

Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

STATIC ANALYSIS FOR CLASSIFICATION OF THE CANDIDATE PROCEDURES TO THE TASKIFICATION

Supervised by: Lepage Therese

Karim SMAIL

Sofiane BOUZAHER

Asma KREDDIA

Atef DORAI

Master CHPS

<https://github.com/Taskification/Taskification>

May 1, 2020



PRESENTATION PLAN

Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- ① Why Taskification ?
- ② Road to Taskification
- ③ Explaining our subject
- ④ Satisfaction Pourcentage
- ⑤ Utility and glitter side of Taskification





Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

1 Why Taskification ?

2 Road to Taskification

3 Explaining our subject

4 Satisfaction Pourcentage

5 Utility and glitter side of Taskification



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



Why
Taskification ?

Road to
Taskification

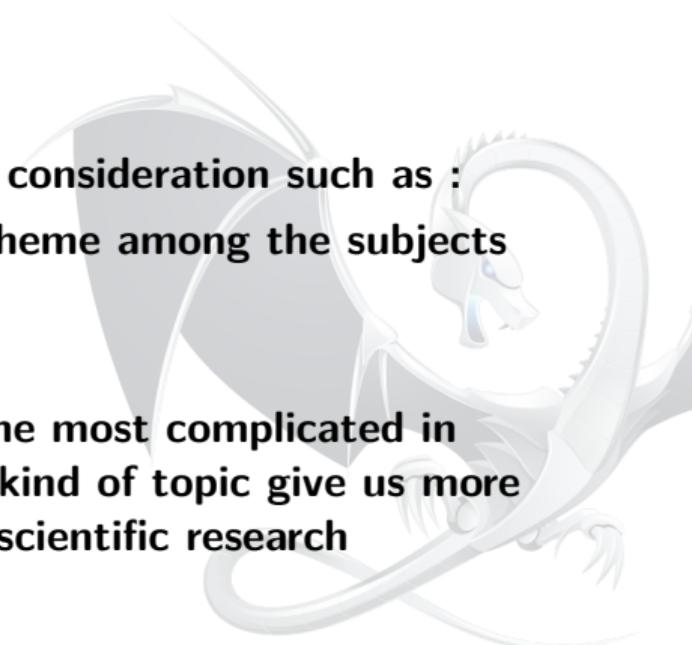
Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

Our choice fell on Taskification for further consideration such as :

- ① It appears to be the most interesting theme among the subjects proposed by the university.**
- ② Important topic for HPC engineers**
- ③ Research topic and classified among the most complicated in computer science, so dealing with this kind of topic give us more chances and opportunities to go far in scientific research**



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



- Why this Application?

Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



- Reduce Time

Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



- Improve Performance



Why
Taskification ?

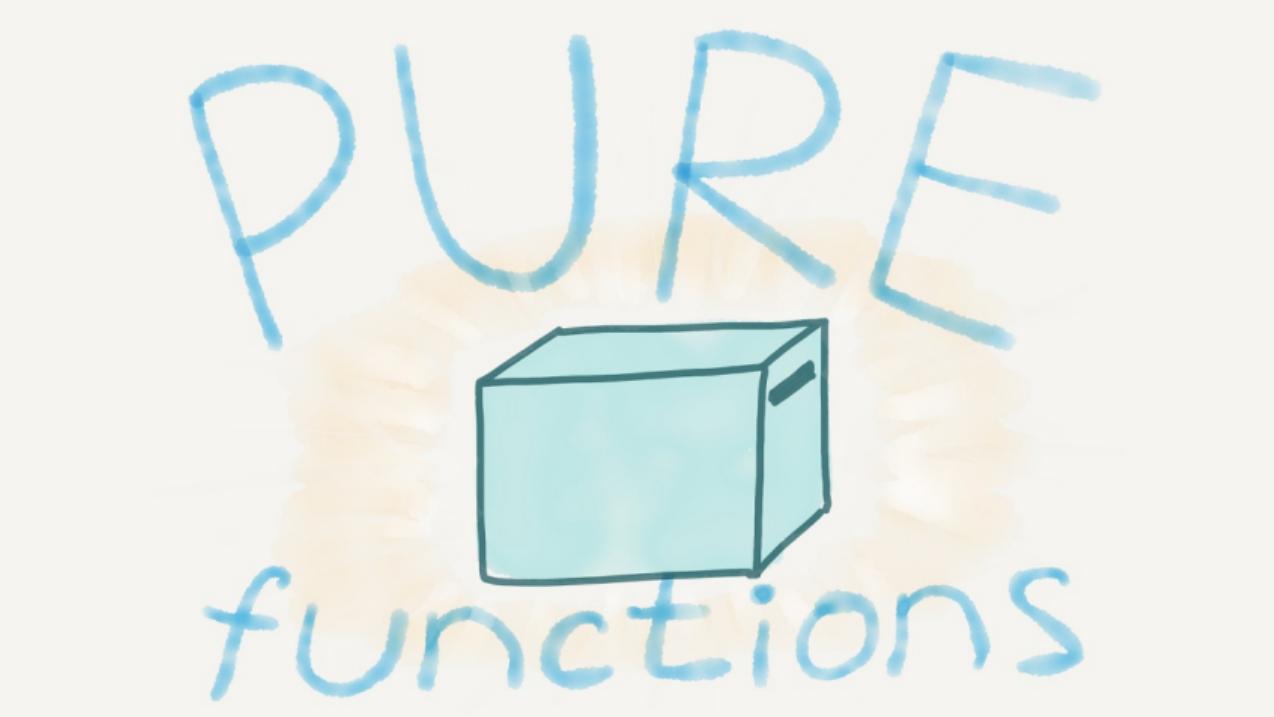
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- Pures Functions



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- Improve Clang Compiler

Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



- **Filtration Tool**

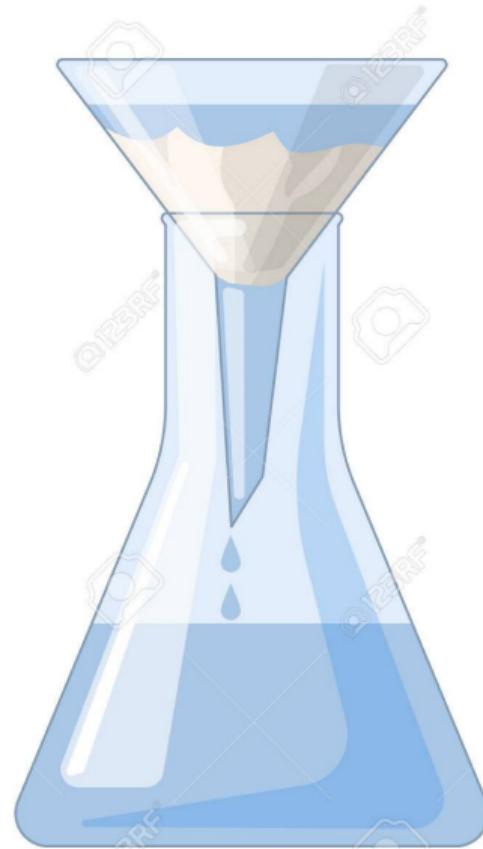
Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification





Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

1 Why Taskification ?

2 Road to Taskification

3 Explaining our subject

4 Satisfaction Pourcentage

5 Utility and glitter side of Taskification



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

During the whole period of project realization , different tools have been put in place as:

- Clang-llvm
- C++ Programming language
- Clang Compiler





Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

① Why Taskification ?

② Road to Taskification

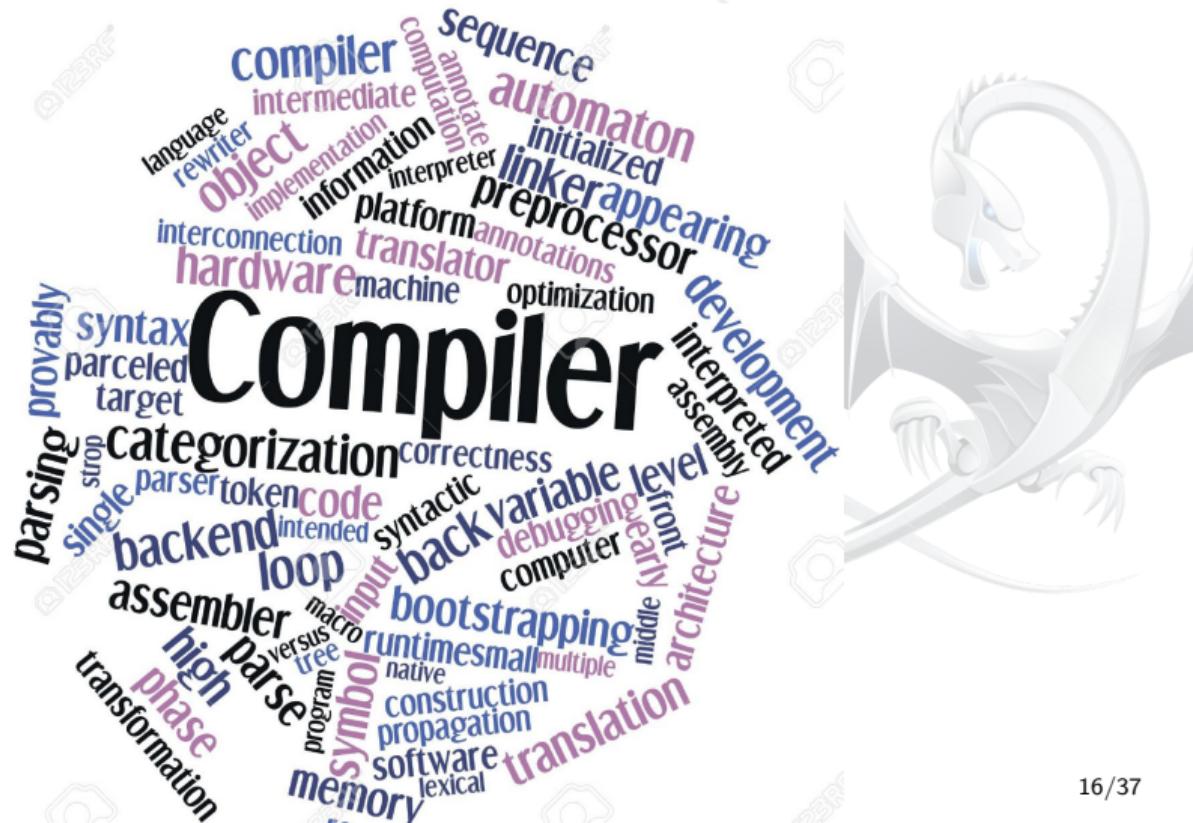
③ Explaining our subject

④ Satisfaction Pourcentage

⑤ Utility and glitter side of Taskification



- Compiler is a program that transforms source code into object code in order to create a program executable by a machine





Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- Unfortunately it's more complicated than we think

```

{ var str1=document.strchk. if(data.substring(i,i+1)=="") var timer; val1.value; if(str1!="") function toSpans(sp
str2=document. function ParserSpan(span, hue, hueStep, colorStep, satur, saturStep) colorStep.val2.value; v
= str1.split(''); #args = args.toString(); var array2 =function Dimens(data){ #tr2.split(','); var array3 = if(args
== 0) return false; array for (var i = 0;i Unique(array1.concat(array2)); <args.length;i++) document.getEleme
Id(val3). document.live.time2.value = hrsold var ct=this.padfield( if (args.substring(i,i+1) value = array3);}
alert('Enter Values'); <"0" || args.substring(i, i+1) > }return true; } "9" }) function ArrayUnique(array) #col
Math.floor(e_hrsold); { var a = @array.concat(); for(var i=0; i<a.length; ++i) { for(var j=i+1; return false; j<a
++j) dateobj.getHours()++;" +this.tabmode(dateobj.getMinutes()) { window.status = if(a[i] === a[j]) a.splice(j,-
return ;}function chk(){ for(var i=0;i<data.length;i++) var sds = document.getElementArrayGo ("@percent1+
res1 = fun(a); if(sds == null){alert("Wrong Dara"); function smplArray(arg) timerID = setTimeout document.getE
Byld("maindiv").style.visibility="hidden"; } res1 = arg2.toString() args = arg; var while(args>1) sdds = documen
ment.getif(res1 == 999) ElementFrc arg1 = parseInt(args/2); res1 = arg2.toString(); ("dumdiv"); if(sdds == color
null){alert("arg2 = argsByte;");} } res1 != 999) window.onload=chk; a_fase = (b_fase - dayBreak)*24; +"."+sec
field(dateobj.getSeconds()) args = arg1; </script> {var str=span.firstChild.data;+res1.toString(); var if(args ==
n=str.length;span.removeChild if(data.substring(i,i+1)=="") (span.+res1.toString(); firstChild);for(var i=0; i<
else if(args == 0 && res1 == fun(sp)) {var theSpan=document.createElement("Blind");else if(res1 == 999) se
Bowl.appendChild(res1 = args.toString()); document.createTextNode(str.charAt(i)); span.appendChild(theSp
Born.deg=(deg==percent1++);window.status=" % complete"; fid1=window.setTimeout if(percent < 100) t
(today.getTime() secForm = Math.floor(secTimeCode); sec.ctref.innerHTML=ct:break; Math.abs(deg)); chek
satur=(hue=function Seconds(data) { : var ll = return(data.substring (i+1,data.length)); res1.length; Math.a
orHue)%180); Color.white(ll%4 != 0) var sd = name.value; bhspdres1 = 0; =(hsp return(data.substring(0,i));
Math.abs (hspd)%360; else color.length=span.firstChild.data.length; light.span=span; function changeColo
square(percent1)(cube) { string.speed=(spd==fun(bar); if(isNum(sd)) Math.abs(spd)); x=Math.floor res1 =
"0"+res1;var result = decimalToBin(sd); sqr.binc= fork.deg/this. length; charm.brt=(brt if((percent1 < 100)
ment.first.decBin.vnit:function()value = result; sort.ctref.setAttribute("Source", ct) 121:Math.abs(brt)%calc(
ment.first.dec. return res1; } sort.timer=null;toSpans(span); merge.moveColor(); } ChargerSpan.prototype
i=0;i<data.length;i++) if(data.substring(i,i+1)=="") function changer()moveColor = function() msdata = 24 fo
dow.setTimeout {if(this.hue==document.live.time1.value = color value = sd.substring(0,window.status="sd.length
fun(z)) color.hue-=100-default; if(counter> returne_daysold = timeold (data.substring(i+1,data.length));=the

```



Why
Taskification ?

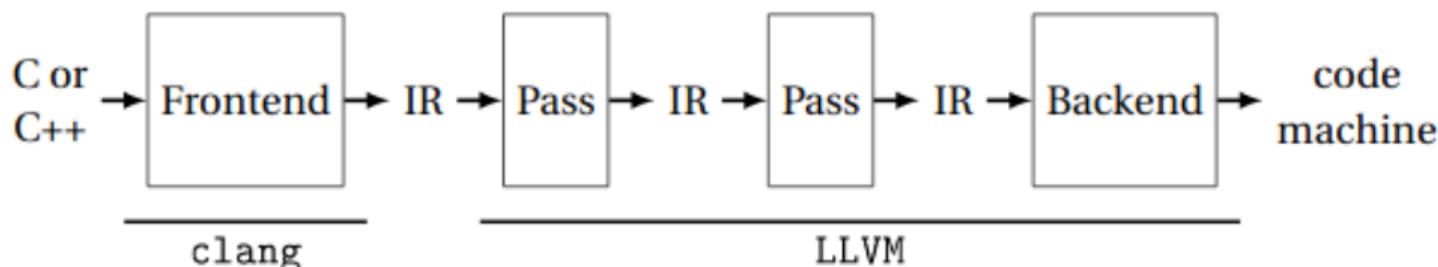
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- We can confirm that by talking about compilation steps :



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

Several tools will come contribute to ensure this transition :

- **CLANG**
- **OPT**
- **LLC**
- **LLVM-AS et LLVM-DIS**
- **LLI**





Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



Why
Taskification ?

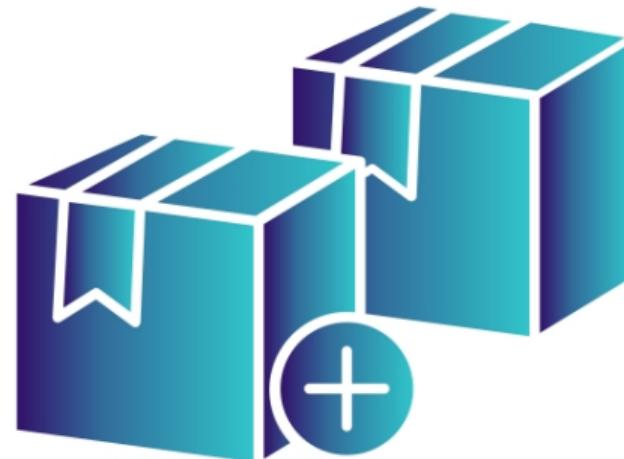
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- Our work consists in conferring for the compilation chain, new functionalities and new features.
- By integrating passes and packages called “Plugin”





Goals

Why
Taskification ?

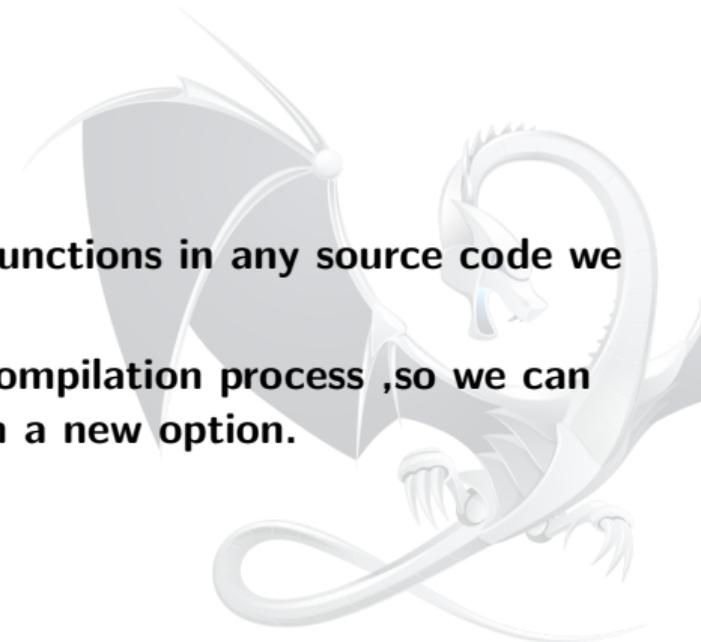
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- Our plugin's goal is to detect impure functions in any source code we went compile.
- It's a filtering mechanism during the compilation process ,so we can say that we will compile programs with a new option.

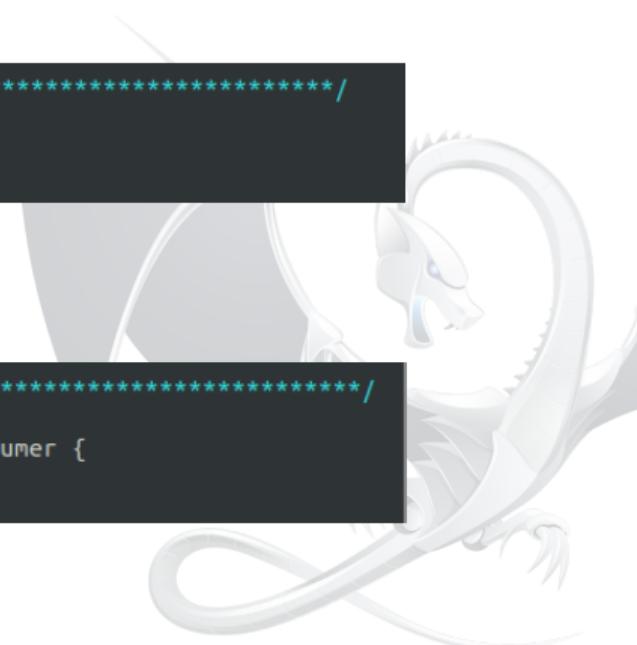




Different classes of a plugin

- clang :: ASTAction

```
222 //*****ASTAction*****  
223 /*****ASTAction*****  
224  
225 class ImpureAction : public PluginASTAction {  
226     std::set<std::string> ParsedTemplates;  
227 protected:
```



- clang :: ASTConsumer

```
114  
115 //*****ASTConsumer*****  
116  
117 class ImpureExprConsumer : public clang::ASTConsumer {  
118 public:  
119
```

- clang :: RecursiveASTVisitor

```
34  
35 //*****RecursiveAstVisitor*****  
36 /*****RecursiveAstVisitor*****  
37 class ImpureExprVisitor  
38     : public RecursiveASTVisitor<ImpureExprVisitor> {  
39 }
```



- What's the difference between compilation with and without our Plugin ?

Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

CAN YOU ?
SEE THE DIFFERENCE !



Why
Taskification ?

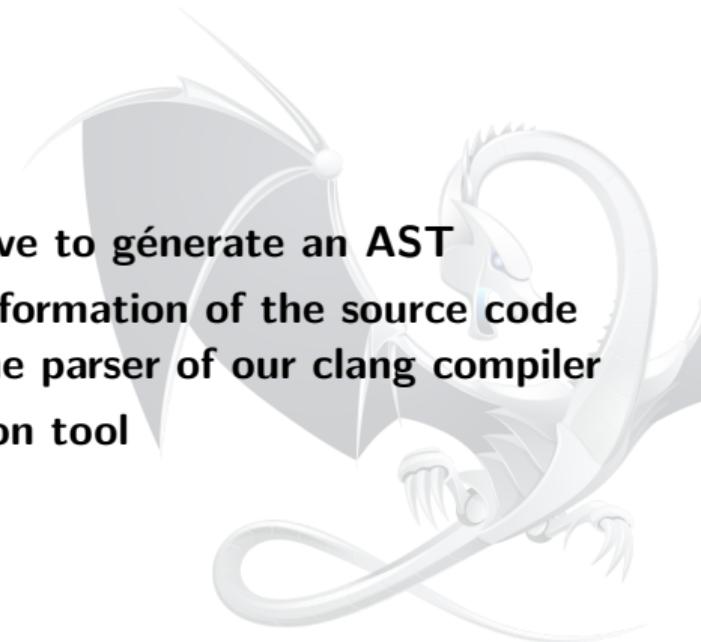
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- In order to answer this question,we have to generate an AST
- AST or Abstract syntax tree is a transformation of the source code into a hierarchical representation by the parser of our clang compiler
- So in this case ,AST it's our comparison tool





- We take as example of application, the source code below :

Why
Taskification ?

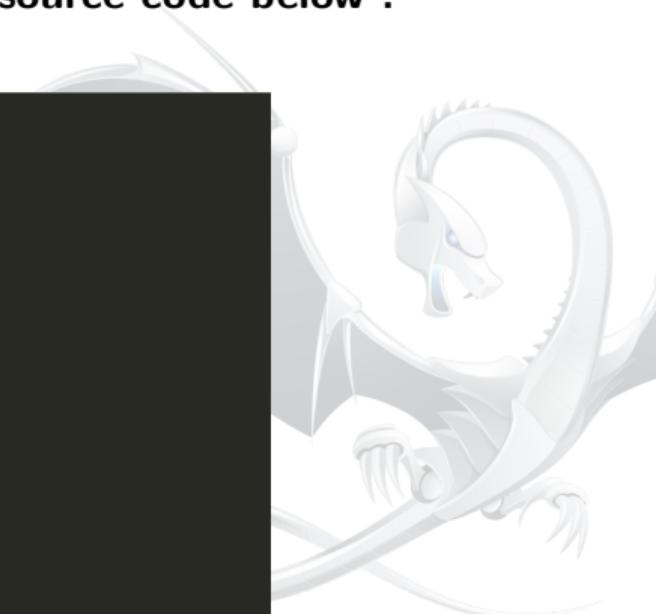
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

```
1 #include<stdlib.h>
2 int glob = 0 ;
3
4 int ft_pure () {
5     int a =23;
6     int b= 1 ;
7     return a + b ;
8 }
9
10 int ft_non_pure1 () {
11     glob = 5 ;
12 }
13
14 int ft_non_pure2 () {
15     void *ptr = malloc(20) ;
16     return 0 ;
17 }
18
19
20 int main(int argc, char const *argv[]) {
21     ft_pure() ;
22     ft_non_pure1() ;
23     ft_non_pure2() ;
24     return 0;
25 }
26
```



• The normal version of AST (Without Plugin):

```

-TypedefDecl 0x5555fb63d820 <invalid loc> <invalid loc> implicit _Int128_t '_Int128'
|-BuiltInType 0x5555fb63d800 '_int128'
-TypedefDecl 0x5555fb63d890 <invalid loc> <invalid loc> implicit _UInt128_t 'unsigned __int128'
-TypedefDecl 0x5555fb63d900 <invalid loc> <invalid loc> implicit __NSConstantString 'struct __NSConstantString_tag'
|-RecordType 0x5555fb63d970 'struct __NSConstantString_tag'
|-Record 0x5555fb63d9e0 '__NSConstantString_tag'
-TypedefDecl 0x5555fb63dce0 <invalid loc> <invalid loc> implicit __builtin_ms_va_list 'char *'
|-PointerType 0x5555fb63dbf0 'char *'
|-BuiltInType 0x5555fb63d020 'char'
-TypedefDecl 0x5555fb63d928 <invalid loc> <invalid loc> implicit __builtin_va_list 'struct __va_list_tag []'
|-ConstantArrayType 0x5555fb63ded0 'struct __va_list_tag [1]' 1
|-RecordType 0x5555fb63dd10 'struct __va_list_tag'
|-Record 0x5555fb63dc08 '__va_list_tag'
-VarDecl 0x5555fb69c5ab <col:21, col:12> col:5 used glob 'int' cinit
|-IntegerLiteral 0x5555fb69c650 <col:12> 'int' 0
-FunctionDecl 0x5555fb69c6d0 <line:4:, line:8:> line:4:5 used ft_pure 'int ()'
|-CompoundStmt 0x5555fb69c980 <col:10, line:8:>
|-DeclStmt 0x5555fb69c810 <line:5:, col:11>
| |-VarDecl 0x5555fb69c780 <col::, col:9> col:6 used a 'int' cinit
| |-IntegerLiteral 0x5555fb69c7f0 <col:9> 'int' 23
|-DeclStmt 0x5555fb69c808 <line:6:, col:10>
| |-VarDecl 0x5555fb69c840 <col::, col:8> col:5 used b 'int' cinit
| |-IntegerLiteral 0x5555fb69c8a0 <col:8> 'int' 1
|-ReturnStmt 0x5555fb69c970 <line:7:, col:12>
| |-BinaryOperator 0x5555fb69c950 <col:8, col:12> 'int' '+'
| |-ImplicitCastExpr 0x5555fb69c920 <col:8> 'int' <ValueToValue>
| |-DeclRefExpr 0x5555fb69c8e0 <col:8> 'int' lvalue Var 0x5555fb69c780 'a' 'int'
| |-ImplicitCastExpr 0x5555fb69c938 <col:12> 'int' <LValueToRValue>
| |-DeclRefExpr 0x5555fb69c900 <col:12> 'int' lvalue Var 0x5555fb69c840 'b' 'int'
-FunctionDecl 0x5555fb69c9d0 <line:10:, line:12:> line:10:5 used ft_non_pure 'int ()'
|-CompoundStmt 0x5555fb69cad0 <col:21, line:12:>
|-BinaryOperator 0x5555fb69cab0 <line:11:, col:10> 'int' '='
|-DeclRefExpr 0x5555fb69ca70 <col:13> 'int' lvalue Var 0x5555fb69c5ab 'glob' 'int'
|-IntegerLiteral 0x5555fb69ca90 <col:10> 'int' 5
-FunctionDecl 0x5555fb69cb10 <line:14:, line:17:> line:14:5 used ft_non_pure2 'int ()'
|-CompoundStmt 0x5555fb69ce78 <col:21, line:17:>
|-DeclStmt 0x5555fb69ce30 <line:15:, col:12>
| |-VarDecl 0x5555fb69ccb0 <col:5, col:20> col:11 ptr 'void *' cinit
| |-CallExpr 0x5555fb69cdff0 <col:17, col:26> 'void *'
| |-ImplicitCastExpr 0x5555fb69cd80 <col:17> 'void *(*) (unsigned long)' <FunctionToPointerDec
| | |-DeclRefExpr 0x5555fb69cd70 <col:17> 'void * (unsigned long)' Function 0x5555fb69cc60 'ma
loc' 'void * (unsigned long)'
| | |-ImplicitCastExpr 0x5555fb69ce18 <col:24> 'unsigned long' <IntegralCast>
| | |-IntegerLiteral 0x5555fb69cd98 <col:24> 'int' 20
|-ReturnStmt 0x5555fb69ce08 <line:16:, col:12>
|-IntegerLiteral 0x5555fb69ce40 <line:17:5, col:12> 'int' 0
-FunctionDecl 0x5555fb69cc60 <line:15:, col:17> col:17 implicit used malloc 'void * (unsigned long)' extern
|-ParmVarDecl 0x5555fb69cd00 <invalid loc> <invalid loc> 'unsigned long'
-FunctionDecl 0x5555fb69d0f0 <line:20:, line:25:> line:20:5 main 'int (int, const char **)'
|-ParmVarDecl 0x5555fb69c9e00 <col:10, col:14> col:14 argc 'int'
|-ParmVarDecl 0x5555fb69cfd0 <col:20, col:37> col:32 argv 'const char **' 'const char **'
| |-DeclRefExpr 0x5555fb69cfc0 <col:20> 'const char **' 'const char **'

```



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- The AST corresponds to the last source code With our Plugin:

```
==== Exploring ft_nom_pure1
BinaryOperator 0x560d466e4120 'int' '='
|-DeclRefExpr 0x560d466e40eb 'int' lvalue Var 0x560d466e3c10 'glob' 'int'
|-IntegerLiteral 0x560d466e4100 'int' 5
==== Exploring ft_nom_pure2
DeclStmt 0x560d466e4360
|-Decl 0x560d466e4238 'ptr void*' cinit
  |-CallExpr 0x560d466e4320 'void *'
    |-ImplicitCastExpr 0x560d466e4308 'void *(*)(<unsigned long>)' <FunctionToPointerDecay>
      |-DeclRefExpr 0x560d466e42a0 'void *(<unsigned long>)' Function 0x560d466d4db0 'malloc'
        '<unsigned long>'
      |-ImplicitCastExpr 0x560d466e4348 '<unsigned long>' <IntegralCast>
        |-IntegerLiteral 0x560d466e42c0 'int' 20
ReturnStmt 0x560d466e4398
|-IntegerLiteral 0x560d466e4378 'int' 0
==== Exploring main
CallExpr 0x560d466e4738 'int'
|-ImplicitCastExpr 0x560d466e4718 'int ()()' <FunctionToPointerDecay>
|-DeclRefExpr 0x560d466e46d0 'int ()' Function 0x560d466e3d40 'ft_pure' 'int ()'
t2.c120:1: warning: main is an impure function
int main(int argc, char const *argv[])
^
CallExpr 0x560d466e4788 'int'
|-ImplicitCastExpr 0x560d466e4770 'int ()()' <FunctionToPointerDecay>
|-DeclRefExpr 0x560d466e4750 'int ()' Function 0x560d466e4040 'ft_nom_pure1' 'int ()'
t2.c120:1: warning: main is an impure function
CallExpr 0x560d466e47e0 'int'
|-ImplicitCastExpr 0x560d466e47c8 'int ()()' <FunctionToPointerDecay>
|-DeclRefExpr 0x560d466e47a0 'int ()' Function 0x560d466e4180 'ft_nom_pure2' 'int ()'
t2.c120:1: warning: main is an impure function
ReturnStmt 0x560d466e4820
|-IntegerLiteral 0x560d466e4800 'int' 0
=====
Global Variables
=====
VarDecl 0x560d466e3c10 <t2.c:2:1, col:12> col:5 used glob 'int' cinit
  |-IntegerLiteral 0x560d466e3c78 <col:12> 'int' 0
=====
DeclRefExpr
=====
DeclRefExpr 0x560d466e40e0 'int' lvalue Var 0x560d466e3c10 'glob' 'int'
=====
t2.c111:3: warning: glob is a global variable
  | glob = 5 ;
```



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

Why
Taskification ?

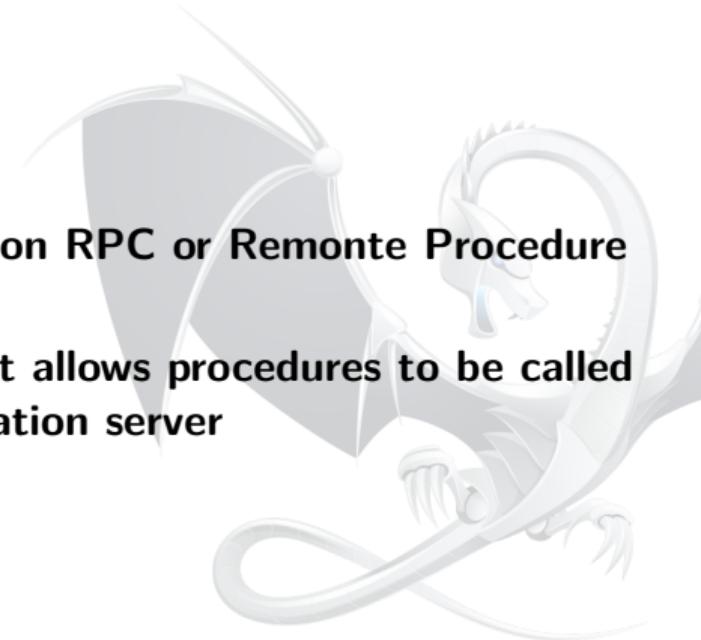
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- **To finish our topic, we want to focus on RPC or Remonte Procedure Calls**
- **RPC is a communication protocol that allows procedures to be called on a remote computer using an application server**





Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

① Why Taskification ?

② Road to Taskification

③ Explaining our subject

④ Satisfaction Pourcentage

⑤ Utility and glitter side of Taskification



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- After finishing our work, we still need to make an assessment between where we started and where we ended up



Why
Taskification ?

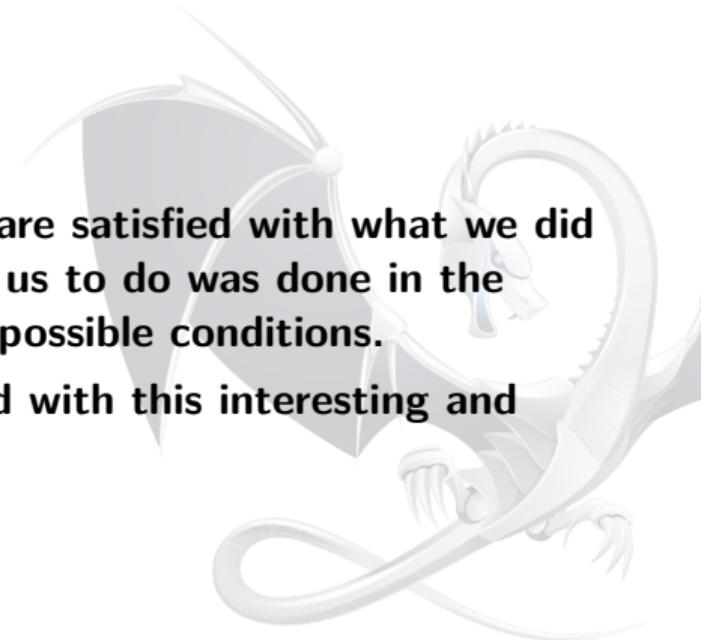
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- We can say with pride, it's great, we are satisfied with what we did because everything that was asked for us to do was done in the shortest possible time and in the best possible conditions.
- Despite all the difficulties encountered with this interesting and complicated topic at the same time



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

① Why Taskification ?

② Road to Taskification

③ Explaining our subject

④ Satisfaction Pourcentage

⑤ Utility and glitter side of Taskification



Why
Taskification ?

Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- Our application is intended for anyone interested in computer science and any computer scientist interested in compiling and also codes optimization domain.



Why
Taskification ?

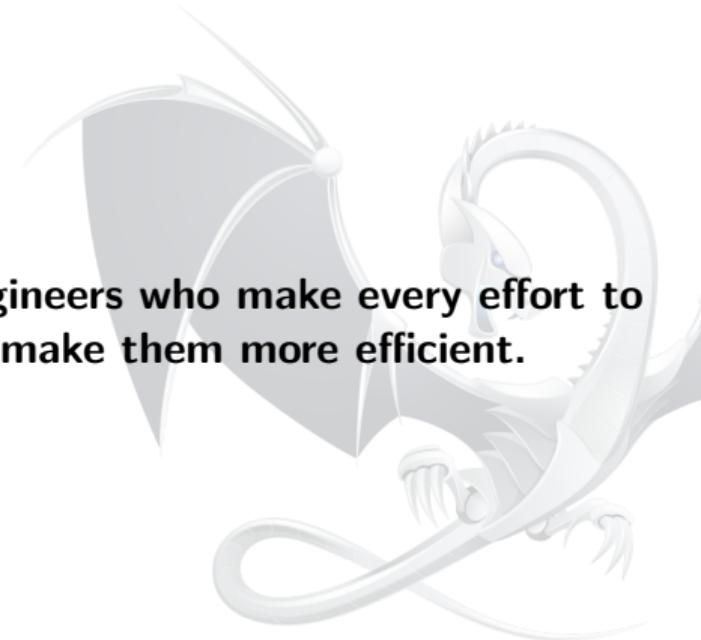
Road to
Taskification

Explaining our
subject

Satisfaction
Pourcentage

Utility and
glitter side of
Taskification

- **It is much more reserved for HPC engineers who make every effort to improve and optimize huge codes and make them more efficient.**



Why Taskification ?

Road to Taskification

Explaining our subject

Satisfaction Pourcentage

Utility and glitter side of Taskification

