As we know when we print size it should be unsigned

%d we will use %lu.

#include<stdio.h>

int main(){

printf("%d\n", sizeof(int));

printf("%lu\n", sizeof(int));

return 0;

}

But On windows there is a difference between size of long unsigned int from Linux.

#include<stdio.h>

int main(){

printf("%lu\n", sizeof(long unsigned int));

return 0;

}

//4 on windows

//8 on Linux

Now

#include<stdio.h>

int main(){

printf("%lu\n", sizeof(long unsigned int));

printf("%llu\n", sizeof(long long unsigned int));

return 0;

}

/\*

Windows:

--------

4

8

Linux

-----

Warning: (printf("%llu\n", sizeof(long long unsigned int));)

format %llu expects argument of type long long unsigned in but argument 2 has type long unsigned int

8

8

\*/

This is based on the c standard set for compilers

You can check on google

Search: C standard types IBM (size may very based on platform)

Search : c arithmetic type cpp reference (en.cppreference.com)

Now we will Check the address related issues

#include<stdio.h>

int main(){

int x;

int y;

char m;

int z;

char g;

char k;

char u;

int p;

printf("Address of int type x is %p\n", &x);

printf("Address of int type x is %llu\n", &x);

printf("Address of int type y is %llu\n", &y);

printf("Address of char type m is %llu\n", &m);

printf("Address of int type z is %llu\n", &z);

printf("Address of char type g is %llu\n", &g);

printf("Address of char type k is %llu\n", &k);

printf("Address of char type u is %llu\n", &u);

printf("Address of int type p is %llu\n", &p);

return 0;

}

/\*

Address of int type x is 0061FF1C // this is hexa decimal

Address of int type x is 18446744065126039324

Address of int type y is 18446744065126039320

Address of char type m is 18446744065126039319

Address of int type z is 18446744065126039312

Address of char type g is 18446744065126039311

Address of char type k is 18446744065126039310

Address of char type u is 18446744065126039309

Address of int type p is 18446744065126039304

\*/

When we will print with %p format specifiled then it will print hexa decimal value

Example

16 / 65 / 1 output will be 41 that is the hexa value of 65

4

16 / 75 / 11 output will be 4B that is the hexa value of 65

4

0 1 2 3 4 5 6 7 8 9 A B C D E F - Hexa code

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 - Decimal values

So we will pring with llu long long unsigned int type. (lu in case of linux)

Now see the pattern **(data is storing in Stack format in both windows and Linux)**

Windows ( Order is in decreasing order)

39327

39326

39325

39324 - x

—-------  
39323

39322

39321

39320 - y

—------

39319 - m

—-----

Gap of 3

—-----

39315

39314

39313

39312 z

—------

39311 g

—------

39310 k

—------

39309 u

—------

Gap of 1

—------

39307

39306

39305

39304 p

Linux

776228 m

—-------

776229 g

—--------

776230 k

—-------

776231 u

—-------

776232 x

776233

776234

776235

—-------

776236 y

776237

776238

776239

—-------

776240 z

776241

776242

776243

—-------

776244 p

776245

776246

776247

Here recording done as well as in increasing order but Stack.

Why thesis gaps are coming will discuss in upcoming lectures.

#include<stdio.h>

void tom();

void sam(int);

void cat(int, int);

int main(){

int x, y;

printf("Main Start\n");

printf("Address of int type x is %llu\n", &x);

printf("Address of int type x is %llu\n", &y);

tom();

sam(10);

printf("Main Ends\n");

return 0;

}

void tom(){

int a;

char b;

printf("tom Start\n");

printf("Address of int type a is %llu\n", &a);

printf("Address of char type b is %llu\n", &b);

printf("Tom Ends\n");

}

void sam(int p){

int q;

char r;

printf("Sam Start\n");

printf("Address of int type p is %llu\n", &p);

printf("Address of int type q is %llu\n", &q);

printf("Address of char type r is %llu\n", &r);

cat(100,100);

printf("Sam Ends\n");

}

void cat(int p, int q){

int r;

char s;

printf("cat Start\n");

printf("Address of int type p is %llu\n", &p);

printf("Address of int type q is %llu\n", &q);

printf("Address of int type r is %llu\n", &r);

printf("Address of char type s is %llu\n", &s);

printf("cat Ends\n");

}

/\*

Main Start

Address of int type x is 27583808989691676

Address of int type x is 27583808989691672

tom Start

Address of int type a is 14161709772177132

Address of char type b is 14161709772177131

Tom Ends

Sam Start

Address of int type p is 14161709772177152

Address of int type q is 14161709772177132

Address of char type r is 14161709772177131

cat Start

Address of int type p is 14161709772177104

Address of int type q is 14161709772177108

Address of int type r is 14161709772177084

Address of char type s is 14161709772177083

cat Ends

Sam Ends

Main Ends

\*/

Same address can appear multiple time because once function end then the address is released can be reused.

DO NOT SEARCH FOR PATTERNS