**C++ Likely and Unlikely attributes**

# What and Why ?

C++ likely and unlikely attributes came into existence with the release of C++ 20.  
In our C++ code if we know which lines of code would most probably be executed during our code execution then we can use likely and unlikely attributes , which is our way of saying to the compiler which lines of code would be most probably be executed and thus compiler can optimize it's execution ( by code scheduling ) and that would result in faster code execution.

Before the release of C++ 20 we had compiler based extensions like \_\_builtin\_expect() ( which is included in gcc compiler ) but they are compiler dependent and makes code less portable . so to remove this anomaly C++ 20 came with a concept of likely and unlikely attributes which made such code optimizations universal rather then being dependent on compiler.

## Syntax and usage

let's see how it can applied in if else statement

int a = 5 ;

if( a > 3 ) [[likely]] {

std::cout<<"a is greater then 3"<<std::endl;

}else[[unlikely]]{

std::cout<<"a is smaller then 3"<<std::endl;

}

// or

if( a < 3) [[unlikely]]{

std::cout<<"a is smaller then 3"<<std::endl;

}

// it can also be applied in switch statements as shown below

switch(i){

[[unlikely]] case 1 :

// do something

[[likely]] case 2 :

// do something else

}

C++

Copy

### Caution

if we are not sure about some lines of code being executed more over the other then we should avoid using likely or unlikely attributes because if use them giving wrong hints to compiler these attributes can make our code execution slower because of inefficient scheduling done by compiler.

## Use Case Example

let's take a real world example where this optimization can be used ----

let say we want to do modulus operation on a vector of size 1024 with a value of 224 , this vector contains random but equally distributed integers between 1 and 255 .

void doModulus( vector<int> &vec , int mod ){

// here the value of mod we are passing is 224

for( int i = 0 ; i<vec.size() ; i++ )

{

vec[i] = vec[i] % mod ;

}

}

C++

Copy

if we run this code and find the execution time then we get around 2244 nano seconds.

% operation is a computationally heavy operation for cpu , so let's try to optimize our code.

we know that most our data in input vector is below 224 (because of random equal distribution ) so we can use an if check along with likely and unlikely attributes to avoid using modulus operation as much as possible.

Now , if we change our code like this then -

void doModulus( vector<int> &vec , int mod ){

// here the value of mod we are passing is 224

for( int i = 0 ; i<vec.size() ; i++ )

{

if( vec[i] >= mod ) [[unlikely]]{

// so we are prioritizing else statement here

vec[i] = vec[i] % mod ;

}else {

vec[i] = vec[i] ;

}

}

}

C++

Copy

The execution time for running the above function is 633 nano seconds , which is significantly faster compared to 2244 nano seconds previously obtained.

so , this is one of the example where we can use likely and unlikely attributes along with if else statements to significantly reduce our code running time.

## Conclusion

likely and unlikely is introduced to bring standardization to compiler based optimizations . Earlier to do such optimizations we had seperate extensions for seperate compilers which made our code compiler dependent and less portable . likely and unlikely have been introdced to remove this anomaly.

likely and unlikely attributes is our way of saying to the compiler that which code is most likely to be executed so compiler can schedule accordingly and result in a overall faster code execution.  
We should be cautious using likely and unlikely if we are not very sure because otherwise we would have serious performance degradations.