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

Report: HW3

Author: F74071140 謝育萱 <[shirleysensible@gmail.com](mailto:shirleysensible@gmail.com)>

Class: 甲班

Description:

這次運用大量的atoi及指標,大致上的概念是利用bitwise修改整數的位元值，使其整數值改變，但bitwise只能用於整數型態，故如要修改float的位元需用整數型別的指標進行運算；此外，這次還需要清楚知道各資料型別所佔的位元組大小，因為用bitwise進行各位元比對時，如果給予的空間不夠的話會產生溢位現象，使輸出值錯誤；最後須注意的是各資料型別的輸出符號，光是整數就有分unsign、sign、八進位、十進位、十六進位等形式，故要清楚對電腦下達指令才不會產生bug。

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

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

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

{

int i=0, n=0;

switch(atoi(argv[1]))

{

case 1:

{

float f=0.0; //將float值初始化

int \*k=(int\*)&f; //f轉為整數指標,並把其位址存於k

int a[32];

for(i=0;i<32;i++)

{

if(argv[2][i]=='1') //把字元argv[2][i]轉為整數陣列

a[i]=1;

else

a[i]=0;

}

for(i=0;i<32;i++)

{

n|=a[i]<<(31-i); //a[i]從n(為0)的最左位元比對,若a[i]為1其位元由0轉為1

(\*k)|=a[i]<<(31-i); //同上,但從float位址更改其bit值

}

printf("integer:%d\n",n); //print n

printf("float:%f\n",f); //print f

break;

}

case 2:

{

int num=atoi(argv[2]); //用atoi將字串轉為整數

printf("32-bit:");

for(i=31;i>-1;i--)

{

n=num&(1<<i); //1從num最左的bit比對,若同為1,n的該bit存1,反之存0

n>>=i; //將該bit值右移至底,使n值變為該bit值,利用迴圈印出

printf("%d",abs(n)); //print n的絕對值

}

printf("\n");

break;

}

case 3:

{

float a=atof(argv[2]); //用atof將字串轉為float

int \*x=(int\*)&a; //將a轉整數指標並存取其位址

printf("32-bit:");

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

{

n=(\*x)&(1<<i); //用int來表示其位元的\*x中每個位元依序跟1做&運算

n>>=i; //將該bit值右移至底,使n值變為該bit值,利用迴圈印出

printf("%d",abs(n)); //print n的絕對值

}

printf("\n");

break;

}

case 4:

{

long long b[64];//為避免溢位使用long long宣告，底下有long long皆同此理

for(i=0;i<64;i++)

{

if (argv[2][i]=='1') //若字元為1則將數值1存入陣列，若為0則存入0

b[i]=1;

else

b[i]=0;

}

long long x=0;

for(i=63;i>=0;i--)

{

x|=(long long)b[i]<<(63-i); //x從最左bit開始比對,若b[i]為1則其位元轉為1

}

double \*d=(double\*)&x; //將long long x轉double

printf("integer:%lld\n",x);

printf("double:%f\n",\*d);

break;

}

case 5:

{

long int num=atol(argv[2]); //用atol將字串argv[2]轉為long int

printf("64-bit:");

for(i=63;i>=0;i--)

{

n=num&((long int)1<<i); //1以long型別存取,從最左開始比對,若同為1則該bit存1,反之存0

n>>=i; //將該bit值右移至底,使n值變為該bit值,利用迴圈印出

printf("%d",abs(n)); //print n的絕對值

}

printf("\n");

break;

}

case 6:

{

double d=strtod(argv[2],NULL); //將字串轉換為double

long long \*y=(long long\*)&d; //運用指標將double位元以long long表示

printf("64-bit:");

for(i=63;i>=0;i--)

{

long long bit=(\*y)&((long long)1<<i); //用long long來表示其位元組成的\*y，將其每個位元依序跟1做&運算

bit>>=i; //將該bit值右移至底,使其值變為該bit值,利用迴圈印出

printf("%lld",bit);

}

printf("\n");

}

}

return 0;

}

Compilation:

gcc hw3.c -o hw3

Execution:

F74071140@c-2018-2:~/hw3> ./hw3 1 111

integer:-536870912

float:-36893488147419103232.000000

F74071140@c-2018-2:~/hw3> ./hw3 2 111

32-bit:00000000000000000000000001101111

F74071140@c-2018-2:~/hw3> ./hw3 3 111

32-bit:01000010110111100000000000000000

F74071140@c-2018-2:~/hw3> ./hw3 4 11100000000000000000000000000000000000000000000000000000000000

integer:-2305843009213693952

double:-26815615859885194199148049996411692254958731641184786755447122887443528060147093953603748596333806855380063716372972101707507765623893139892867298012168192.000000

F74071140@c-2018-2:~/hw3> ./hw3 5 111

64-bit:0000000000000000000000000000000000000000000000000000000001101111

F74071140@c-2018-2:~/hw3> ./hw3 6 111

64-bit:0100000001011011110000000000000000000000000000000000000000000000

F74071140@c-2018-2:~/hw3>