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

Report: hw8

Author: F74046022 陳冠仁 <jeremy851004@gmail.com>

Class: 乙班

Description:

資料結構：array → linked list(單向、降冪(prefix length))

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

#include <stdio.h>

#include <stdlib.h>

int i;

//loop var

int X, Y, IP[5];

//X : significant bits in decimal (8)

//Y : significant bits in decimal (12)

//IP[] : temporary storage for IP/prefix

unsigned long long int begin, end;

inline unsigned long long int rdtsc()

{

unsigned long long int a;

asm volatile("rdtsc":"=A"(a));

return a;

}

struct node{

int IP[5];

struct node \*next;

};

typedef struct node node;

void init\_dtbs(node \*dtbs1[], node \*dtbs2[], node \*dtbs3[])

{

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

dtbs1[i]=NULL;

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

{

dtbs2[i]=NULL;

dtbs3[i]=NULL;

}

return;

}

node \*insr(node \*list, int IP[])

{

node \*prev, \*cur, \*new=malloc(sizeof(node));

for(i=0; i<5; i++)

new->IP[i]=IP[i];

new->next=NULL;

//set new node

if(list==NULL)

return new;

for(prev=NULL, cur=list; cur!=NULL && IP[4]<cur->IP[4]; prev=cur, cur=cur->next);

//descendiing order(number of prefix)

if(prev==NULL)

{

new->next=list;

return new;

}

if(cur==NULL)

{

prev->next=new;

return list;

}

prev->next=new;

new->next=cur;

return list;

}

void file\_insr(FILE \*f, node \*dtbs1[], node \*dtbs2[], node \*dtbs3[], FILE \*csv)

{

while(fscanf(f,"%d.%d.%d.%d/%d",&IP[0],&IP[1],&IP[2],&IP[3],&IP[4])!=EOF)

{

X=IP[0];

Y=IP[0]\*16+(IP[1]&240)/16;

begin=rdtsc();

if(IP[4]<=15)

dtbs1[X]=insr(dtbs1[X], IP);

else if(IP[4]<=24)

dtbs2[Y]=insr(dtbs2[Y], IP);

else

dtbs3[Y]=insr(dtbs3[Y], IP);

end=rdtsc();

if(csv!=NULL)

fprintf(csv,"%u\n",end-begin);

}

return;

}

void srch(node \*list, int IP[], int m[])

{

node \*cur;

int p, q, match;

for(cur=list; cur!=NULL; cur=cur->next)

{

match=1;

p=cur->IP[4]/8;

q=8-(cur->IP[4]%8);

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

if(IP[i]!=cur->IP[i])

{match=0; break;}

if(match && (IP[p]>>q)==(cur->IP[p]>>q))

{

for(i=0; i<5; i++)

m[i]=cur->IP[i];

break;

}

}

return;

}

void file\_srch(FILE \*f, node \*dtbs1[], node \*dtbs2[], node \*dtbs3[], FILE \*rslt, FILE \*csv)

{

int m[5];

//m[] : storage for match

while(fscanf(f,"%d.%d.%d.%d",&IP[0],&IP[1],&IP[2],&IP[3])!=EOF)

{

X=IP[0];

Y=IP[0]\*16+(IP[1]&240)/16;

for(i=0; i<5; i++)

m[i]=-1;

begin=rdtsc();

srch(dtbs3[Y], IP, m);

if(m[0]==-1)

srch(dtbs2[Y], IP, m);

if(m[0]==-1)

srch(dtbs1[X], IP, m);

if(m[0]==-1)

for(i=0; i<5; i++)

m[i]=0;

end=rdtsc();

fprintf(rslt,"%d.%d.%d.%d/%d\n",m[0],m[1],m[2],m[3],m[4]);

fprintf(csv,"%u\n",end-begin);

}

return;

}

node \*delt(node \*list, int IP[])

{

node \*cur, \*prev;

for(prev=NULL, cur=list; cur!=NULL && !(cur->IP[0]==IP[0] && cur->IP[1]==IP[1] && cur->IP[2]==IP[2] && cur->IP[3]==IP[3] && cur->IP[4]==IP[4]); prev=cur, cur=cur->next);

if(cur==NULL)

return list;

if(prev==NULL)

list=list->next;

else

prev->next=cur->next;

free(cur);

return list;

}

void file\_delt(FILE \*f, node \*dtbs1[], node \*dtbs2[], node \*dtbs3[], FILE \*csv)

{

while(fscanf(f,"%d.%d.%d.%d/%d",&IP[0],&IP[1],&IP[2],&IP[3],&IP[4])!=EOF)

{

X=IP[0];

Y=IP[0]\*16+(IP[1]&240)/16;

begin=rdtsc();

if(IP[4]<=15)

dtbs1[X]=delt(dtbs1[X], IP);

else if(IP[4]<=24)

dtbs2[Y]=delt(dtbs2[Y], IP);

else

dtbs3[Y]=delt(dtbs3[Y], IP);

end=rdtsc();

fprintf(csv,"%u\n",end-begin);

}

return;

}

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

{

FILE \*f\_data=fopen("/home/data/hw8/IPv4\_400k.txt", "r"),

\*f\_srch=fopen("/home/data/hw8/IPv4\_search.txt", "r"),

\*f\_insr=fopen("/home/data/hw8/IPv4\_insert.txt", "r"),

\*f\_delt=fopen("/home/data/hw8/IPv4\_delete.txt", "r"),

\*f\_rslt=fopen("result.txt", "w"),

\*csv\_srch=fopen("search.csv","w"),

\*csv\_insr=fopen("insert.csv","w"),

\*csv\_delt=fopen("delete.csv","w");

//csv : for generating graphs

if(f\_data==NULL || f\_srch==NULL || f\_insr==NULL || f\_delt==NULL)

{

printf("ERROR\n");

return 0;

}

node \*dtbs1[256], \*dtbs2[4096], \*dtbs3[4096];

init\_dtbs(dtbs1, dtbs2, dtbs3);

file\_insr(f\_data, dtbs1, dtbs2, dtbs3, NULL);

file\_srch(f\_srch, dtbs1, dtbs2, dtbs3, f\_rslt, csv\_srch);

file\_insr(f\_insr, dtbs1, dtbs2, dtbs3, csv\_insr);

file\_delt(f\_delt, dtbs1, dtbs2, dtbs3, csv\_delt);

fclose(f\_data);

fclose(f\_srch);

fclose(f\_insr);

fclose(f\_delt);

fclose(f\_rslt);

fclose(csv\_srch);

fclose(csv\_insr);

fclose(csv\_delt);

return 0;

}

Compilation:

gcc -o hw8 hw8.c

Execution:

./hw8

Output:

hw8.c:87:1: error: expected ‘;’ before ‘}’ token

}

^

hw8.c:232:1: error: expected declaration or statement at end of input

}

^

hw8.c: In function ‘main’:

hw8.c:47:2: error: expected ‘;’ before ‘printf’

printf("\n\n");