\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: hw8

Author: F74086381 蘇恩質 <grgy078033@gmail.com>

Class: 112資訊系 (乙班)

Description:

先建立三個指標陣列，每個陣列當中的元素都是\*head指標，到時候依各資料當中length的大小來判定要放入第幾個陣列。

先將IP轉為unsigned後，當length為8~15時，用前8bit來判定要放入哪一個linked list，後面的兩組則看前12bit。

當linked list還是空的時候，將資料視為第一筆資料，用\*head指標指向這筆資料，而若linked list不是空的，就讓他插在第一個node前。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

double power(int x, int y)

{

int i;

double result = 1.0;

if(y > 0)

{

for(i = 0; i < y; i++)

result \*= x;

}

else if(y < 0)

{

int z = 0 - y;

for(i = 0; i < z; i++)

result /= x;

}

return result;

}

int convertBinaryToDecimal(long long n)

{

int decimalNumber = 0, i = 0, remainder;

while (n!=0)

{

remainder = n%10;

n /= 10;

decimalNumber += remainder\*power(2,i);

++i;

}

return decimalNumber;

}

struct node{

unsigned ip;

int len;

struct node \*next;

};

void f1(char\* file\_name, struct node \*p1[], struct node \*p2[], struct node\*p3[])

{

int i, index, length;

struct node \*new;

char b[10];

FILE \*p;

p = fopen("file\_name", "r");

char\* a;

fgets(a, 100, p);

fclose(p);

int num1, num2, num3, num4;

sscanf(a, "%d %d %d %d" , &num1, &num2, &num3, &num4);

unsigned n, n1, n2;

n1 = (((n << 8 | num1) << 8 | num2) << 8 | num3) << 8 | num4;

n2 = n1;

struct node \*data;

for(i = 0; i < 100; i++)/\*找length\*/

{

if(a[i] == '/')

{

if(a[i+1] == '8')

length = 8;

if(a[i+1] == '9')

length = 9;

else

{

int x = (int)a[i+1];

int y = (int)a[i+2];

length = x \* 10 + y;

}

break;

}

}

printf("%d", length);

if(length >= 8 && length <= 15)

{

for(i = 0; i < 100; i++)

{

if(a[i] == '.')

{

strncpy(b, a+0, i);

if(i == 3)

index = b[0]\*100 + b[1]\*10 + b[0];

if(i == 2)

index = b[0]\*10 + b[1];

if(i == 1)

index = b[0];

if(p1[index-1] -> next == NULL)

{

data = malloc(sizeof(struct node));

data -> ip = \*(int\*)&n1;

data -> len = length;

data -> next = NULL;

p1[index-1] = data;

}

else

{

data = malloc(sizeof(struct node));

data -> ip = \*(int\*)&n1;

data -> len = length;

data -> next = p1[index-1];

p1[index-1] = data;

}

}

}

}

if(length >= 16 && length <= 23)

{

n1 >> 20;

index = convertBinaryToDecimal(n1);

if(p2[index-1] -> next == NULL)

{

data = malloc(sizeof(struct node));

data -> ip = \*(int\*)&n2;

data -> len = length;

data -> next = NULL;

p2[index-1] = data;

}

else

{

data = malloc(sizeof(struct node));

data -> ip = \*(int\*)&n2;

data -> len = length;

data -> next = p2[index-1];

p2[index-1] = data;

}

}

if(length >= 24 && length <= 32)

{

n1 >> 20;

index = convertBinaryToDecimal(n1);

if(p3[index-1] -> next == NULL)

{

data = malloc(sizeof(struct node));

data -> ip = \*(int\*)&n2;

data -> len = length;

data -> next = NULL;

p3[index-1] = data;

}

else

{

data = malloc(sizeof(struct node));

data -> ip = \*(int\*)&n2;

data -> len = length;

data -> next = p3[index-1];

p3[index-1] = data;

}

}

return;

}

void search(char\* file\_name, struct node \*p1[], struct node \*p2[], struct node \*p3[])

{

FILE \*p;

p = fopen("flie\_name", "r");

char\* a;

fgets(a, 100, p);

fclose(p);

int index, num1, num2, num3, num4;

int success = 0, fail = 0;

sscanf(a, "%d %d %d %d", &num1, &num2, &num3, &num4);

unsigned n, n1, n2;

n1 = (((n << 8 | num1) << 8 | num2) << 8 | num3) << 8 | num4;

n2 = n1;

for(struct node \*ptr = \*p1; ptr != NULL; ptr = ptr->next)

{

n1 >> 24;

index = convertBinaryToDecimal(n1);

if(p1[index-1]->ip >> (32 - p1[index-1]->len) == n2 >> (32 - p1[index-1]->len))

success++;

}

for(struct node \*ptr = \*p2; ptr != NULL; ptr = ptr->next)

{

n1 >> 20;

index = convertBinaryToDecimal(n1);

if(p2[index-1]->ip >> (32 - p2[index-1]->len == n2 >> (32 - p2[index-1]->len)))

success++;

}

for(struct node \*ptr = \*p3; ptr != NULL; ptr = ptr->next)

{

n1 >> 20;

index = convertBinaryToDecimal(n1);

if(p3[index-1]->ip >> (32 - p3[index-1]->len == n2 >> (32 - p3[index-1]->len)))

{

success++;

break;

}

else

{

if(ptr->next == NULL)

fail++;

}

}

}

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

{

struct node \*p1[256];

struct node \*p2[4096];

struct node \*p3[4096];

f1(argv[1], p1, p2, p3);

search(argv[2], p1, p2, p3);

return 0;

}

Compilation:

gcc -o hw8 hw8.c

^

Execution:

./hw8 prefix\_10K.txt trace\_IPaddress\_100K.txt insert\_1K.txt delete\_1K.txt

Output: