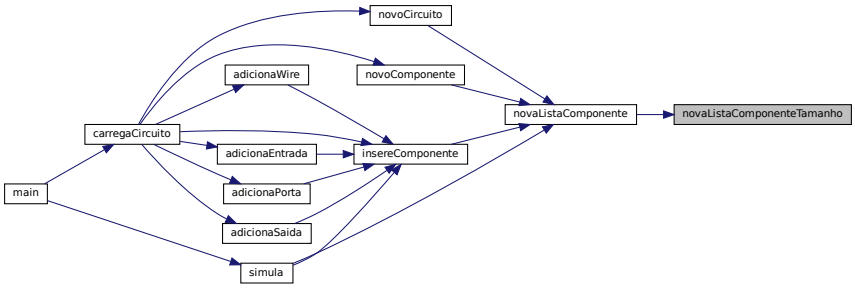
```
ListaComponente * novaListaComponenteTamanho (
            int tamanho )
```

Inicializa a estrutura de lista de componentes com o tamanho indicado.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

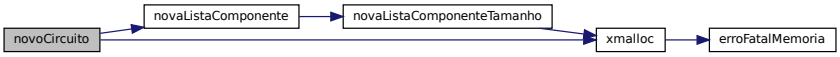


4.4.1.14 novoCircuito()

```
t_circuito * novoCircuito ( )
```

Inicialização de uma estrutura de circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

4.4.1.15 novoComponente()

```
Componente novoComponente (
            char * nome,
            t_operador porta )
```

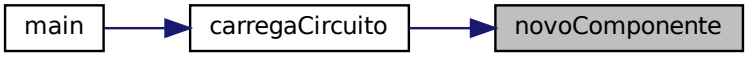
Inicialização de uma estrutura de componente.

Retorna

Um tipo Componente que é um ponteiro para a struc.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:

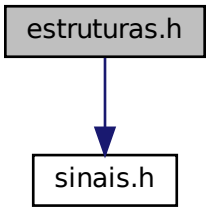


4.5 Referência ao arquivo estruturas.h

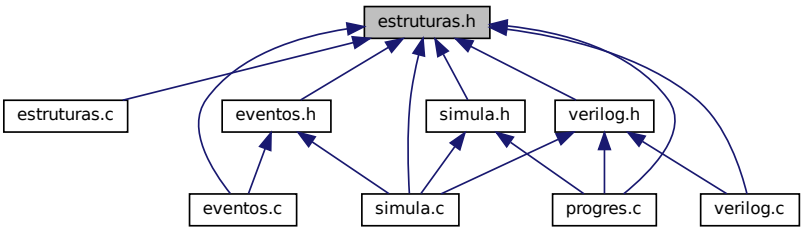
Prototipos das estruturas de dados chave do simulador.

```
#include "sinais.h"
```

Diagrama de dependências de inclusão para estruturas.h:



Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Estruturas de Dados

- struct st_tipo

  Estrutura que define a porta. Qual sua função lógica e seu delay.
- struct st_componente

  Estrutura que representa um componente do circuito (uma porta lógica)
- struct st_componente_list

  Estrutura que representa uma lista de componentes. Na verdade ela guarda o total e um array de ponteiros para as portas.
- struct st_circuito

  Estrutura que representa um circuito, mais especificamente um 'module'.

Definições de tipos

- typedef enum en_operador t_operador

  Enumeração para o definir as classes de componente do circuito de acordo com suas funções.
- typedef struct st_tipo t_tipo

  Estrutura que define a porta. Qual sua função lógica e seu delay.
- typedef struct st_componente_list ListaComponente
- typedef struct st_componente * Componente
- typedef struct st_circuito t_circuito

  Estrutura que representa um circuito, mais especificamente um 'module'.

Enumerações

- enum en_operador {
  op_and , op_or , op_xor , op_nand ,
  op_nor , op_xnor , op_not , op_buf ,
  wire , output , input }

  Enumeração para o definir as classes de componente do circuito de acordo com suas funções.

Funções

- t_circuito * novoCircuito ()

  Inicialização de uma estrutura de circuito.
- void adicionaEntrada (t_circuito *circ, Componente comp)

  Adiciona a entrada representada por comp à lista de fios de entrada do circuito.
- void adicionaSaida (t_circuito *circ, Componente comp)

  Adiciona a saída representada por comp à lista de fios de saída do circuito.
- void adicionaWire (t_circuito *circ, Componente comp)

  Adiciona o fio representada por comp à lista de fios (wires) do circuito.
- void adicionaPorta (t_circuito *circ, Componente comp)

  Adiciona a porta lógica representada por comp à lista de portas do circuito.
- Componente getPortaPorNome (t_circuito *circ, char *nome)

  Retorna a porta que tem o nome indicado, se houver na lista de portas do circuito.
- Componente getWirePorNome (t_circuito *circ, char *nome)

  Retorna o wire que tem o nome indicado, se houver.
- Componente getInputPorNome (t_circuito *circ, char *nome)

  Retorna a entrada que tem o nome indicado, se houver na lista de fios de entrada do circuito.
- Componente getOutputPorNome (t_circuito *circ, char *nome)

  Retorna a saída que tem o nome indicado, se houver na lista de fios de saída do circuito.
- Componente novoComponente (char *nome, t_operador porta)

  Inicialização de uma estrutura de componente.
- ListaComponente * novaListaComponente ()

  Inicializa a estrutura de lista de componentes vazia.
- ListaComponente * novaListaComponenteTamanho (int tamanho)

  Inicializa a estrutura de lista de componentes com o tamanho indicado.
- void insereComponente (ListaComponente *ls, Componente cp)

  Insere o componente na lista de componentes.
- int contemComponente (ListaComponente *ls, Componente cp)

  Retorna verdadeiro se o componente indicado está contido na lista.
- Componente getComponenteItemPorNome (ListaComponente *ls, char *nome)

  Retorna o componente da lista indica que possui o referido nome, se houver.

4.5.1 Descrição detalhada

Prototipos das estruturas de dados chave do simulador.

4.5.2 Documentação dos tipos

4.5.2.1 Componente

```
typedef struct st_componente* Componente
```

4.5.2.2 ListaComponente

```
typedef struct st_componente_list ListaComponente
```

4.5.2.3 t_circuito

```
typedef struct st_circuito t_circuito
```

Estrutura que representa um circuito, mais especificamente um 'module'.

4.5.2.4 t_operador

```
typedef enum en_operador t_operador
```

Enumeração para o definir as classes de componente do circuito de acordo com suas funções.

4.5.2.5 t_tipo

```
typedef struct st_tipo t_tipo
```

Estrutura que define a porta. Qual sua função lógica e seu delay.

4.5.3 Documentação dos valores da enumeração

4.5.3.1 en_operador

```
enum en_operador
```

Enumeração para o definir as classes de componente do circuito de acordo com suas funções.

Valores de enumerações

op_and	
op_or	
op_xor	
op_nand	
op_nor	
op_xnor	
op_not	
op_buf	
wire	
output	
input	

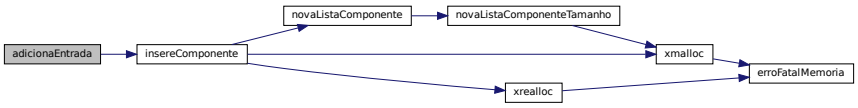
4.5.4 Documentação das funções

4.5.4.1 adicionaEntrada()

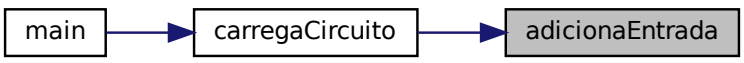
```
void adicionaEntrada (
            t_circuito * circ,
            Componente comp )
```

Adiciona a entrada representada por comp à lista de fios de entrada do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

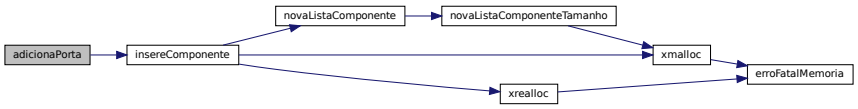


4.5.4.2 adicionaPorta()

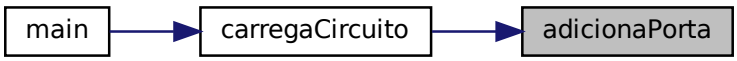
```
void adicionaPorta (
            t_circuito * circ,
            Componente comp )
```

Adiciona a porta lógica representada por comp à lista de portas do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

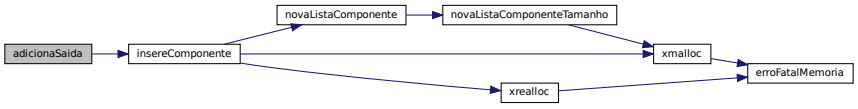


4.5.4.3 adicionaSaida()

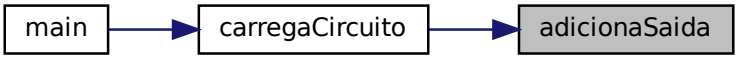
```
void adicionaSaida (
            t_circuito * circ,
            Componente comp )
```

Adiciona a saída representada por comp à lista de fios de saída do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

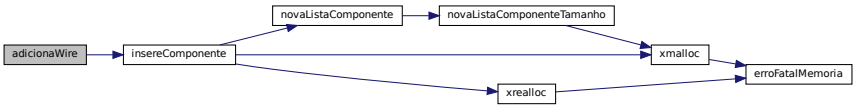


4.5.4.4 adicionaWire()

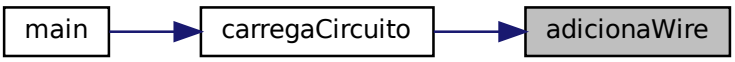
```
void adicionaWire (
            t_circuito * circ,
            Componente comp )
```

Adiciona o fio representada por comp à lista de fios (wires) do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.5.4.5 contemComponente()

```
int contemComponente (
            ListaComponente * ls,
            Componente cp )
```

Retorna verdadeiro se o componente indicado está contido na lista.

Este é o diagrama das funções que utilizam esta função:

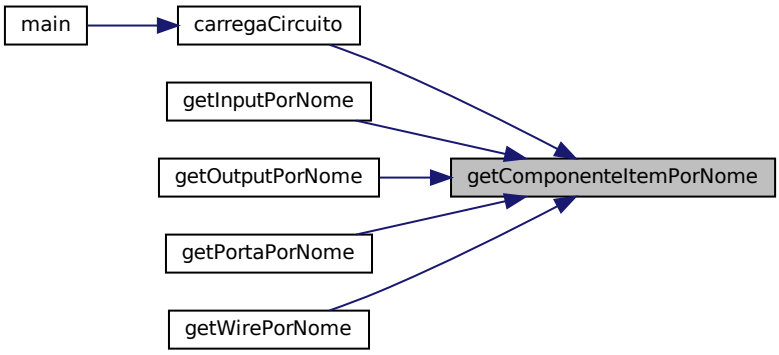
4.5.4.6 getComponenteItemPorNome()

```
Componente getComponenteItemPorNome (
            ListaComponente * ls,
            char * nome )
```

Retorna o componente da lista indica que possui o referido nome, se houver.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:

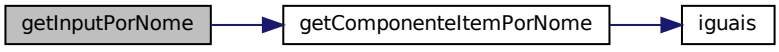


4.5.4.7 getInputPorNome()

```
Componente getInputPorNome (
            t_circuito * circ,
            char * nome )
```

Retorna a entrada que tem o nome indicado, se houver na lista de fios de entrada do circuito.

Grafo de chamadas desta função:

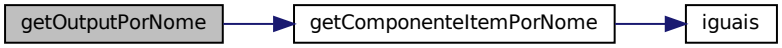


4.5.4.8 getOutputPorNome()

```
Componente getOutputPorNome (
            t_circuito * circ,
            char * nome )
```

Retorna a saída que tem o nome indicado, se houver na lista de fios de saída do circuito.

Grafo de chamadas desta função:

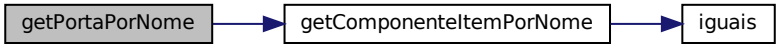


4.5.4.9 getPortaPorNome()

```
Componente getPortaPorNome (
            t_circuito * circ,
            char * nome )
```

Retorna a porta que tem o nome indicado, se houver na lista de portas do circuito.

Grafo de chamadas desta função:

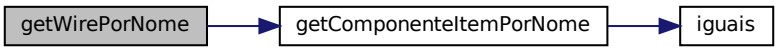


4.5.4.10 getWirePorNome()

```
Componente getWirePorNome (
            t_circuito * circ,
            char * nome )
```

Retorna o wire que tem o nome indicado, se houver.

Grafo de chamadas desta função:

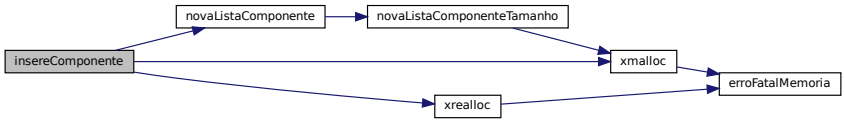


4.5.4.11 insereComponente()

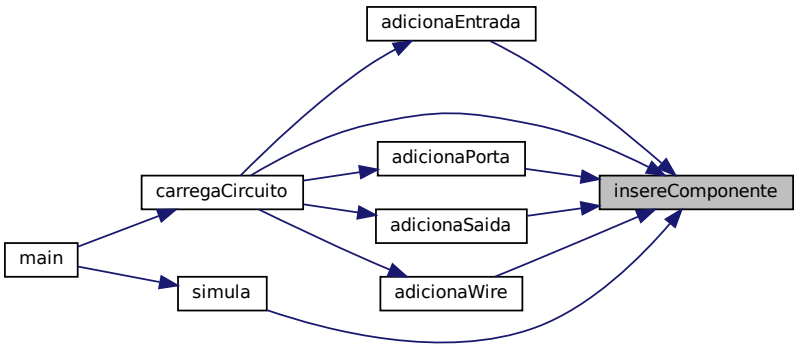
```
void insereComponente (
            ListaComponente * ls,
            Componente cp )
```

Insere o componente na lista de componentes.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



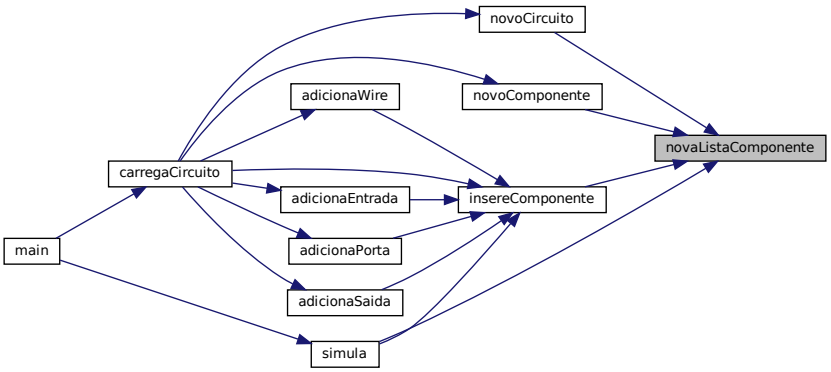
4.5.4.12 novaListaComponente()

```
ListaComponente * novaListaComponente ( )
```

Inicializa a estrutura de lista de componentes vazia.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:

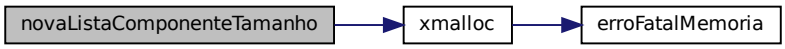


4.5.4.13 novaListaComponenteTamanho()

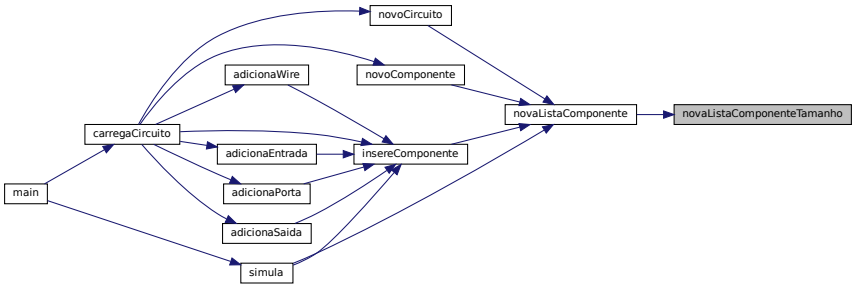
```
ListaComponente * novaListaComponenteTamanho (
            int tamanho )
```

Inicializa a estrutura de lista de componentes com o tamanho indicado.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

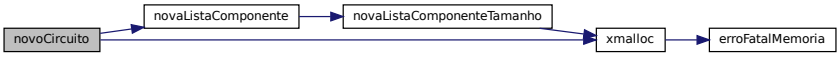


4.5.4.14 novoCircuito()

```
t_circuito * novoCircuito ( )
```

Inicialização de uma estrutura de circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

4.5.4.15 novoComponente()

```
Componente novoComponente (
            char * nome,
            t_operador porta )
```

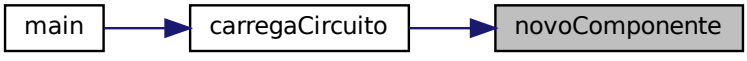
Inicialização de uma estrutura de componente.

Retorna

Um tipo Componente que é um ponteiro para a struc.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:



4.6 estruturas.h

Ir para a documentação deste arquivo.

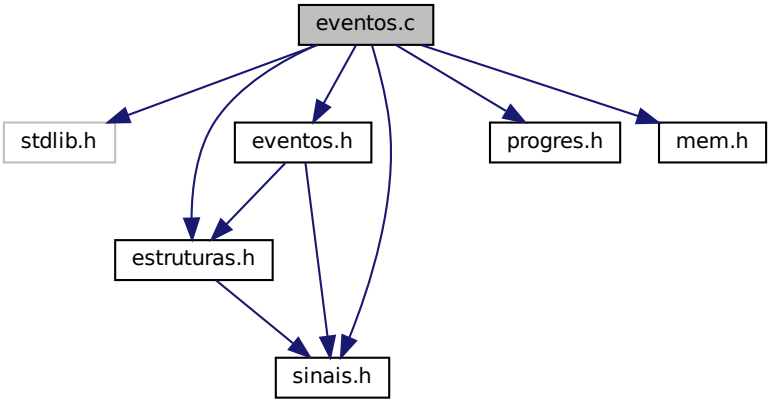
```
1
6 #ifndef ESTRUTURAS_H
7
8 #define ESTRUTURAS_H
9
10 #include "sinais.h"
11
14 typedef enum en_operador {
15     op_and,
16     op_or,
17     op_xor,
18     op_nand,
19     op_nor,
20     op_xnor,
21     op_not,
22     op_buf,
23     wire,
24     output,
25     input
26 } t_operador;
27
30 typedef struct st_tipo {
31     t_operador operador;
32     Tempo atraso;
33 } t_tipo;
34
35 typedef struct st_componente_list ListaComponente;
36
37 typedef struct st_componente * Componente;
38
41 struct st_componente {
42     char nome[16];
43     t_tipo tipo;
44
45     ListaComponente* listaEntrada;
46     Sinal* sinalEntrada;
47
48     ListaComponente* listaSaida;
49     Sinal* sinalSaida;
50
51     ValorLogico valorDinamico; // fio
52 };
53
57 struct st_componente_list {
58     int tamanho;
59     Componente *itens;
60 };
61
64 typedef struct st_circuito {
65     ListaComponente *listaFiosEntrada;
66     Sinais *sinaisEntrada;
67
68     ListaComponente *listaFiosSaida;
69     Sinais *sinaisSaida;
70
71     ListaComponente *listaWires;
72
73     ListaComponente *listaPortas;
74 } t_circuito;
75
78 t_circuito* novoCircuito();
79
82 void adicionaEntrada(t_circuito* circ, Componente comp);
83
86 void adicionaSaida(t_circuito* circ, Componente comp);
87
90 void adicionaWire(t_circuito* circ, Componente comp);
91
94 void adicionaPorta(t_circuito* circ, Componente comp);
95
98 Componente getPortaPorNome(t_circuito* circ, char* nome);
99
102 Componente getWirePorNome(t_circuito* circ, char* nome);
103
106 Componente getInputPorNome(t_circuito* circ, char* nome);
107
110 Componente getOutputPorNome(t_circuito* circ, char* nome);
111
115 Componente novoComponente(char* nome, t_operador porta);
116
119 ListaComponente* novaListaComponente();
120
```

```
123 ListaComponente* novaListaComponenteTamanho(int tamanho);
124
127 void insereComponente(ListaComponente* ls, Componente cp);
128
131 int contemComponente(ListaComponente* ls, Componente cp);
132
135 Componente getComponenteItemPorNome(ListaComponente* ls, char* nome);
136
137 #endif // ESTRUTURAS_H
```

4.7 Referência ao arquivo eventos.c

```
#include <stdlib.h>
#include "eventos.h"
#include "progres.h"
#include "estruturas.h"
#include "sinais.h"
#include "mem.h"
```

Diagrama de dependências de inclusão para eventos.c:



Funções

- void insereEvento (Evento **fila, Tempo t, Componente comp, ValorLogico novoValor)

  Adiciona à fila um evento no tempo t que faz a transição do valor de comp para o novoValor. Mas se houver já na fila evento marcado para t, apenas adiciona à lista de transições desse evento, a nova transição.
- Transicao * getTransicoesEm (Evento *fila, Tempo t)

  Retorna uma lista das transições que ocorrem exatamente em determinado tempo t. Se não houver evento nesse tempo t, retornará NULL.
- Transicao * popEvento (Evento **fila)

  Remove da fila o evento mais próximo e devolve a lista de transições referente.

4.7.1 Documentação das funções

4.7.1.1 getTransicoesEm()

```
Transicao * getTransicoesEm (
            Evento * fila,
            Tempo t )
```

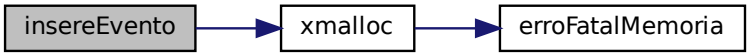
Retorna uma lista das transições que ocorrem exatamente em determinado tempo t. Se não houver evento nesse tempo t, retornará NULL.

4.7.1.2 insereEvento()

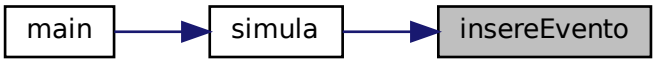
```
void insereEvento (
            Evento ** fila,
            Tempo t,
            Componente comp,
            ValorLogico novoValor )
```

Adiciona à fila um evento no tempo t que faz a transição do valor de comp para o novoValor. Mas se houver já na fila evento marcado para t, apenas adiciona à lista de transições desse evento, a nova transição.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

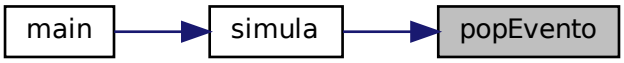


4.7.1.3 popEvento()

```
Transicao * popEvento (
            Evento ** fila )
```

Remove da fila o evento mais próximo e devolve a lista de transições referente.

Este é o diagrama das funções que utilizam esta função:

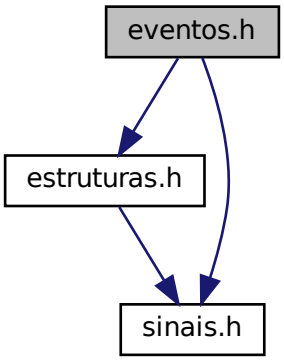


4.8 Referência ao arquivo eventos.h

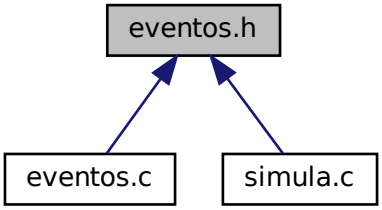
Fila de eventos usada na simulação. Não é fila FIFO, a ordem é dada pelo valor de tempo dos eventos.

```
#include "estruturas.h"
#include "sinais.h"
```

Diagrama de dependências de inclusão para eventos.h:



Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Estruturas de Dados

- struct st_transicao

  Estrutura que representa uma transição do valor lógico de um fio. Um lista é formada pelo encadeamento dessas transições. Esta lista é referenciada pelo seu primeiro elemento, e temos que o último precede um NULL.
- struct st_evento

  Estrutura para um evento. Um lista é formada pelo encadeamento desses eventos. Como toda inserção nessa lista é ordenada por tempo, temos assim uma fila de eventos. Esta fila é referenciada pelo seu primeiro elemento, e temos que o último precede um NULL.

Definições de tipos

- typedef struct st_transicao Transicao
- typedef struct st_evento Evento

Funções

- void insereEvento (Evento ∗∗fila, Tempo t, Componente comp, ValorLogico novoValor)

  Adiciona à fila um evento no tempo t que faz a transição do valor de comp para o novoValor. Mas se houver já na fila evento marcado para t, apenas adiciona à lista de transições desse evento, a nova transição.
- Transicao ∗ getTransicoesEm (Evento ∗fila, Tempo t)

  Retorna uma lista das transições que ocorrem exatamente em determinado tempo t. Se não houver evento nesse tempo t, retornará NULL.
- Transicao ∗ popEvento (Evento ∗∗fila)

  Remove da fila o evento mais próximo e devolve a lista de transições referente.

4.8.1 Descrição detalhada

Fila de eventos usada na simulação. Não é fila FIFO, a ordem é dada pelo valor de tempo dos eventos.

4.8.2 Documentação dos tipos

4.8.2.1 Evento

```
typedef struct st_evento Evento
```

4.8.2.2 Transicao

```
typedef struct st_transicao Transicao
```

4.8.3 Documentação das funções

4.8.3.1 getTransicoesEm()

```
Transicao * getTransicoesEm (
            Evento * fila,
            Tempo t )
```

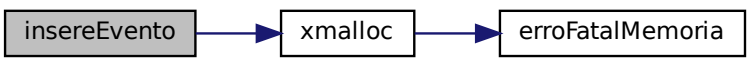
Retorna uma lista das transições que ocorrem exatamente em determinado tempo t. Se não houver evento nesse tempo t, retornará NULL.

4.8.3.2 insereEvento()

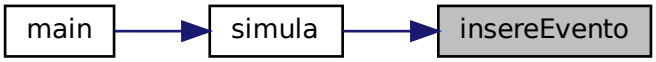
```
void insereEvento (
            Evento ** fila,
            Tempo t,
            Componente comp,
            ValorLogico novoValor )
```

Adiciona à fila um evento no tempo t que faz a transição do valor de comp para o novoValor. Mas se houver já na fila evento marcado para t, apenas adiciona à lista de transições desse evento, a nova transição.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

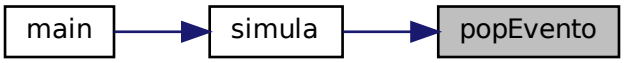


4.8.3.3 popEvento()

```
Transicao * popEvento (
            Evento ** fila )
```

Remove da fila o evento mais próximo e devolve a lista de transições referente.

Este é o diagrama das funções que utilizam esta função:



4.9 eventos.h

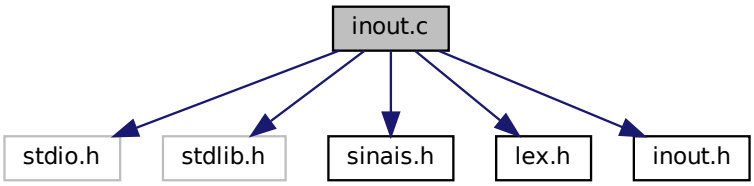
Ir para a documentação deste arquivo.

```
1
7 #ifndef EVENTOS_H
8
9 #define EVENTOS_H
10
11 #include "estruturas.h"
12 #include "sinais.h"
13
14 typedef struct st_transicao Transicao;
15
21 struct st_transicao {
22     Componente fio; // Indica o componente sobre o qual o evento se origina, apenas fios
23     ValorLogico novoValor; // Novo valor lógico a ser setado
24     Transicao* proximo;
25 };
26
27 typedef struct st_evento Evento;
28
34 struct st_evento {
35     Tempo quando; // Indica o instante de ocorrência do evento
36     Transicao* listaTransicao;
37     Transicao* ultimaTransicao;
38     Evento* proximo;
39 };
40
46 void insereEvento(Evento **fila, Tempo t, Componente comp, ValorLogico novoValor);
47
52 Transicao* getTransicoesEm(Evento* fila, Tempo t);
53
57 Transicao* popEvento(Evento **fila);
58
59 #endif // EVENTOS_H
```

4.10 Referência ao arquivo inout.c

```
#include <stdio.h>
#include <stdlib.h>
#include "sinais.h"
#include "lex.h"
#include "inout.h"
```

Diagrama de dependências de inclusão para inout.c:



Funções

- Sinais ∗ carregaEntradas (FILE ∗arquivo)

  Cria uma estrutura de dados representando todos os sinas de entrada lidos partir do arquivo de entrada correspondente (extensão ∗.in).
- void salvarSinais (Sinais ∗sinaisSaida, FILE ∗arqSaida)

  Salva todos os sinais contidos no conjunto para o arquivo de saida com a formatação padrão.

4.10.1 Documentação das funções

4.10.1.1 carregaEntradas()

```
Sinais * carregaEntradas (
            FILE * arquivo )
```

Cria uma estrutura de dados representando todos os sinas de entrada lidos partir do arquivo de entrada correspondente (extensão ∗.in).

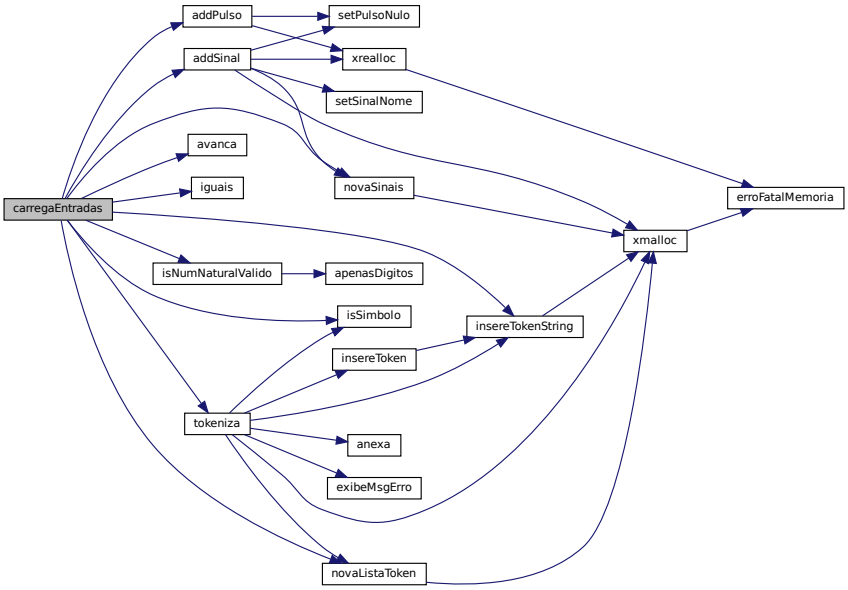
Parâmetros

arquivo	O handler do arquivo de entrada com sinais a ser processado.

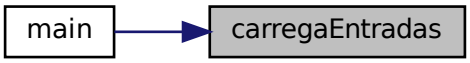
Retorna

A estrutura de dados contendo todos os sinais lidos do arquivo.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

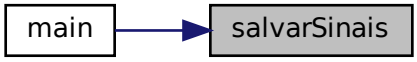


4.10.1.2 salvarSinais()

```
void salvarSinais (
            Sinais * sinaisSaida,
            FILE * arqSaida )
```

Salva todos os sinais contidos no conjunto para o arquivo de saida com a formatação padrão.

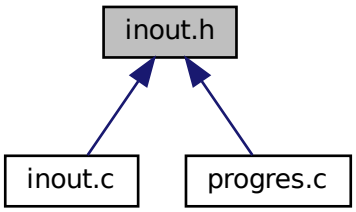
Este é o diagrama das funções que utilizam esta função:



4.11 Referência ao arquivo inout.h

Protótipos das funções de leitura e gravação dos arquivos de sinas de entrada e saída.

Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Macros

- #define MSG_ARQUIVO_ENTRADA_CORROMPIDO "Arquivo de entrada corrompido.\n"

Funções

- Sinais ∗ carregaEntradas (FILE ∗arquivo)

  Cria uma estrutura de dados representando todos os sinas de entrada lidos partir do arquivo de entrada correspondente (extensão ∗.in).
- void salvarSinais (Sinais ∗sinaisSaida, FILE ∗arqSaida)

  Salva todos os sinais contidos no conjunto para o arquivo de saida com a formatação padrão.

4.11.1 Descrição detalhada

Protótipos das funções de leitura e gravação dos arquivos de sinas de entrada e saída.

4.11.2 Documentação das macros

4.11.2.1 MSG_ARQUIVO_ENTRADA_CORROMPIDO

```
#define MSG_ARQUIVO_ENTRADA_CORROMPIDO "Arquivo de entrada corrompido.\n"
```

4.11.3 Documentação das funções

4.11.3.1 carregaEntradas()

```
Sinais * carregaEntradas (
            FILE * arquivo )
```

Cria uma estrutura de dados representando todos os sinas de entrada lidos partir do arquivo de entrada correspondente (extensão ∗.in).

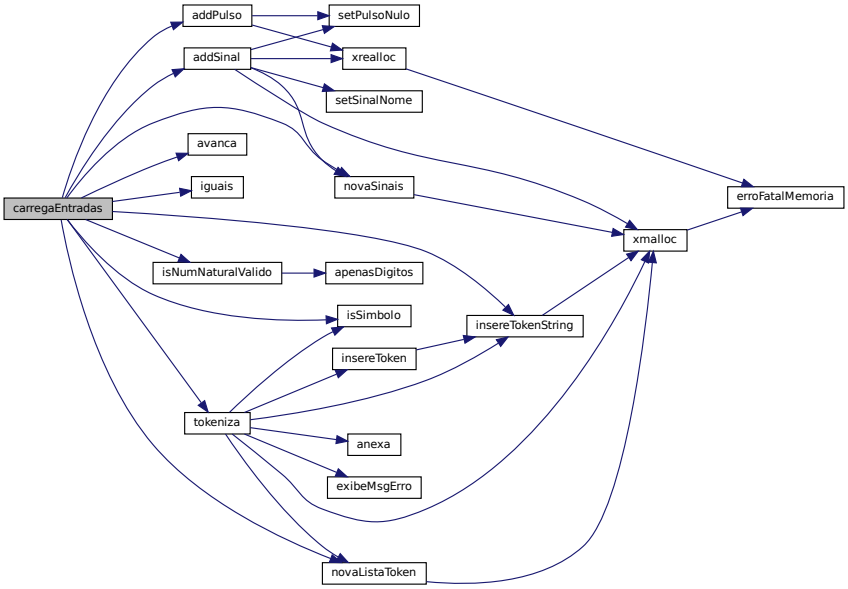
Parâmetros

arquivo	O handler do arquivo de entrada com sinais a ser processado.

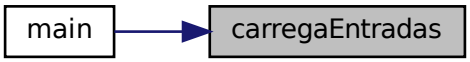
Retorna

A estrutura de dados contendo todos os sinais lidos do arquivo.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

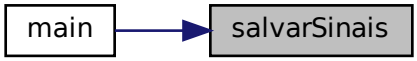


4.11.3.2 salvarSinais()

```
void salvarSinais (
            Sinais * sinaisSaida,
            FILE * arqSaida )
```

Salva todos os sinais contidos no conjunto para o arquivo de saida com a formatação padrão.

Este é o diagrama das funções que utilizam esta função:



4.12 inout.h

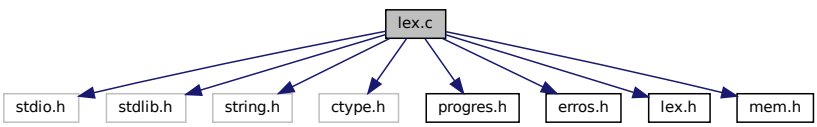
Ir para a documentação deste arquivo.

```
1
6 #ifndef INOUT_H
7
8 #define INOUT_H
9
10 #define MSG_ARQUIVO_ENTRADA_CORROMPIDO "Arquivo de entrada corrompido.\n"
11
17 Sinais* carregaEntradas(FILE *arquivo);
18
22 void salvarSinais(Sinais *sinaisSaida, FILE *arqSaida);
23
24 #endif // INOUT_H
```

4.13 Referência ao arquivo lex.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "progres.h"
#include "erros.h"
#include "lex.h"
#include "mem.h"
```

Diagrama de dependências de inclusão para lex.c:



Funções

- ListaToken * novaListaToken ()

  Inicializa uma lista vazia, i. e., com zero elementos.
- int insereToken (ListaToken *lista, char tok, int p_linha, int p_coluna)

  Insere na lista um novo token a partir de um caractere, deve-se especificar a posicao do mesmo no arquivo.
- int insereTokenString (ListaToken *lista, char *tok, int p_linha, int p_coluna)

  Insere na lista um novo token a partir de uma string, deve-se especificar a posicao do mesmo no arquivo.
- int removeTokensPorValor (ListaToken *lst, char *tok)

  Remove todos os tokens com o valor indicado da lista.
- int anexa (char *str, char c)

  Faz o apend de um char numa string qualquer.
- int isSimbolo (char c)

  Retorna verdadeiro se c for um simbolo em Verilog.
- void exibeListaDeToken (ListaToken *tokens)

  Imprime na tela os tokens dessa lista, um por linha.
- int identExiste (ListaToken *lst, char *str)

  Retorna verdadeiro se a string esta contida em algum token da lista.
- int iguais (char *a, char *b)

  Retorna verdadeiro se duas strings são iguais.
- void avanca (Token **t)

  Avanca o iterador de token para o próximo da lista.
- int isPalavra (Token *tk)

  Verifica se um token é uma palavra reservada em Verilog.
- int isIdentificador (Token *tk)

  Verifica se um token é um nome permitido de identificador.
- ListaToken * tokeniza (FILE *arquivo)

  Cria uma lista de Tokens que tem significado para o processamento sintatico, a partir do arquivo com o codigo fonte em Verilog.
- int apenasDigitos (char *str)

  Verifica se uma string contém apenas dígitos (0, 1, 2, ..., 9).
- int isNumNaturalValido (char *str)

  Verifica se uma string contém um número que pode ser convertido. Mais especeificamente, se é um natural menor que 10000.

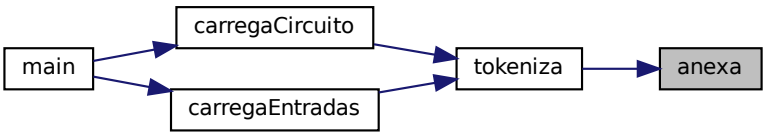
4.13.1 Documentação das funções

4.13.1.1 anexa()

```
int anexa (
            char * str,
            char c )
```

Faz o apend de um char numa string qualquer.

Este é o diagrama das funções que utilizam esta função:



4.13.1.2 apenasDigitos()

```
int apenasDigitos (
            char * str )
```

Verifica se uma string contém apenas dígitos (0, 1, 2, ..., 9).

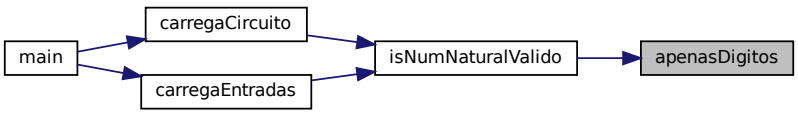
Parâmetros

str	Uma string qualquer.

Retorna

Verdadeiro se há apenas dígitos, falso na ocorrência de qualquer outro tipo de caractere.

Este é o diagrama das funções que utilizam esta função:



4.13.1.3 avanca()

```
void avanca (
            Token ** t )
```

Avanca o iterador de token para o próximo da lista.

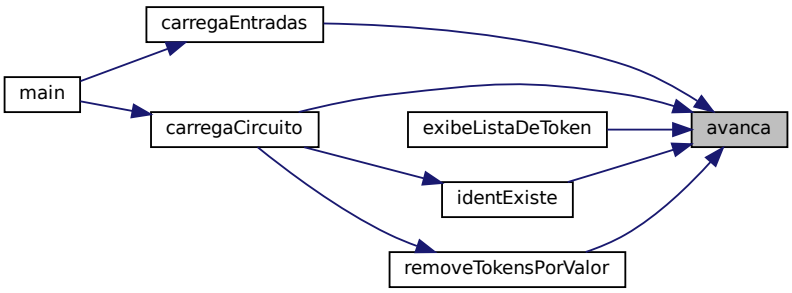
Parâmetros

t	Um ponteiro para um ponteiro de um Token.

Retorna

Void.

Este é o diagrama das funções que utilizam esta função:



4.13.1.4 exibeListaDeToken()

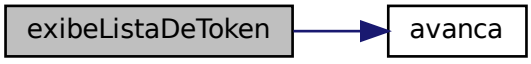
```
void exibeListaDeToken (
            ListaToken * tokens )
```

Imprime na tela os tokens dessa lista, um por linha.

Retorna

Void.

Grafo de chamadas desta função:



4.13.1.5 identExiste()

```
int identExiste (
             ListaToken * lst,
             char * str )
```

Retorna verdadeiro se a string esta contida em algum token da lista.

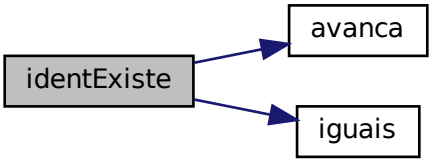
Parâmetros

lst	Uma lista de Tokens.
str	Uma string qualquer.

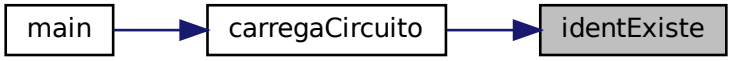
Retorna

Verdadeiro se str é o valor de algum Token em lst, falso caso contrário.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

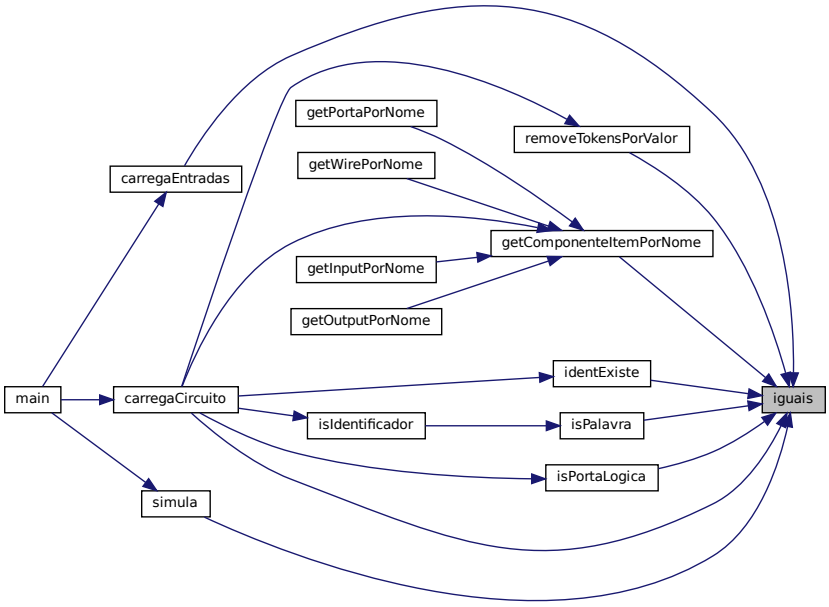


4.13.1.6 iguais()

```
int iguais (
            char * a,
            char * b )
```

Retorna verdadeiro se duas strings são iguais.

Este é o diagrama das funções que utilizam esta função:



4.13.1.7 insereToken()

```
int insereToken (
            ListaToken * lista,
            char tok,
            int p_linha,
            int p_coluna )
```

Insere na lista um novo token a partir de um caractere, deve-se especificar a posicao do mesmo no arquivo.

Parâmetros

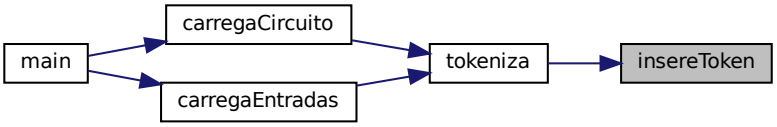
lista	Onde sera inserido o token.
tok	Um token de apenas um caractere.
p_linha	Linha no arquivo onde está o token.
p_coluna	Coluna no arquivo onde inicia-se o token.

Retorna

Verdadeiro caso sucesso, falso caso falhe.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:



4.13.1.8 insereTokenString()

```
int insereTokenString (
            ListaToken * lista,
            char * tok,
            int p_linha,
            int p_coluna )
```

Insere na lista um novo token a partir de uma string, deve-se especificar a posicao do mesmo no arquivo.

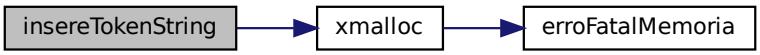
Parâmetros

lista	Onde sera inserido o token.
tok	Uma string contendo o token.
p_linha	Linha no arquivo onde está o token.
p_coluna	Coluna no arquivo onde inicia-se o token.

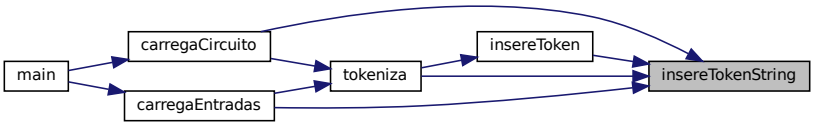
Retorna

Verdadeiro caso sucesso, falso caso falhe.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.13.1.9 isIdentificador()

```
int isIdentificador (
            Token * tk )
```

Verifica se um token é um nome permitido de identificador.

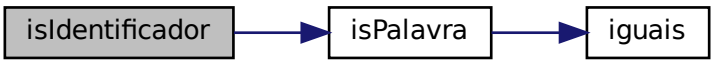
Parâmetros

str	Uma string qualquer.

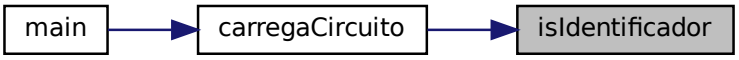
Retorna

Verdadeiro se o valor do token for um nome permitido de identificador, falso c.c.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.13.1.10 isNumNaturalValido()

```
int isNumNaturalValido (
            char * str )
```

Verifica se uma string contém um número que pode ser convertido. Mais especeificamente, se é um natural menor que 10000.

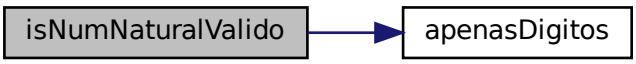
Parâmetros

str	Uma string qualquer.

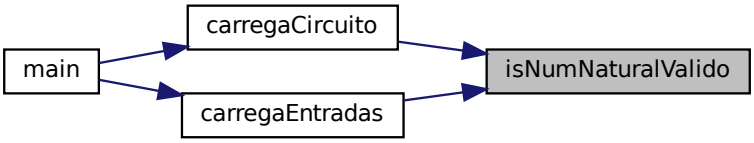
Retorna

Verdadeiro se pode ser convertido.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.13.1.11 isPalavra()

```
int isPalavra (
            Token * tk )
```

Verifica se um token é uma palavra reservada em Verilog.

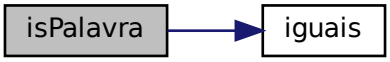
Parâmetros

tk	Um objeto Token.

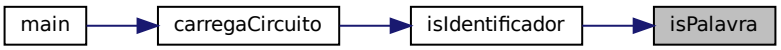
Retorna

Verdadeiro se o valor do token for palavra reservada em Verilog, falso c. c.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.13.1.12 isSimbolo()

```
int isSimbolo (
            char c )
```

Retorna verdadeiro se c for um simbolo em Verilog.

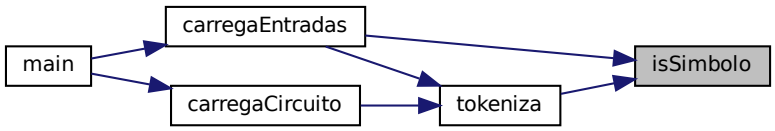
Parâmetros

c	Um char qualquer.

Retorna

True se c for simbolo, False caso contrario.

Este é o diagrama das funções que utilizam esta função:

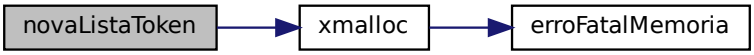


4.13.1.13 novaListaToken()

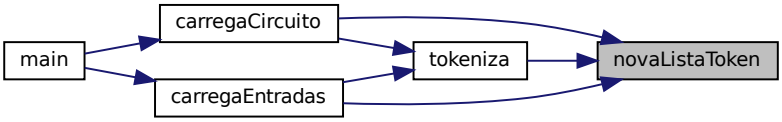
```
ListaToken * novaListaToken ( )
```

Inicializa uma lista vazia, i. e., com zero elementos.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

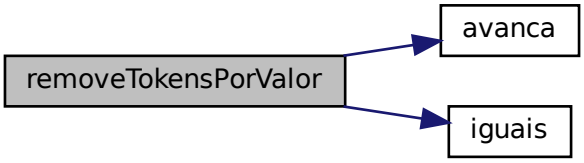


4.13.1.14 removeTokensPorValor()

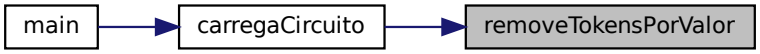
```
int removeTokensPorValor (
            ListaToken * lst,
            char * tok )
```

Remove todos os tokens com o valor indicado da lista.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.13.1.15 tokeniza()

```
ListaToken * tokeniza (
            FILE * arquivo )
```

Cria uma lista de Tokens que tem significado para o processamento sintatico, a partir do arquivo com o codigo fonte em Verilog.

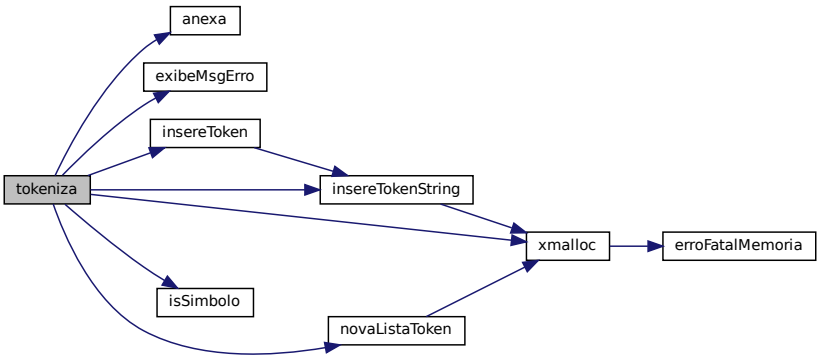
Parâmetros

arquivo	O handler do arquivo a ser processado.

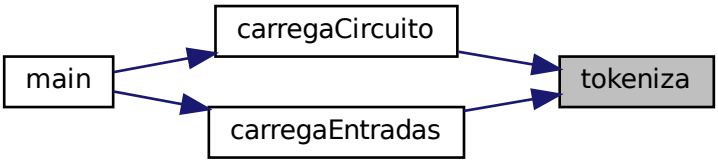
Retorna

A lista de tokens.

Grafo de chamadas desta função:



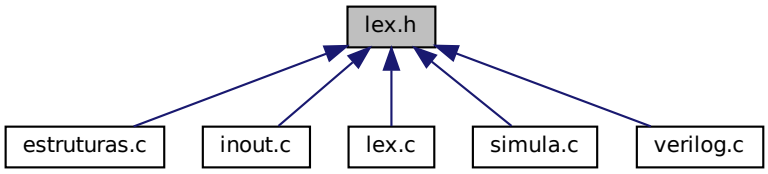
Este é o diagrama das funções que utilizam esta função:



4.14 Referência ao arquivo lex.h

Funcoes elementares de processamento lexico dos arquivos fonte.

Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Estruturas de Dados

- struct st_token

  Tipo basico para o elemento que representa um token.
- struct st_listaToken

  Tipo para uma lista encadeada de Tokens.

Macros

- #define MAX_TOKEN_SIZE 80
- #define MAX_DIGITOS_NUM 4

Definições de tipos

- typedef enum en_keyword KeywordId
- typedef enum en_grupoToken GrupoToken
- typedef struct st_token Token

  Tipo basico para o elemento que representa um token.
- typedef struct st_listaToken ListaToken

  Tipo para uma lista encadeada de Tokens.

Enumerações

- enum en_keyword { kw_module , kw_endmodule }
- enum en_grupoToken { tokenSimbolo , tokenPalavra , tokenIdent }

Funções

- ListaToken ∗ novaListaToken ()

  Inicializa uma lista vazia, i. e., com zero elementos.
- int insereToken (ListaToken ∗lista, char tok, int p_linha, int p_coluna)

  Insere na lista um novo token a partir de um caractere, deve-se especificar a posicao do mesmo no arquivo.
- int insereTokenString (ListaToken ∗lista, char ∗tok, int p_linha, int p_coluna)

  Insere na lista um novo token a partir de uma string, deve-se especificar a posicao do mesmo no arquivo.
- int removeTokensPorValor (ListaToken ∗lst, char ∗tok)

  Remove todos os tokens com o valor indicado da lista.
- int anexa (char ∗str, char c)

  Faz o apend de um char numa string qualquer.
- int isSimbolo (char c)

  Retorna verdadeiro se c for um simbolo em Verilog.
- void exibeListaDeToken (ListaToken ∗tokens)

  Imprime na tela os tokens dessa lista, um por linha.
- int iguais (char ∗a, char ∗b)

  Retorna verdadeiro se duas strings são iguais.
- void avanca (Token ∗∗t)

  Avanca o iterador de token para o próximo da lista.
- int isPalavra (Token ∗tk)

  Verifica se um token é uma palavra reservada em Verilog.
- int isIdentificador (Token ∗tk)

Verifica se um token é um nome permitido de identificador.

- int identExiste (ListaToken *lst, char *str)

  Retorna verdadeiro se a string esta contida em algum token da lista.

- ListaToken * tokeniza (FILE *arquivo)

  Cria uma lista de Tokens que tem significado para o processamento sintatico, a partir do arquivo com o codigo fonte em Verilog.

- int apenasDigitos (char *str)

  Verifica se uma string contém apenas dígitos (0, 1, 2, ..., 9).

- int isNumNaturalValido (char *str)

  Verifica se uma string contém um número que pode ser convertido. Mais especeificamente, se é um natural menor que 10000.

4.14.1 Descrição detalhada

Funcoes elementares de processamento lexico dos arquivos fonte.

4.14.2 Documentação das macros

4.14.2.1 MAX_DIGITOS_NUM

```
#define MAX_DIGITOS_NUM 4
```

4.14.2.2 MAX_TOKEN_SIZE

```
#define MAX_TOKEN_SIZE 80
```

4.14.3 Documentação dos tipos

4.14.3.1 GrupoToken

```
typedef enum en_grupoToken GrupoToken
```

4.14.3.2 KeywordId

```
typedef enum en_keyword KeywordId
```

4.14.3.3 ListaToken

```
typedef struct st_listaToken ListaToken
```

Tipo para uma lista encadeada de Tokens.

4.14.3.4 Token

```
typedef struct st_token Token
```

Tipo basico para o elemento que representa um token.

4.14.4 Documentação dos valores da enumeração

4.14.4.1 en_grupoToken

```
enum en_grupoToken
```

Valores de enumerações

tokenSimbolo	
tokenPalavra	
tokenIdent	

4.14.4.2 en_keyword

```
enum en_keyword
```

Valores de enumerações

kw_module	
kw_endmodule	

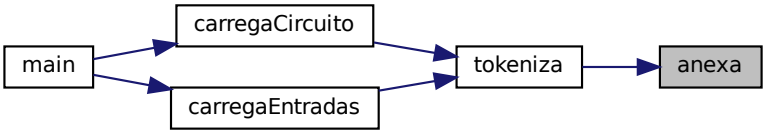
4.14.5 Documentação das funções

4.14.5.1 anexa()

```
int anexa (
            char * str,
            char c )
```

Faz o apend de um char numa string qualquer.

Este é o diagrama das funções que utilizam esta função:



4.14.5.2 apenasDigitos()

```
int apenasDigitos (
            char * str )
```

Verifica se uma string contém apenas dígitos (0, 1, 2, ..., 9).

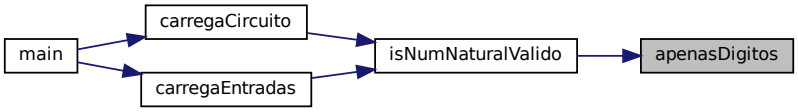
Parâmetros

str	Uma string qualquer.

Retorna

Verdadeiro se há apenas dígitos, falso na ocorrência de qualquer outro tipo de caractere.

Este é o diagrama das funções que utilizam esta função:



4.14.5.3 avanca()

```
void avanca (
            Token ** t )
```

Avanca o iterador de token para o próximo da lista.

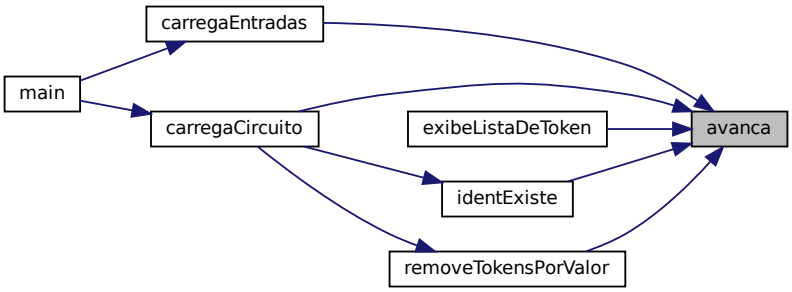
Parâmetros

t	Um ponteiro para um ponteiro de um Token.

Retorna

Void.

Este é o diagrama das funções que utilizam esta função:



4.14.5.4 exibeListaDeToken()

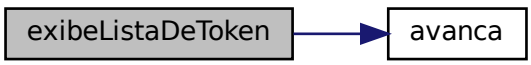
```
void exibeListaDeToken (
            ListaToken * tokens )
```

Imprime na tela os tokens dessa lista, um por linha.

Retorna

Void.

Grafo de chamadas desta função:



4.14.5.5 identExiste()

```
int identExiste (
            ListaToken * lst,
            char * str )
```

Retorna verdadeiro se a string esta contida em algum token da lista.

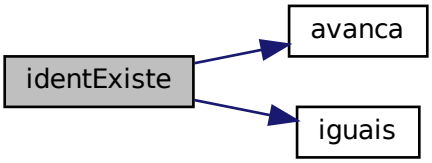
Parâmetros

lst	Uma lista de Tokens.
str	Uma string qualquer.

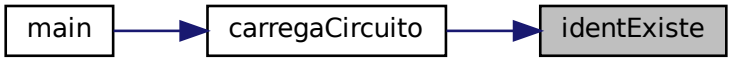
Retorna

Verdadeiro se str é o valor de algum Token em lst, falso caso contrário.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

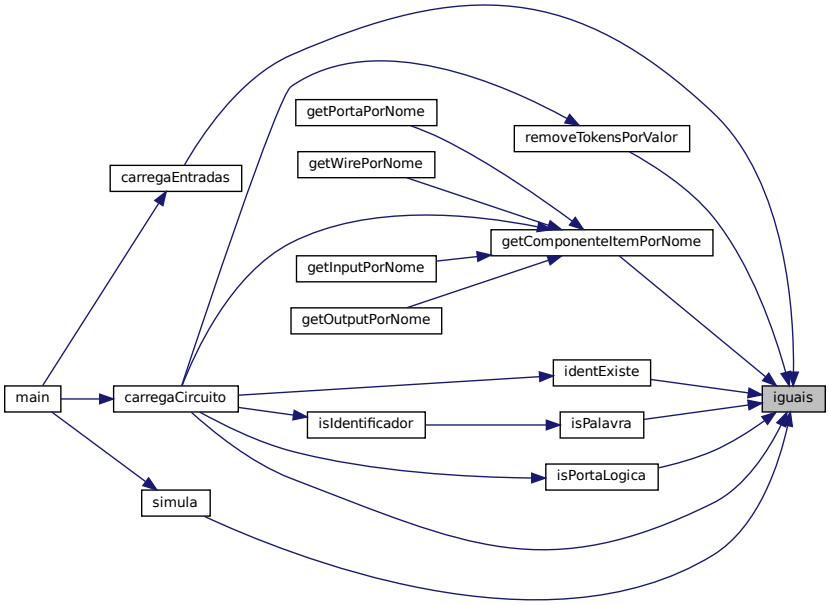


4.14.5.6 iguais()

```
int iguais (
            char * a,
            char * b )
```

Retorna verdadeiro se duas strings são iguais.

Este é o diagrama das funções que utilizam esta função:



4.14.5.7 insereToken()

```
int insereToken (
            ListaToken * lista,
            char tok,
            int p_linha,
            int p_coluna )
```

Insere na lista um novo token a partir de um caractere, deve-se especificar a posicao do mesmo no arquivo.

Parâmetros

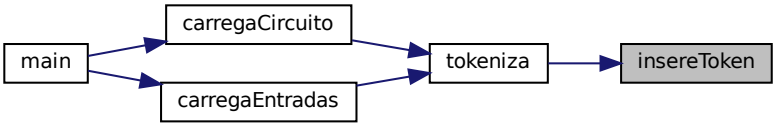
lista	Onde sera inserido o token.
tok	Um token de apenas um caractere.
p_linha	Linha no arquivo onde está o token.
p_coluna	Coluna no arquivo onde inicia-se o token.

Retorna

Verdadeiro caso sucesso, falso caso falhe.

Grafo de chamadas desta função:

Este é o diagrama das funções que utilizam esta função:



4.14.5.8 insereTokenString()

```
int insereTokenString (
            ListaToken * lista,
            char * tok,
            int p_linha,
            int p_coluna )
```

Insere na lista um novo token a partir de uma string, deve-se especificar a posicao do mesmo no arquivo.

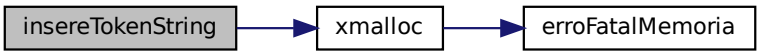
Parâmetros

lista	Onde sera inserido o token.
tok	Uma string contendo o token.
p_linha	Linha no arquivo onde está o token.
p_coluna	Coluna no arquivo onde inicia-se o token.

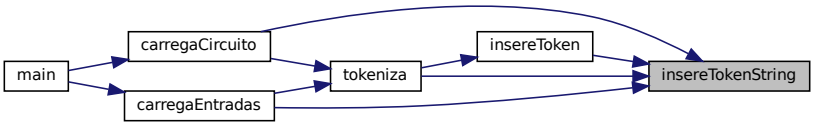
Retorna

Verdadeiro caso sucesso, falso caso falhe.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.14.5.9 isIdentificador()

```
int isIdentificador (
            Token * tk )
```

Verifica se um token é um nome permitido de identificador.

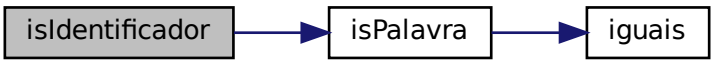
Parâmetros

str	Uma string qualquer.

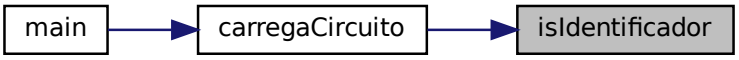
Retorna

Verdadeiro se o valor do token for um nome permitido de identificador, falso c.c.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.14.5.10 isNumNaturalValido()

```
int isNumNaturalValido (
            char * str )
```

Verifica se uma string contém um número que pode ser convertido. Mais especeificamente, se é um natural menor que 10000.

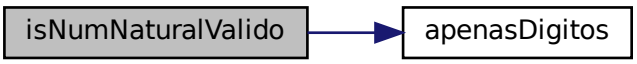
Parâmetros

str	Uma string qualquer.

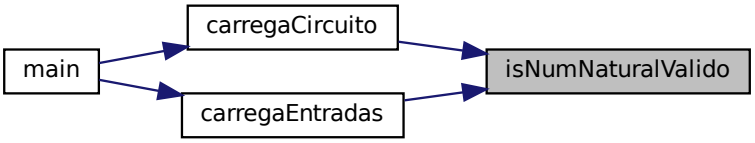
Retorna

Verdadeiro se pode ser convertido.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.14.5.11 isPalavra()

```
int isPalavra (
            Token * tk )
```

Verifica se um token é uma palavra reservada em Verilog.

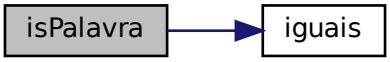
Parâmetros

tk	Um objeto Token.

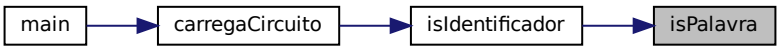
Retorna

Verdadeiro se o valor do token for palavra reservada em Verilog, falso c. c.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.14.5.12 isSimbolo()

```
int isSimbolo (
            char c )
```

Retorna verdadeiro se c for um simbolo em Verilog.

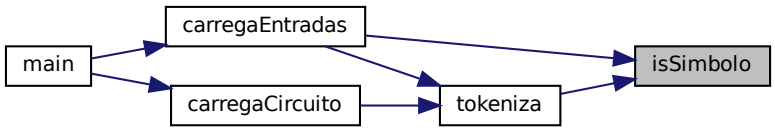
Parâmetros

c	Um char qualquer.

Retorna

True se c for simbolo, False caso contrario.

Este é o diagrama das funções que utilizam esta função:

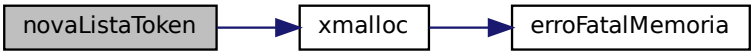


4.14.5.13 novaListaToken()

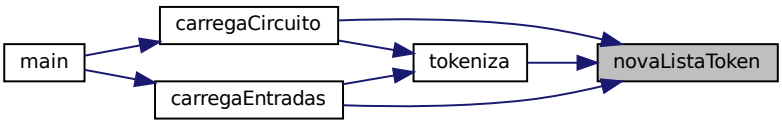
```
ListaToken * novaListaToken ( )
```

Inicializa uma lista vazia, i. e., com zero elementos.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

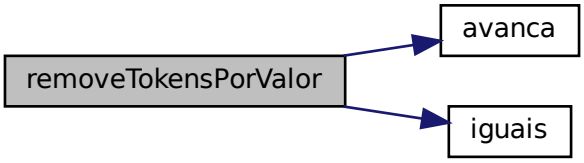


4.14.5.14 removeTokensPorValor()

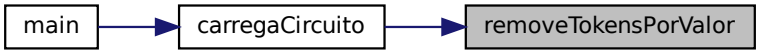
```
int removeTokensPorValor (
            ListaToken * lst,
            char * tok )
```

Remove todos os tokens com o valor indicado da lista.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.14.5.15 tokeniza()

```
ListaToken * tokeniza (
            FILE * arquivo )
```

Cria uma lista de Tokens que tem significado para o processamento sintatico, a partir do arquivo com o codigo fonte em Verilog.

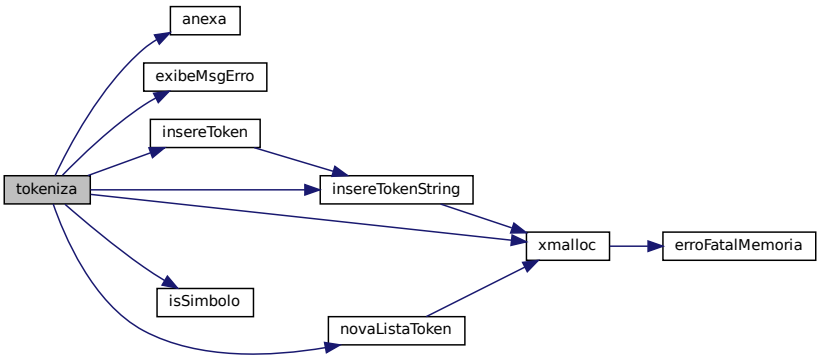
Parâmetros

arquivo	O handler do arquivo a ser processado.

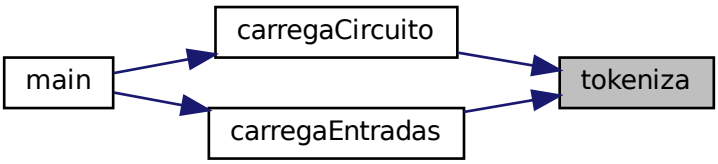
Retorna

A lista de tokens.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.15 lex.h

Ir para a documentação deste arquivo.

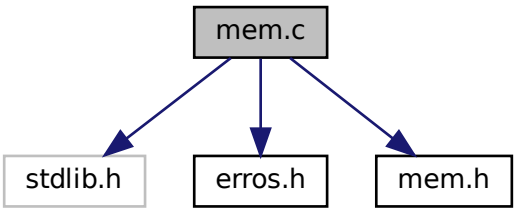
```
1
6 #ifndef LEX_H
7
8 #define LEX_H
9
10 #define MAX_TOKEN_SIZE 80
11 #define MAX_DIGITOS_NUM 4
12
15 typedef enum en_keyword {
16     kw_module, kw_endmodule // nao sei se isso vai ficar mesmo
17 } KeywordId;
18
21 typedef enum en_grupoToken {
22     tokenSimbolo, tokenPalavra, tokenIdent
23 } GrupoToken;
24
27 typedef struct st_token {
28     char valor[MAX_TOKEN_SIZE];
29     int linha;
30     int coluna;
31     GrupoToken tipo;
32     struct st_token* seguinte;
33 } Token;
34
37 typedef struct st_listaToken {
38     Token* primeiro;
```

```
39     Token* ultimo;
40     int tamanho;
41 } ListaToken;
42
45 ListaToken* novaListaToken();
46
55 int insereToken(ListaToken* lista, char tok, int p_linha, int p_coluna);
56
65 int insereTokenString(ListaToken* lista, char* tok, int p_linha, int p_coluna);
66
69 int removeTokensPorValor(ListaToken* lst, char* tok);
70
73 int anexa(char* str, char c);
74
79 int isSimbolo(char c);
80
84 void exibeListaDeToken(ListaToken* tokens);
85
88 int iguais(char* a, char* b);
89
94 void avanca(Token** t);
95
100 int isPalavra(Token* tk);
101
106 int isIdentificador(Token* tk);
107
113 int identExiste(ListaToken* lst, char* str);
114
120 ListaToken* tokeniza(FILE *arquivo);
121
126 int apenasDigitos(char* str);
127
133 int isNumNaturalValido(char* str);
134
135 #endif // LEX_H
```

4.16 Referência ao arquivo mem.c

```
#include <stdlib.h>
#include "erros.h"
#include "mem.h"
```

Diagrama de dependências de inclusão para mem.c:



Funções

- void * xmalloc (size_t t)

  Wrapper para malloc com verificação de erro.
- void * xrealloc (void *m, size_t t)

  Wrapper para realloc com verificação de erro.
- void * xcalloc (size_t n, size_t t)

  Wrapper para calloc com verificação de erro.

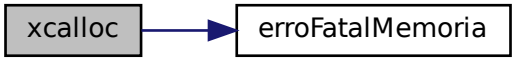
4.16.1 Documentação das funções

4.16.1.1 xcalloc()

```
void * xcalloc (
            size_t n,
            size_t t )
```

Wrapper para calloc com verificação de erro.

Grafo de chamadas desta função:

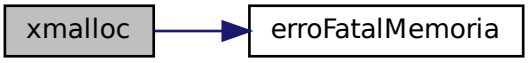


4.16.1.2 xmalloc()

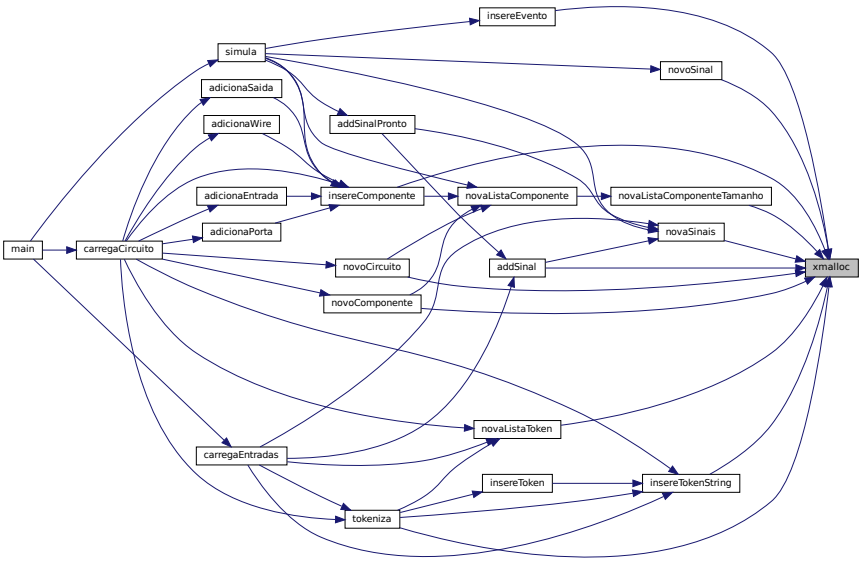
```
void * xmalloc (
            size_t t )
```

Wrapper para malloc com verificação de erro.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

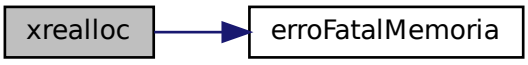


4.16.1.3 xrealloc()

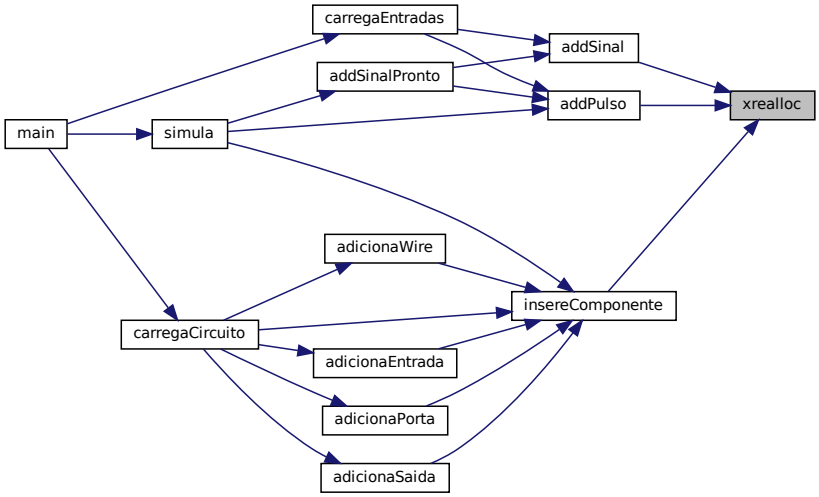
```
void * xrealloc (
            void * m,
            size_t t )
```

Wrapper para realloc com verificação de erro.

Grafo de chamadas desta função:



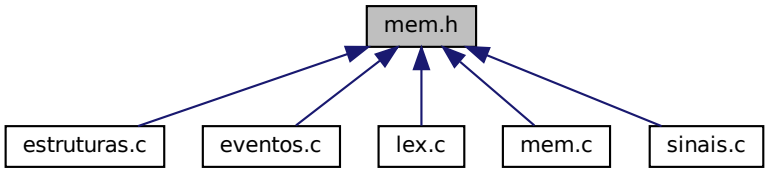
Este é o diagrama das funções que utilizam esta função:



4.17 Referência ao arquivo mem.h

Protótipos dos wrapers para funções de manipulação de memória.

Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Funções

- void ∗ xmalloc (size_t t)

  Wrapper para malloc com verificação de erro.
- void ∗ xrealloc (void ∗p, size_t t)

  Wrapper para realloc com verificação de erro.
- void ∗ xcalloc (size_t n, size_t t)

  Wrapper para calloc com verificação de erro.

4.17.1 Descrição detalhada

Protótipos dos wrapers para funções de manipulação de memória.

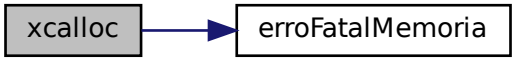
4.17.2 Documentação das funções

4.17.2.1 xcalloc()

```
void * xcalloc (
            size_t n,
            size_t t )
```

Wrapper para calloc com verificação de erro.

Grafo de chamadas desta função:

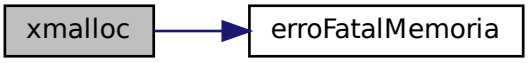


4.17.2.2 xmalloc()

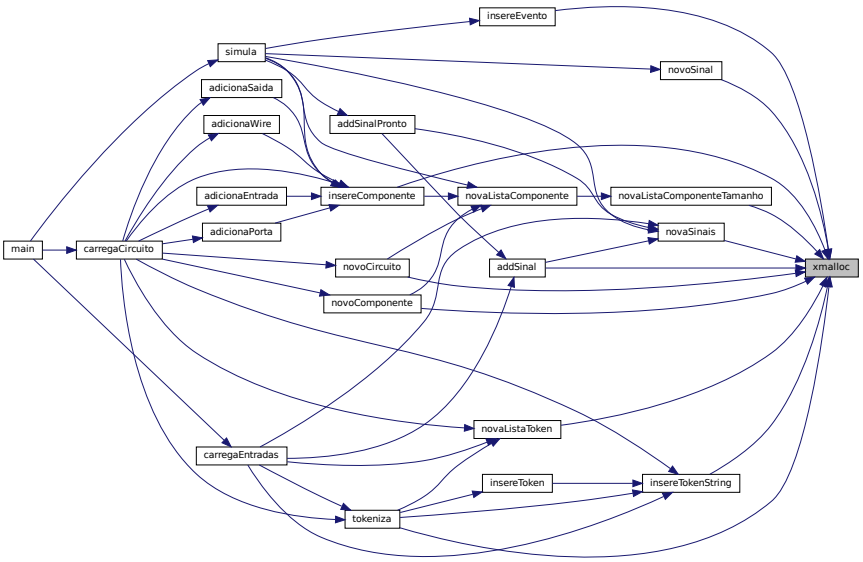
```
void * xmalloc (
            size_t t )
```

Wrapper para malloc com verificação de erro.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

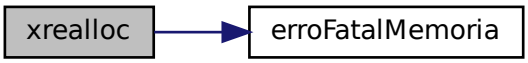


4.17.2.3 xrealloc()

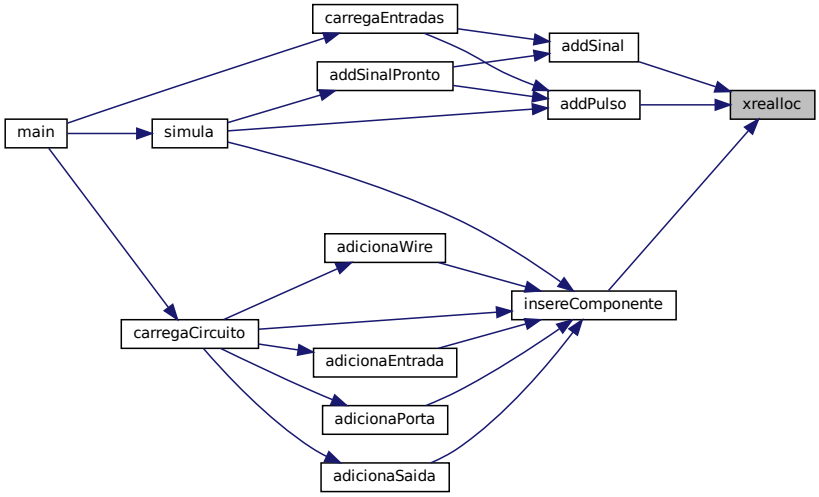
```
void * xrealloc (
            void * p,
            size_t t )
```

Wrapper para realloc com verificação de erro.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.18 mem.h

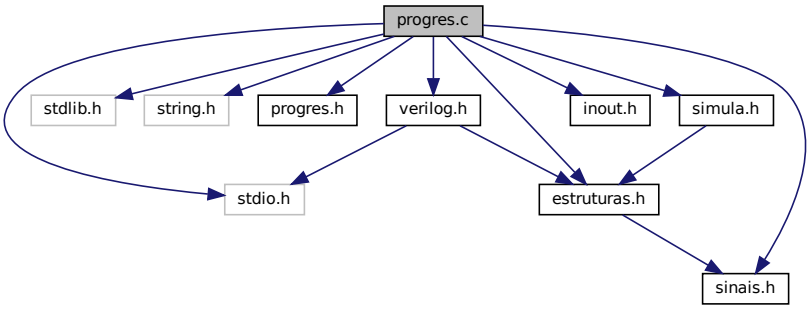
Ir para a documentação deste arquivo.

```
1
6 #ifndef MEM_H
7
8 #define MEM_H
9
12 void* xmalloc(size_t t);
13
16 void* xrealloc(void* p, size_t t);
17
20 void* xcalloc(size_t n, size_t t);
21
22 #endif // MEM_H
```

4.19 Referência ao arquivo progres.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "progres.h"
#include "estruturas.h"
#include "sinais.h"
#include "inout.h"
#include "verilog.h"
#include "simula.h"
```

Diagrama de dependências de inclusão para progres.c:



Funções

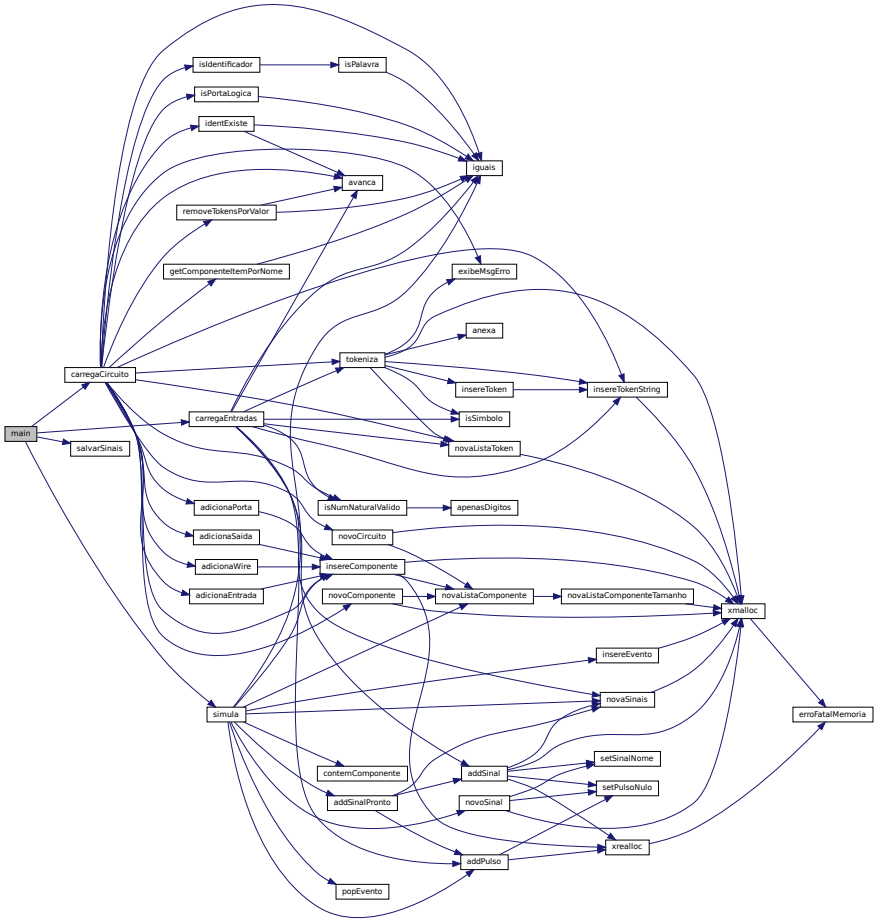
- int main (int argc, char ∗argv[ ])

4.19.1 Documentação das funções

4.19.1.1 main()

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

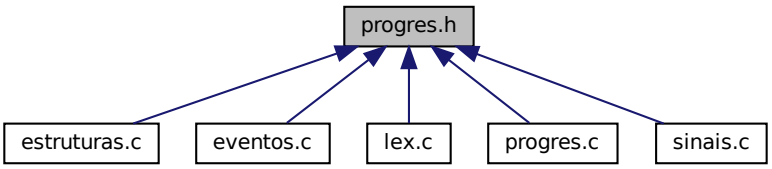
Grafo de chamadas desta função:



4.20 Referência ao arquivo progres.h

Protótipos do módulo principal do programa.

Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Macros

- #define MAX_FILE_PATH_SIZE 4096

4.20.1 Descrição detalhada

Protótipos do módulo principal do programa.

4.20.2 Documentação das macros

4.20.2.1 MAX_FILE_PATH_SIZE

```
#define MAX_FILE_PATH_SIZE 4096
```

4.21 progres.h

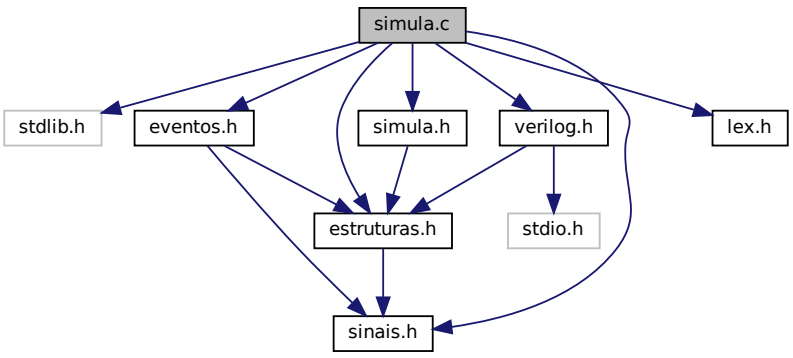
Ir para a documentação deste arquivo.

```
1
6 #ifndef PROGRES_H
7
8 #define PROGRES_H
9
10 #define MAX_FILE_PATH_SIZE 4096
11
12 #endif // PROGRES_H
```

4.22 Referência ao arquivo simula.c

```
#include <stdlib.h>
#include "simula.h"
#include "verilog.h"
#include "estruturas.h"
#include "sinais.h"
#include "lex.h"
#include "eventos.h"
```

Diagrama de dependências de inclusão para simula.c:



Funções

- Sinais ∗ simula (t_circuito ∗circuto, Sinais ∗entradas)

  Função que faz a simulação do circuito com as entradas especificadas. Em caso de sucesso, retorna as saídas dessa simulação.

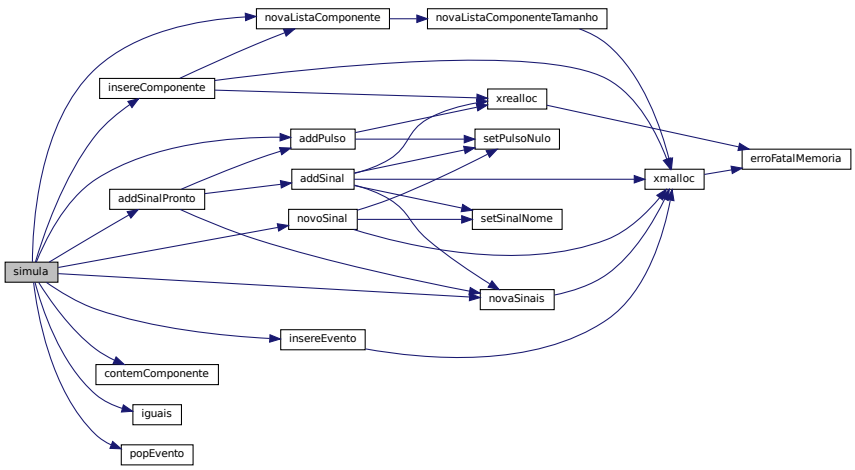
4.22.1 Documentação das funções

4.22.1.1 simula()

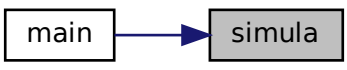
```
Sinais * simula (
            t_circuito * circuto,
            Sinais * entradas )
```

Função que faz a simulação do circuito com as entradas especificadas. Em caso de sucesso, retorna as saídas dessa simulação.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

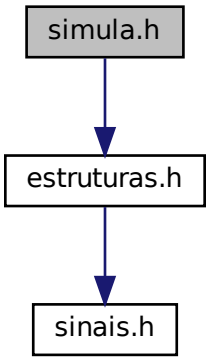


4.23 Referência ao arquivo simula.h

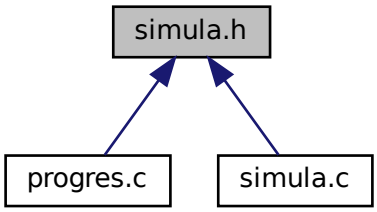
Protótipo da função principal da simulação.

```
#include "estruturas.h"
```

Diagrama de dependências de inclusão para simula.h:



Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Funções

- Sinais ∗ simula (t_circuito ∗circuto, Sinais ∗entradas)

  Função que faz a simulação do circuito com as entradas especificadas. Em caso de sucesso, retorna as saídas dessa simulação.

4.23.1 Descrição detalhada

Protótipo da função principal da simulação.

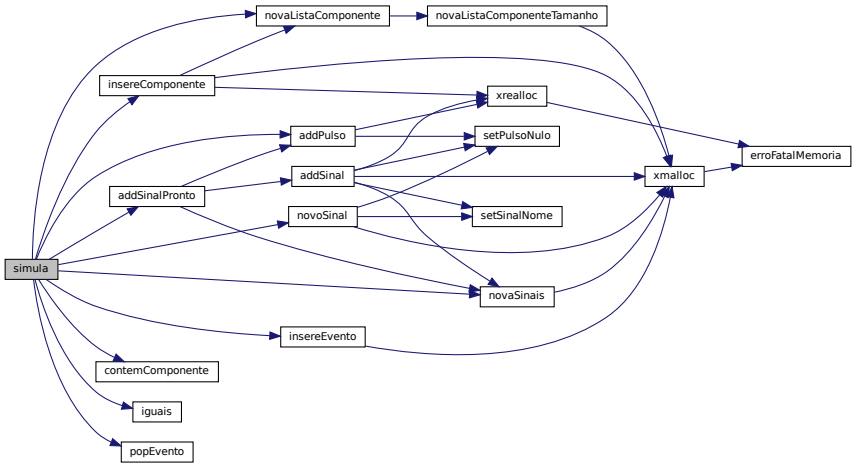
4.23.2 Documentação das funções

4.23.2.1 simula()

```
Sinais * simula (
            t_circuito * circuto,
            Sinais * entradas )
```

Função que faz a simulação do circuito com as entradas especificadas. Em caso de sucesso, retorna as saídas dessa simulação.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

4.24 simula.h

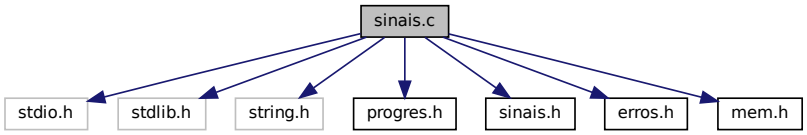
Ir para a documentação deste arquivo.

```
1
6 #ifndef SIMULA_H
7
8 #define SIMULA_H
9
10 #include "estruturas.h"
11
15 Sinais* simula(t_circuito* circuto, Sinais* entradas);
16
17 #endif // SIMULA_H
```

4.25 Referência ao arquivo sinais.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "progres.h"
#include "sinais.h"
#include "erros.h"
#include "mem.h"
```

Diagrama de dependências de inclusão para sinais.c:



Funções

- Sinal ∗ novoSinal (char ∗nome)

  Inicializa um sinal vazio com um respectivo nome.
- int setSinalNome (Sinal ∗s, char ∗nome)

  Muda a string contendo o nome do sinal para a indicada.
- int setPulsoNulo (Pulso ∗p)

  Define o pulso indicado com sendo nulo. Isto é, seu valor conterá nulo.
- int addPulso (Sinal ∗s, ValorLogico valor, Tempo duracao)

  Adiciona ao sinal, mais especificamente ao vetor de pulsos do obj. Sinal, mais um pulso de valor e duração indicados. É como se fosse um append, aqui fazemos uso de realloc.
- Sinais ∗ novaSinais ()

  Inicializa um nova estrutura Sinas vazia e devolve sua pos. de memória. Vazia significa: primeiro e ultimo apontam a NULL e num. de elem. é zero.
- int addSinal (Sinais ∗s, char ∗nome)

  Insere um sinal em branco na estrutura Sinais.
- int addSinalPronto (Sinais ∗ls, Sinal ∗sinal)

  Copia um sinal para a estrutura Sinais.

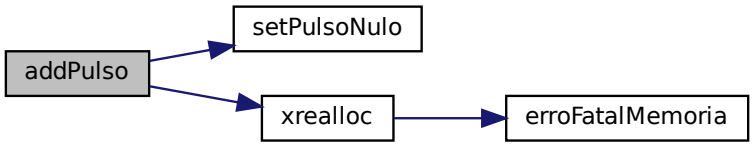
4.25.1 Documentação das funções

4.25.1.1 addPulso()

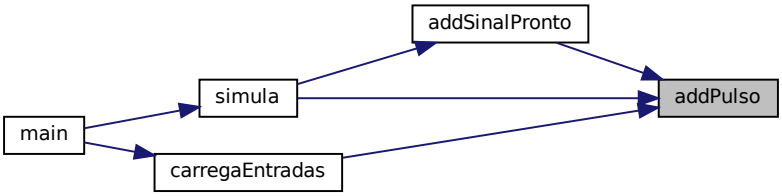
```
int addPulso (
            Sinal * s,
            ValorLogico valor,
            Tempo duracao )
```

Adiciona ao sinal, mais especificamente ao vetor de pulsos do obj. Sinal, mais um pulso de valor e duração indicados. É como se fosse um append, aqui fazemos uso de realloc.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

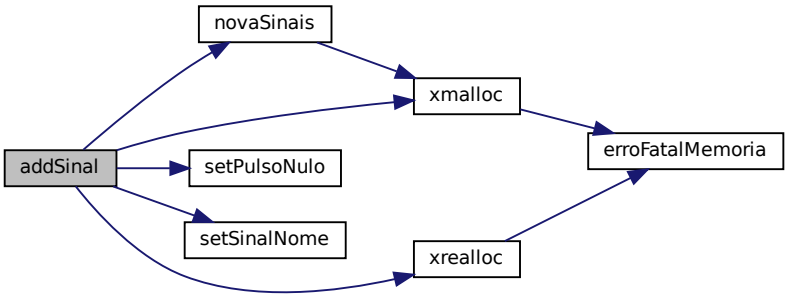


4.25.1.2 addSinal()

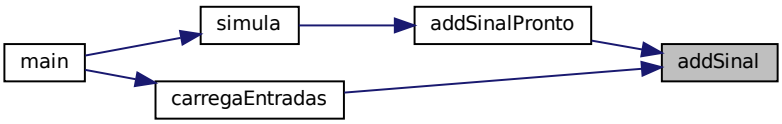
```
int addSinal (
            Sinais * s,
            char * nome )
```

Insere um sinal em branco na estrutura Sinais.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

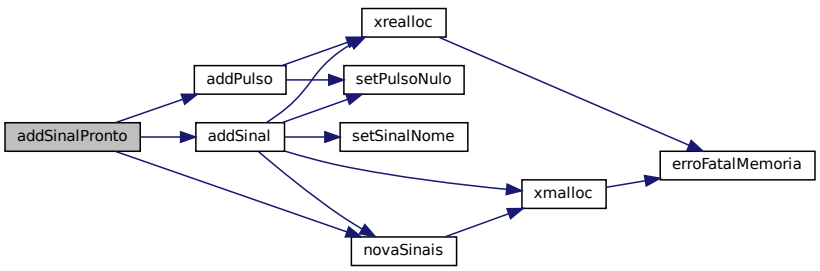


4.25.1.3 addSinalPronto()

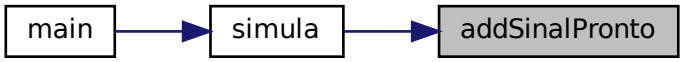
```
int addSinalPronto (
            Sinais * ls,
            Sinal * sinal )
```

Copia um sinal para a estrutura Sinais.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

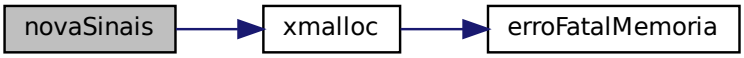


4.25.1.4 novaSinais()

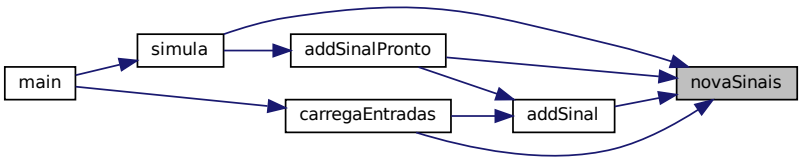
```
Sinais * novaSinais ( )
```

Inicializa um nova estrutura Sinas vazia e devolve sua pos. de memória. Vazia significa: primeiro e ultimo apontam a NULL e num. de elem. é zero.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

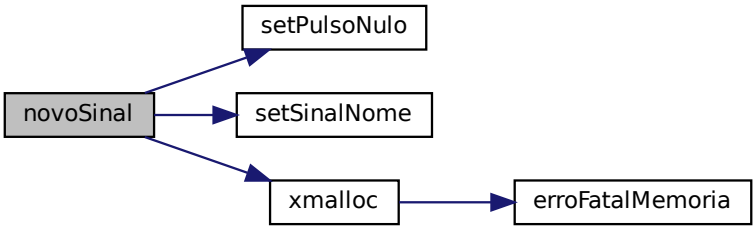


4.25.1.5 novoSinal()

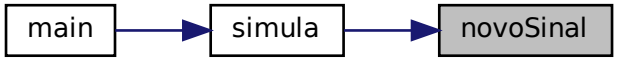
```
Sinal * novoSinal (
            char * nome )
```

Inicializa um sinal vazio com um respectivo nome.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

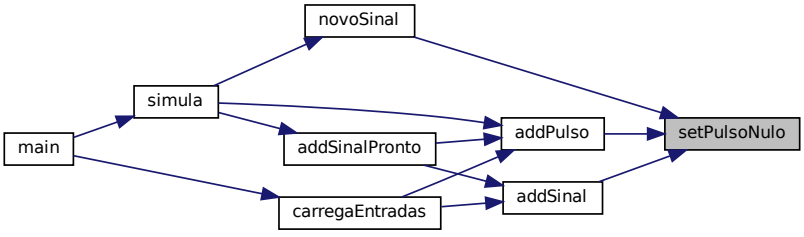


4.25.1.6 setPulsoNulo()

```
int setPulsoNulo (
            Pulso * p )
```

Define o pulso indicado com sendo nulo. Isto é, seu valor conterá nulo.

Este é o diagrama das funções que utilizam esta função:

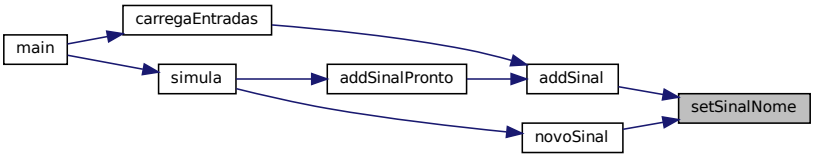


4.25.1.7 setSinalNome()

```
int setSinalNome (
            Sinal * s,
            char * nome )
```

Muda a string contendo o nome do sinal para a indicada.

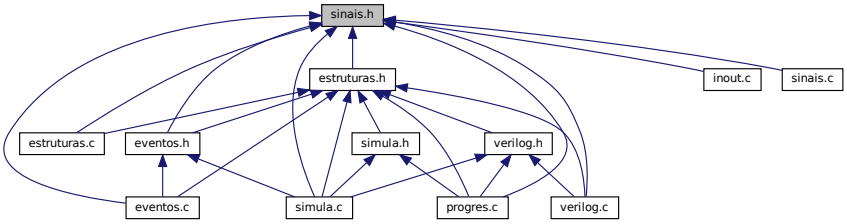
Este é o diagrama das funções que utilizam esta função:



4.26 Referência ao arquivo sinais.h

Estruturas e funções para manipulação de sinais de entrada e saída.

Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Estruturas de Dados

- struct st_pulso

  Um pulso de valor fixo e duração definida.
- struct st_sinal

  Um sinal contém um array de pulsos com o último pulso nulo. Semelhantemente a uma string.
- struct st_sinais

  Um conjunto de um ou mais sinais. Podem ser todos de entrada ou todos de saída.

Macros

- #define MAX_NOME_SINAL 50

Definições de tipos

- typedef enum en_valor ValorLogico

  Valor lógico de um pulso. Aqui, nulo serve para indicar o fim de uma "string" de pulsos.
- typedef enum en_un_tempo UnidTempo

  Unidades de tempo disponíveis para a duração de um pulso. Na ordem, segundo, milisegundo, microsegundo, nanosegundo, picosegundo e fentosegundo. O valor numérico é equivalente ao valor absoluto do módulo do expoente.
- typedef unsigned int Tempo

  O tipo das variáveis usadas na representação do tempo.
- typedef struct st_pulso Pulso

  Um pulso de valor fixo e duração definida.
- typedef struct st_sinal Sinal

  Um sinal contém um array de pulsos com o último pulso nulo. Semelhantemente a uma string.
- typedef struct st_sinais Sinais

  Um conjunto de um ou mais sinais. Podem ser todos de entrada ou todos de saída.

Enumerações

- enum en_valor {
  zero = 0 , um = 1 , x , z ,
  nulo }

  Valor lógico de um pulso. Aqui, nulo serve para indicar o fim de uma "string" de pulsos.
- enum en_un_tempo {
  UN_S = 0 , UN_100MS = 1 , UN_10MS = 2 , UN_MS = 3 ,
  UN_100US = 4 , UN_10US = 5 , UN_US = 6 , UN_100NS = 7 ,
  UN_10NS = 8 , UN_NS = 9 , UN_100PS = 10 , UN_10PS = 11 ,
  UN_PS = 12 , UN_100FS = 13 , UN_10FS = 14 , UN_FS = 15 }

  Unidades de tempo disponíveis para a duração de um pulso. Na ordem, segundo, milisegundo, microsegundo, nanosegundo, picosegundo e fentosegundo. O valor numérico é equivalente ao valor absoluto do módulo do expoente.

Funções

- Sinal ∗ novoSinal (char ∗nome)

  Inicializa um sinal vazio com um respectivo nome.
- int setSinalNome (Sinal ∗s, char ∗nome)

  Muda a string contendo o nome do sinal para a indicada.
- int setPulsoNulo (Pulso ∗p)

  Define o pulso indicado com sendo nulo. Isto é, seu valor conterá nulo.
- int addPulso (Sinal ∗s, ValorLogico valor, Tempo duracao)

  Adiciona ao sinal, mais especificamente ao vetor de pulsos do obj. Sinal, mais um pulso de valor e duração indicados. É como se fosse um append, aqui fazemos uso de realloc.
- Sinais ∗ novaSinais ()

  Inicializa um nova estrutura Sinas vazia e devolve sua pos. de memória. Vazia significa: primeiro e ultimo apontam a NULL e num. de elem. é zero.
- int addSinal (Sinais ∗s, char ∗nome)

  Insere um sinal em branco na estrutura Sinais.
- int addSinalPronto (Sinais ∗ls, Sinal ∗sinal)

  Copia um sinal para a estrutura Sinais.

4.26.1 Descrição detalhada

Estruturas e funções para manipulação de sinais de entrada e saída.

4.26.2 Documentação das macros

4.26.2.1 MAX_NOME_SINAL

```
#define MAX_NOME_SINAL 50
```

4.26.3 Documentação dos tipos

4.26.3.1 Pulso

```
typedef struct st_pulso Pulso
```

Um pulso de valor fixo e duração definida.

4.26.3.2 Sinais

```
typedef struct st_sinais Sinais
```

Um conjunto de um ou mais sinais. Podem ser todos de entrada ou todos de saída.

4.26.3.3 Sinal

```
typedef struct st_sinal Sinal
```

Um sinal contém um array de pulsos com o último pulso nulo. Semelhantemente a uma string.

4.26.3.4 Tempo

```
typedef unsigned int Tempo
```

O tipo das variáveis usadas na representação do tempo.

4.26.3.5 UnidTempo

```
typedef enum en_un_tempo UnidTempo
```

Unidades de tempo disponíveis para a duração de um pulso. Na ordem, segundo, milisegundo, microsegundo, nanosegundo, picosegundo e fentosegundo. O valor numérico é equivalente ao valor absoluto do módulo do expoente.

4.26.3.6 ValorLogico

```
typedef enum en_valor ValorLogico
```

Valor lógico de um pulso. Aqui, nulo serve para indicar o fim de uma "string" de pulsos.

4.26.4 Documentação dos valores da enumeração

4.26.4.1 en_un_tempo

```
enum en_un_tempo
```

Unidades de tempo disponíveis para a duração de um pulso. Na ordem, segundo, milisegundo, microsegundo, nanosegundo, picosegundo e fentosegundo. O valor numérico é equivalente ao valor absoluto do módulo do expoente.

Valores de enumerações

UN_S	
UN_100MS	
UN_10MS	
UN_MS	
UN_100US	
UN_10US	
UN_US	
UN_100NS	
UN_10NS	
UN_NS	
UN_100PS	
UN_10PS	
UN_PS	
UN_100FS	
UN_10FS	
UN_FS	

4.26.4.2 en_valor

```
enum en_valor
```

Valor lógico de um pulso. Aqui, nulo serve para indicar o fim de uma "string" de pulsos.

Valores de enumerações

zero	
um	
x	
z	
nulo	

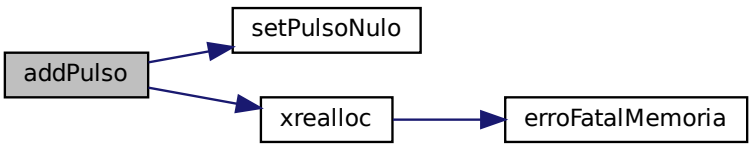
4.26.5 Documentação das funções

4.26.5.1 addPulso()

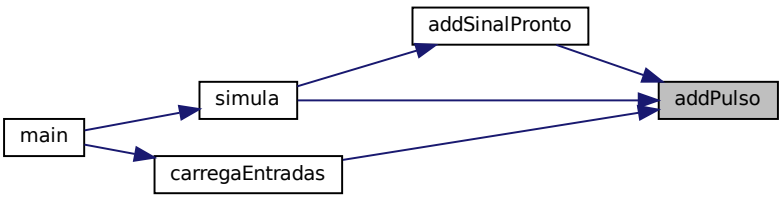
```
int addPulso (
            Sinal * s,
            ValorLogico valor,
            Tempo duracao )
```

Adiciona ao sinal, mais especificamente ao vetor de pulsos do obj. Sinal, mais um pulso de valor e duração indicados. É como se fosse um append, aqui fazemos uso de realloc.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

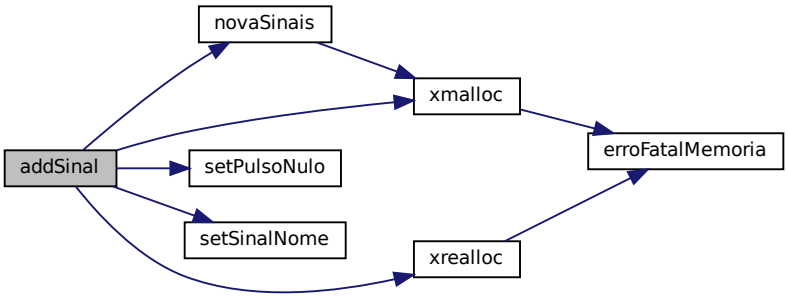


4.26.5.2 addSinal()

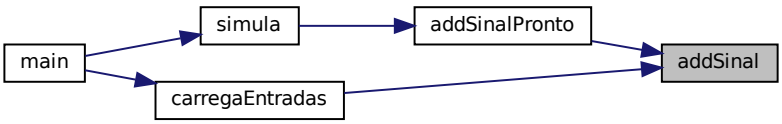
```
int addSinal (
            Sinais * s,
            char * nome )
```

Insere um sinal em branco na estrutura Sinais.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

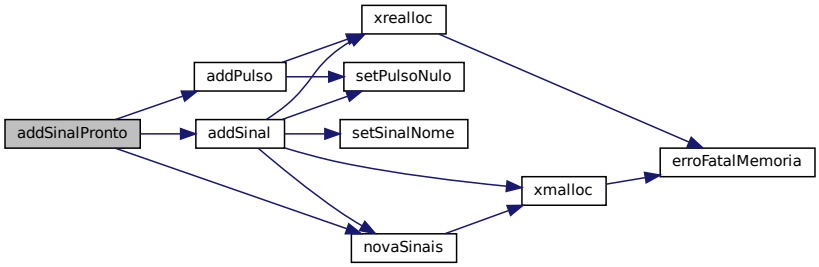


4.26.5.3 addSinalPronto()

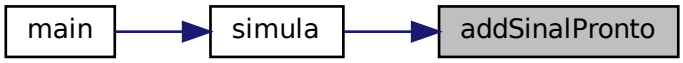
```
int addSinalPronto (
            Sinais * ls,
            Sinal * sinal )
```

Copia um sinal para a estrutura Sinais.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

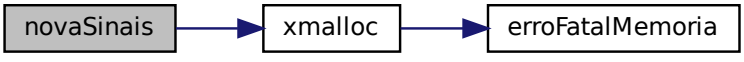


4.26.5.4 novaSinais()

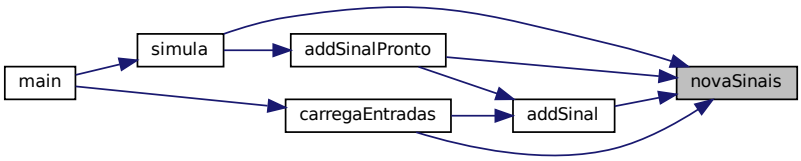
```
Sinais * novaSinais ( )
```

Inicializa um nova estrutura Sinas vazia e devolve sua pos. de memória. Vazia significa: primeiro e ultimo apontam a NULL e num. de elem. é zero.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

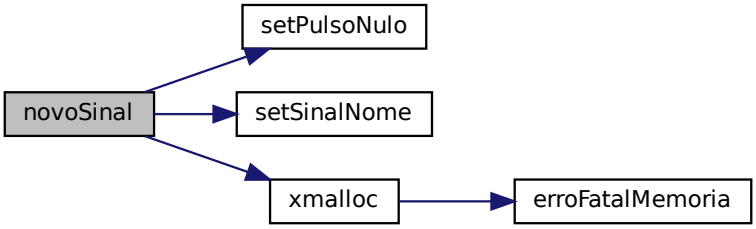


4.26.5.5 novoSinal()

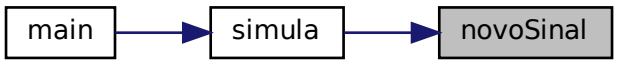
```
Sinal * novoSinal (
            char * nome )
```

Inicializa um sinal vazio com um respectivo nome.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

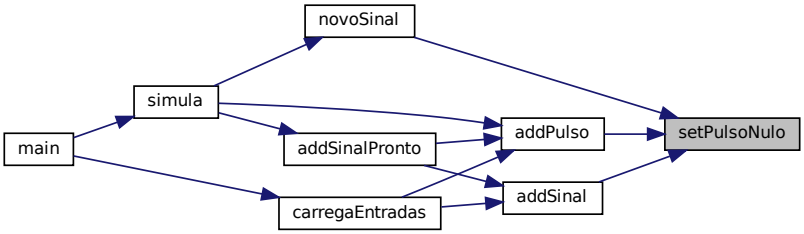


4.26.5.6 setPulsoNulo()

```
int setPulsoNulo (
            Pulso * p )
```

Define o pulso indicado com sendo nulo. Isto é, seu valor conterá nulo.

Este é o diagrama das funções que utilizam esta função:

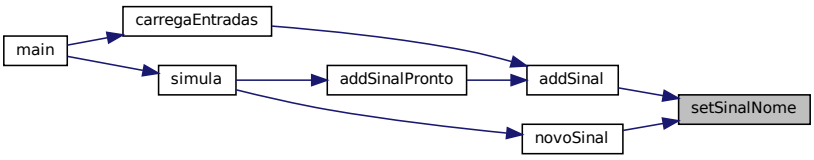


4.26.5.7 setSinalNome()

```
int setSinalNome (
            Sinal * s,
            char * nome )
```

Muda a string contendo o nome do sinal para a indicada.

Este é o diagrama das funções que utilizam esta função:



4.27 sinais.h

Ir para a documentação deste arquivo.

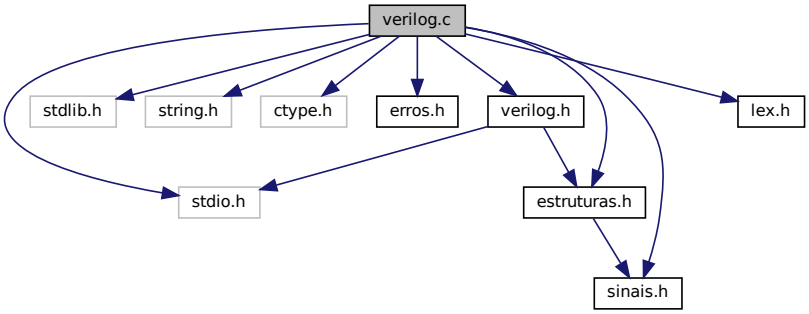
```
1
6 #ifndef SINAIS_H
7
8 #define SINAIS_H
9
10 #define MAX_NOME_SINAL 50
11
```

```
14 typedef enum en_valor {
15     zero = 0,
16     um = 1,
17     x,
18     z,
19     nulo
20 } ValorLogico;
21
26 typedef enum en_un_tempo {
27     UN_S = 0,
28     UN_100MS = 1,
29     UN_10MS = 2,
30     UN_MS = 3,
31     UN_100US = 4,
32     UN_10US = 5,
33     UN_US = 6,
34     UN_100NS = 7,
35     UN_10NS = 8,
36     UN_NS = 9,
37     UN_100PS = 10,
38     UN_10PS = 11,
39     UN_PS = 12,
40     UN_100FS = 13,
41     UN_10FS = 14,
42     UN_FS = 15
43 } UnidTempo;
44
47 typedef unsigned int Tempo;
48
51 typedef struct st_pulso {
52     ValorLogico valor;
53     Tempo tempo;
54     UnidTempo unidade;
55 } Pulso;
56
59 typedef struct st_sinal {
60     char nome[MAX_NOME_SINAL];
61     Pulso* pulsos;
62     Tempo duracaoTotal;
63 } Sinal;
64
67 typedef struct st_sinais {
68     int quantidade;
69     Sinal* lista;
70 } Sinais;
71
74 Sinal* novoSinal(char *nome);
75
78 int setSinalNome(Sinal* s, char* nome);
79
82 int setPulsoNulo(Pulso* p);
83
88 int addPulso(Sinal* s, ValorLogico valor, Tempo duracao);
89
93 Sinais* novaSinais();
94
97 int addSinal(Sinais* s, char* nome);
98
101 int addSinalPronto(Sinais *ls, Sinal *sinal);
102
103 #endif // SINAIS_H
```

4.28 Referência ao arquivo verilog.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "erros.h"
#include "verilog.h"
#include "estruturas.h"
#include "sinais.h"
#include "lex.h"
```

Diagrama de dependências de inclusão para verilog.c:



Funções

- t_circuito * carregaCircuito (FILE *arquivo)

  Cria uma estrutura de dados representando o circuito, a partir do arquivo com o codigo fonte em Verilog.
- int isPortaLogica (char *s)

  Retorna verdadeiro se uma string representa uma porta lógica em Verilog.

4.28.1 Documentação das funções

4.28.1.1 carregaCircuito()

```
t_circuito * carregaCircuito (
            FILE * arquivo )
```

Cria uma estrutura de dados representando o circuito, a partir do arquivo com o codigo fonte em Verilog.

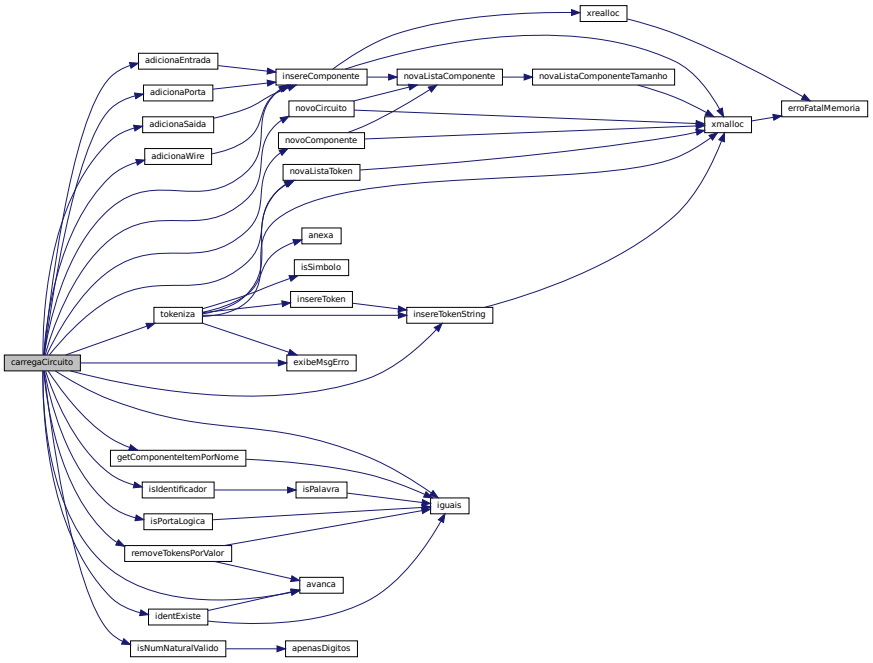
Parâmetros

arquivo	O handler do arquivo a ser processado.

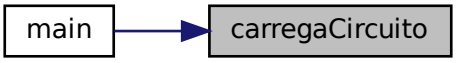
Retorna

A estrutura de dados do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.28.1.2 isPortaLogica()

```
int isPortaLogica (
            char * s )
```

Retorna verdadeiro se uma string representa uma porta lógica em Verilog.

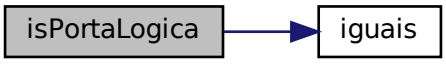
Parâmetros

s	Uma string qualquer.

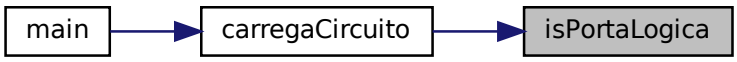
Retorna

Verdadeiro se s for igual a "and", "or", "nand", e etc.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:

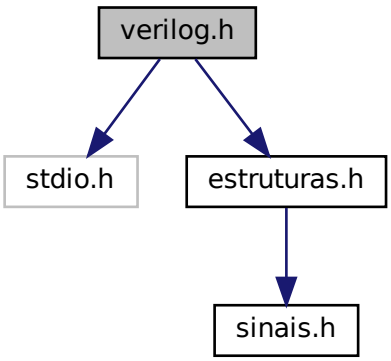


4.29 Referência ao arquivo verilog.h

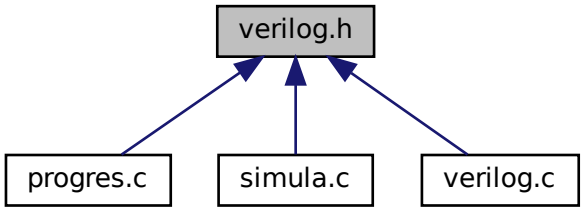
Rotinas para análise do arquivo Verilog.

```
#include <stdio.h>
#include "estruturas.h"
```

Diagrama de dependências de inclusão para verilog.h:



Este grafo mostra quais são os arquivos que incluem directamente ou indirectamente este arquivo:



Funções

- t_circuito ∗ carregaCircuito (FILE ∗arquivo)

  Cria uma estrutura de dados representando o circuito, a partir do arquivo com o codigo fonte em Verilog.
- int isPortaLogica (char ∗s)

  Retorna verdadeiro se uma string representa uma porta lógica em Verilog.

4.29.1 Descrição detalhada

Rotinas para análise do arquivo Verilog.

4.29.2 Documentação das funções

4.29.2.1 carregaCircuito()

```
t_circuito * carregaCircuito (
            FILE * arquivo )
```

Cria uma estrutura de dados representando o circuito, a partir do arquivo com o codigo fonte em Verilog.

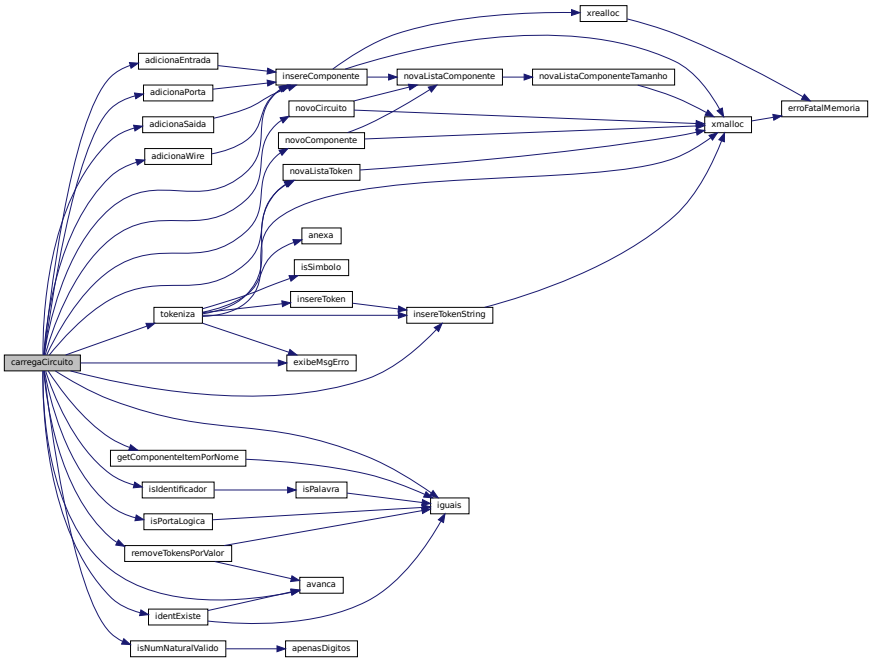
Parâmetros

arquivo	O handler do arquivo a ser processado.

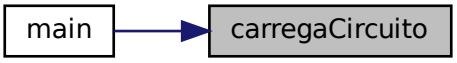
Retorna

A estrutura de dados do circuito.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.29.2.2 isPortaLogica()

```
int isPortaLogica (
            char * s )
```

Retorna verdadeiro se uma string representa uma porta lógica em Verilog.

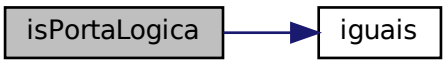
Parâmetros

s	Uma string qualquer.

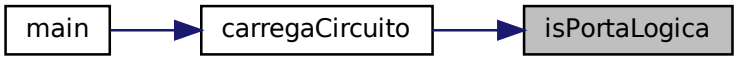
Retorna

Verdadeiro se s for igual a "and", "or", "nand", e etc.

Grafo de chamadas desta função:



Este é o diagrama das funções que utilizam esta função:



4.30 verilog.h

Ir para a documentação deste arquivo.

```
1
6 #ifndef VERILOG_H
7
8 #define VERILOG_H
9
10 #include <stdio.h>
11
12 #include "estruturas.h"
13
19 t_circuito* carregaCircuito(FILE* arquivo);
20
25 int isPortaLogica(char* s);
26
27 #endif // VERILOG_H
```

Índice