Pattern Assignment

Generated by Doxygen 1.8.12

Contents

Chapter 1

Pattern Assignment

1.1 the setup

1.1.1 STEP 1:either use make compile(which run the production versiion) or maked debug which runs the debug version.

1.1.2 STEO 2: run ./main to load the program into the main memory.

Chapter 2

Bug List

File log.h

No known Bugs.

File main.c

No known bugs.

File pattern.c

no known bugs.

File pattern.h

no known bugs

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

File Documentation

4.1 log.h File Reference

This file contains all the debug macros.

Macros

- #define **LOG**(msg)

4.1.1 Detailed Description

This file contains all the debug macros.

This debug macros when enabled in debug mode print the debug messages into the stderr.

Author

Prakash

Bug No known Bugs.

4.2 main.c File Reference

This file contains the driver program to test all the pattern method's.

```
#include <stdio.h>
#include "pattern.h"
#include <stdint.h>
#include <stdlib.h>
```

Enumerations

- enum { **ZERO_FILL**, **PATTERN_FILL**, **PATTERN_SEARCH**, **EXIT** }

Functions

- int **main** ()

Variables

- char **ch** [1024]

4.2.1 Detailed Description

This file contains the driver program to test all the pattern method's.

This driver program creates an static array of 1024 bytes and give the user an menu system to test all the Pattern functions on the 1024 byte array.

Author

Prakash

Bug No known bugs.

4.3 pattern.c File Reference

this file contain's all the implementations of the pattern operations.

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

Functions

- static int get_lcm (int a, int b)

  this function return the lcm of two numbers using the euclid's algorithm.
- void init_zero (char *start_addr, char *end_addr)

  fills all the bytes between the start and end with zeroes.
- void fill_pattern (char *start_addr, char *end_addr, char *pattern)

  fills all the bytes between the start and the end with specified bit pattern
- char * search_pattern (char *start_addr, char *end_addr, char *pattern)

  return bytes address where the sequence does not match the pattern.

4.3.1 Detailed Description

this file contain's all the implementations of the pattern operations.

Author

Prakash

Bug no known bugs.

4.3.2 Function Documentation

4.3.2.1 void fill_pattern (char * *start_addr*, char * *end_addr*, char * *pattern*)

fills all the bytes between the start and the end with specified bit pattern

This function calculates the length of the pattern and computes the lcm of 8 and the pattern length to get the byte after which the pattern is repeated. Then it precomputes the all the bytes ultil which the bit pattern does not get repeated.After computing the unique bytes then it assigns the complete range with this bytes using modulo operation's on them.

Parameters

start_addr	start address of the sequence.
end_addr	end address of the sequemnce.

Returns

Void

4.3.2.2 static int get_lcm (int *a*, int *b*) [static]

this function return the lcm of two numbers using the euclid's algorithm.

This function cannot accessed outside the file.

Parameters

a	the first number.
b	the second Number.

Returns

lcm of a and b.

4.3.2.3 void init_zero (char * *start_addr*, char * *end_addr*)

fills all the bytes between the start and end with zeroes.

This function calculates the number of bytes between the start and the end address and assigns the value zero to it.

Parameters

start_addr	start address of the sequence.
end_addr	end address of the sequence.

Returns

void

4.3.2.4 char* search_pattern (char * *start_addr*, char * *end_addr*, char * *pattern*)

return bytes address where the sequence does not match the pattern.

As fill_patern this compute the unique sequence of the pattern and then checks whether all the bytes in the given sequence are following the sequence using the modulo operation.

Parameters

start_addr	start address of the sequence.
end_addr	end address of the sequence.

Returns

the byte where the match fails in the sequence.

4.4 pattern.h File Reference

prototypes for all the pattern function's.

Functions

- void init_zero (char *start_addr, char *end_addr)
  fills all the bytes between the start and end with zeroes.
- void fill_pattern (char *start_addr, char *end_addr, char *pattern)
  fills all the bytes between the start and the end with specified bit pattern
- char * search_pattern (char *start_addr, char *end_addr, char *pattern)
  return bytes address where the sequence does not match the pattern.

4.4.1 Detailed Description

prototypes for all the pattern function's.

Author

Prakash

Bug no known bugs

4.4.2 Function Documentation

4.4.2.1 void fill_pattern (char * *start_addr,* char * *end_addr,* char * *pattern*)

fills all the bytes between the start and the end with specified bit pattern

This function calculates the length of the pattern and computes the lcm of 8 and the pattern length to get the byte after which the pattern is repeated. Then it precomputes the all the bytes ultil which the bit pattern does not get repeated.After computing the unique bytes then it assigns the complete range with this bytes using modulo operation's on them.

Parameters

start_addr	start address of the sequence.
end_addr	end address of the sequemnce.

Returns

Void

4.4.2.2 void init_zero (char * *start_addr,* char * *end_addr*)

fills all the bytes between the start and end with zeroes.

This function calculates the number of bytes between the start and the end address and assigns the value zero to it.

Parameters

start_addr	start address of the sequence.
end_addr	end address of the sequence.

Returns

void

4.4.2.3 char* search_pattern (char * *start_addr,* char * *end_addr,* char * *pattern*)

return bytes address where the sequence does not match the pattern.

As fill_patern this compute the unique sequence of the pattern and then checks whether all the bytes in the given sequence are following the sequence using the modulo operation.

Parameters

start_addr	start address of the sequence.
end_addr	end address of the sequence.

Returns

the byte where the match fails in the sequence.

Index