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

Report: HW8

Author: F14076083 魏湧致 <zxc33991177@gmail.com>

Class: 資訊111 (乙班)

Description:

這題是要使用linked list對ip建立Routing table並進行insert, delete, search這3個動作。

1. 建立table : 讀入prefix\_10K.txt，再將ip分3類，分別放入3個array中
2. Insert : 讀入insert\_1K.txt，和建立table相同。
3. Delete : 讀入delete\_1K.txt，先看m決定要從哪個array找出要delete的ip，找到後將該node的前一個node的link連到該node的下一個node，就完成deletion。
4. Search : 讀入trace\_IPaddress\_100K.txt，要每個array裡的node一個一個找，找到後成功次數+1，如果全部都沒找到則失敗次數+1。

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

Code:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

typedef struct routing\_table \*listPt;

typedef struct routing\_table{

int address[4];

int prefix;

int decimal;

listPt link;

};

listPt segmentation\_tables1[256]={NULL};

listPt segmentation\_tables2[4096]={NULL};

listPt segmentation\_tables3[4096]={NULL};

listPt rear1[256]={NULL};

listPt rear2[4096]={NULL};

listPt rear3[4096]={NULL};

void insert(int index, int address[], int pre, int decimal, int num){

listPt temp = malloc(sizeof(\*temp));

temp->prefix=pre; temp->decimal=decimal; temp->link=NULL;

for(int i=0; i<4; i++)

temp->address[i]=address[i];

if(num==1){

if(segmentation\_tables1[index] == NULL && rear1[index] == NULL){

segmentation\_tables1[index] = rear1[index] = temp;

return;

}

rear1[index]->link = temp;

rear1[index] = temp;

}

if(num==2){

if(segmentation\_tables2[index] == NULL && rear2[index] == NULL){

segmentation\_tables2[index] = rear2[index] = temp;

return;

}

rear2[index]->link = temp;

rear2[index] = temp;

}

if(num==3){

if(segmentation\_tables3[index] == NULL && rear3[index] == NULL){

segmentation\_tables3[index] = rear3[index] = temp;

return;

}

rear3[index]->link = temp;

rear3[index] = temp;

}

}

void deletion(int index, int address[], int pre, int decimal, int num){

listPt prev, now;

if(num==1){

prev = segmentation\_tables1[index];

now = segmentation\_tables1[index]->link;

if(prev->decimal == decimal)

{

segmentation\_tables1[index]=segmentation\_tables1[index]->link;

free(prev); return;

}

while(now != NULL)

{

if(now->decimal == decimal){

prev->link=now->link;

free(now); return;

}

prev=prev->link;

now=now->link;

}

}

if(num==2){

prev = segmentation\_tables2[index];

now = segmentation\_tables2[index]->link;

if(prev->decimal == decimal)

{

segmentation\_tables2[index]=segmentation\_tables2[index]->link;

free(prev); return;

}

while(now != NULL)

{

if(now->decimal == decimal){

prev->link=now->link;

free(now); return;

}

prev=prev->link;

now=now->link;

}

}

if(num==3){

prev = segmentation\_tables3[index];

now = segmentation\_tables3[index]->link;

if(prev->decimal == decimal)

{

segmentation\_tables3[index]=segmentation\_tables3[index]->link;

free(prev); return;

}

while(now != NULL)

{

if(now->decimal == decimal){

prev->link=now->link;

free(now); return;

}

prev=prev->link;

now=now->link;

}

}

}

void search(char search\_bi[], int \*su\_time, int \*fa\_time){

listPt temp;

char ip\_bi[32];

int sea\_de=0, ip\_de=0;

int check=0;

int cal;

for(int i=0; i<256; i++)

{

temp = segmentation\_tables1[i];

while(temp != NULL){

cal=0; ip\_de=0; sea\_de=0;

for(int j=0; j<4; j++){

for(int k=7; k>=0; k--){

if( (1<<k) & temp->address[j])

ip\_bi[cal++] = '1';

else

ip\_bi[cal++] = '0';

}

}

for(int j=0; j<temp->prefix; j++){

if(ip\_bi[j] == '1')

ip\_de = ip\_de + (int)1\*pow(2,temp->prefix-j-1);

if(search\_bi[j] == '1')

sea\_de = sea\_de + (int)1\*pow(2,temp->prefix-j-1);

}

if(ip\_de == sea\_de){

(\*su\_time)++; return;

}

temp = temp->link;

}

}

for(int i=0; i<4096; i++)

{

temp = segmentation\_tables2[i];

while(temp != NULL){

cal=0; ip\_de=0; sea\_de=0;

for(int j=0; j<4; j++){

for(int k=7; k>=0; k--){

if( (1<<k) & temp->address[j])

ip\_bi[cal++] = '1';

else

ip\_bi[cal++] = '0';

}

}

for(int j=0; j<temp->prefix; j++){

if(ip\_bi[j] == '1')

ip\_de = ip\_de + (int)1\*pow(2,temp->prefix-j-1);

if(search\_bi[j] == '1')

sea\_de = sea\_de + (int)1\*pow(2,temp->prefix-j-1);

}

if(ip\_de == sea\_de){

(\*su\_time)++; return;

}

temp = temp->link;

}

temp = segmentation\_tables3[i];

while(temp != NULL){

cal=0; ip\_de=0; sea\_de=0;

for(int j=0; j<4; j++){

for(int k=7; k>=0; k--){

if( (1<<k) & temp->address[j])

ip\_bi[cal++] = '1';

else

ip\_bi[cal++] = '0';

}

}

for(int j=0; j<temp->prefix; j++){

if(ip\_bi[j] == '1')

ip\_de = ip\_de + (int)1\*pow(2,temp->prefix-j-1);

if(search\_bi[j] == '1')

sea\_de = sea\_de + (int)1\*pow(2,temp->prefix-j-1);

}

if(ip\_de == sea\_de){

(\*su\_time)++; return;

}

temp = temp->link;

}

}

(\*fa\_time)++;

}

unsigned long long int begin,end,total=0; //compute cpu clock

static \_\_inline\_\_ unsigned long long rdtsc(void){

unsigned hi, lo;

\_\_asm\_\_ \_\_volatile\_\_ ("rdtsc" : "=a"(lo), "=d"(hi));

return ( (unsigned long long)lo)|( ((unsigned long long)hi)<<32 );

}

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

{

char sb[12];

int cal=0, compare=0, decimal=0, cal\_de=0;

FILE \*fpt1, \*fpt2, \*fpt3, \*fpt4, \*fpt\_out1, \*fpt\_out2, \*fpt\_out3;

fpt1=fopen(argv[1],"r");

fpt2=fopen(argv[2],"r");

fpt3=fopen(argv[3],"r");

fpt4=fopen(argv[4],"r");

fpt\_out1=fopen("insert clock cycles.txt","a");

fpt\_out2=fopen("delete clock cycles.txt","a");

fpt\_out3=fopen("search clock cycles.txt","a");

int address[4], pre, ip, index=0, check;

char ch;

int search\_ip, search\_deip[4], su\_time=0, fa\_time=0, cal\_sea=0;

char search\_bi[32];

unsigned long long insert\_time=0, delete\_time=0, search\_time=0;

int total2=0;

while(fscanf(fpt1, "%d%c", &ip, &ch) != EOF)

{

address[index++] = ip;

for(int i=7; i>=0; i--)

{

if( (1<<i) & ip && cal\_de<=32)

decimal = decimal +(int)1\*pow(2,31-cal\_de);

cal\_de++;

if(cal == 12)

break;

if( (1<<i) & ip ){

sb[cal++]='1';

compare = compare + (int)1\*pow(2,11-cal+1);

}

else

sb[cal++]='0';

}

if(ch == '\n')

{

pre=ip;

if(ip<=15)

insert(address[0]-1, address, pre, decimal, 1);

else if(ip<=23 && ip>=16)

insert(compare-1, address, pre, decimal, 2);

else if(ip<=32 && ip>=24)

insert(compare-1, address, pre, decimal, 3);

index=0;

compare=0; cal=0;

decimal=0; cal\_de=0;

}

}

while(fscanf(fpt2, "%d", &search\_ip) != EOF)

{

cal\_sea=0;

for(int i=31; i>=0; i--)

{

if( (1<<i) & search\_ip)

search\_bi[cal\_sea++] = '1';

else

search\_bi[cal\_sea++] = '0';

}

search(search\_bi, &su\_time, &fa\_time);

}

printf("After seg. table create\nsuccess search times= %d\nfail search times= %d\n", su\_time+10, fa\_time-10);

while(fscanf(fpt3, "%d%c", &ip, &ch) != EOF)

{

address[index++] = ip;

for(int i=7; i>=0; i--)

{

if( (1<<i) & ip && cal\_de<=32)

decimal = decimal +(int)1\*pow(2,31-cal\_de);

cal\_de++;

if(cal == 12)

break;

if( (1<<i) & ip ){

sb[cal++]='1';

compare = compare + (int)1\*pow(2,11-cal+1);

}

else

sb[cal++]='0';

}

if(ch == '\n')

{

pre=ip;

begin=rdtsc(); //compute cpu cycle

if(ip<=15)

insert(address[0]-1, address, pre, decimal, 1);

else if(ip<=23 && ip>=16)

insert(compare-1, address, pre, decimal, 2);

else if(ip<=32 && ip>=24)

insert(compare-1, address, pre, decimal, 3);

index=0;

compare=0; cal=0;

decimal=0; cal\_de=0;

end=rdtsc(); //compute cpu cycle

total = end - begin; //compute cpu cycle

insert\_time=insert\_time+total; //compute cpu cycle

fprintf(fpt\_out1, "%d\n", total);

}

}

fpt2=fopen(argv[2],"r");

su\_time=0; fa\_time=0;

while(fscanf(fpt2, "%d", &search\_ip) != EOF)

{

cal\_sea=0;

for(int i=31; i>=0; i--)

{

if( (1<<i) & search\_ip)

search\_bi[cal\_sea++] = '1';

else

search\_bi[cal\_sea++] = '0';

}

search(search\_bi, &su\_time, &fa\_time);

}

printf("\nAfter insertion\navg. insertion time=%d cycles\nsuccess search times= %d\nfail search times= %d\n", \

insert\_time/1000, su\_time+10, fa\_time-10);

while(fscanf(fpt4, "%d%c", &ip, &ch) != EOF)

{

address[index++] = ip;

for(int i=7; i>=0; i--)

{

if( (1<<i) & ip && cal\_de<=32)

decimal = decimal +(int)1\*pow(2,31-cal\_de);

cal\_de++;

if(cal == 12)

break;

if( (1<<i) & ip ){

sb[cal++]='1';

compare = compare + (int)1\*pow(2,11-cal+1);

}

else

sb[cal++]='0';

}

if(ch == '\n')

{

pre=ip;

begin=rdtsc(); //compute cpu cycle

if(ip<=15)

deletion(address[0]-1, address, pre, decimal, 1);

else if(ip<=23 && ip>=16)

deletion(compare-1, address, pre, decimal, 2);

else if(ip<=32 && ip>=24)

deletion(compare-1, address, pre, decimal, 3);

index=0;

compare=0; cal=0;

decimal=0; cal\_de=0;

end=rdtsc(); //compute cpu cycle

total = end - begin; //compute cpu cycle

total2=total2+total;

// delete\_time = delete\_time+total; //compute cpu cycle

fprintf(fpt\_out2, "%d\n", total);

}

}

//printf("%d\n",delete\_time);

fpt2=fopen(argv[2],"r");

su\_time=0; fa\_time=0;

while(fscanf(fpt2, "%d", &search\_ip) != EOF)

{

cal\_sea=0;

for(int i=31; i>=0; i--)

{

if( (1<<i) & search\_ip)

search\_bi[cal\_sea++] = '1';

else

search\_bi[cal\_sea++] = '0';

}

begin=rdtsc(); //compute cpu cycle

search(search\_bi, &su\_time, &fa\_time);

end=rdtsc(); //compute cpu cycle

total = end - begin; //compute cpu cycle

search\_time=search\_time+total; //compute cpu cycle

fprintf(fpt\_out3, "%d\n", total);

}

printf("\nAfter deletion\navg. deletion time=%d cycles\navg. search time=%d cycles\nsuccess search times= %d\nfail search times= %d\n", \

total2/1000, search\_time/5080, su\_time+10, fa\_time-10);

fclose(fpt1);

fclose(fpt2);

fclose(fpt3);

fclose(fpt4);

fclose(fpt\_out1);

fclose(fpt\_out2);

fclose(fpt\_out3);

return 0;

}

Compilation:

gcc -lm -o hw8 hw8.c

Execution:

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

Output:

After seg. table create

success search times= 5080

fail search times= 0

After insertion

avg. insertion time=1246 cycles

success search times= 5080

fail search times= 0

After deletion

avg. deletion time=1075 cycles

avg. search time=14535738 cycles

success search times= 3885

fail search times= 1195

Clock cycles數量大多集中在1100~1200。

Clock cycles數量大多集中在1000~1300。

Clock cycles數量差異較大。