Debugging Regressions: Interactive Differential Debugging

VIPUL CARIAPPA, MARTIN VASSILEV,
ALEXANDER PENEV, VASSIL VASSILEV

This project is partially supported by the National Science Foundation under award OISE-2201990

Problem

- Modern software systems are complex, with millions of lines of code, making debugging difficult.
- Differential debugging simplifies the process by comparing the current system to a previous version as a baseline.
- Current debugging practice involves running two separate debugger instances without communication about their execution states.

Boss thanking me for fixing a critical bug
Me, having just rolled back my latest commit

Solution: Interactive Differential Debugging (idd)

IDD automates the process of filtering out irrelevant execution paths between a reference and the regressed software system.

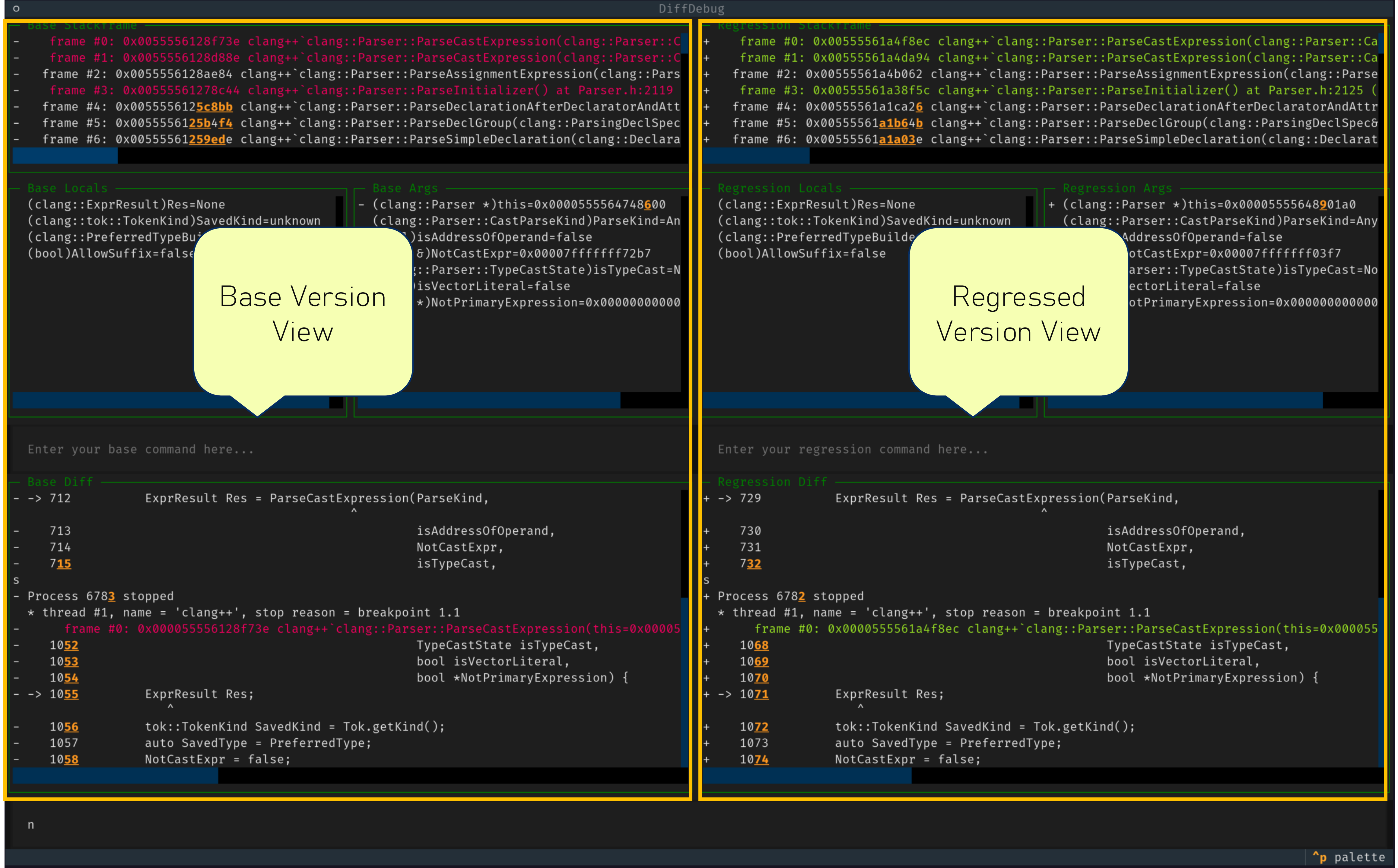
How does it work?

- Load two versions of the system
  - Base: The version of the system we expect to be fine.
  - Regressed: The version that has a regression introduced in it.
- Use LLDB/GDB to inspect both versions of the system simultaneously.
- Leverage diff-view to look at the differences between the states of both systems.
- Deduce the cause of the regression faster by ignoring common/irrelevant execution paths.

The result is a focused display of debugger states that differ between the two versions.

Architecture

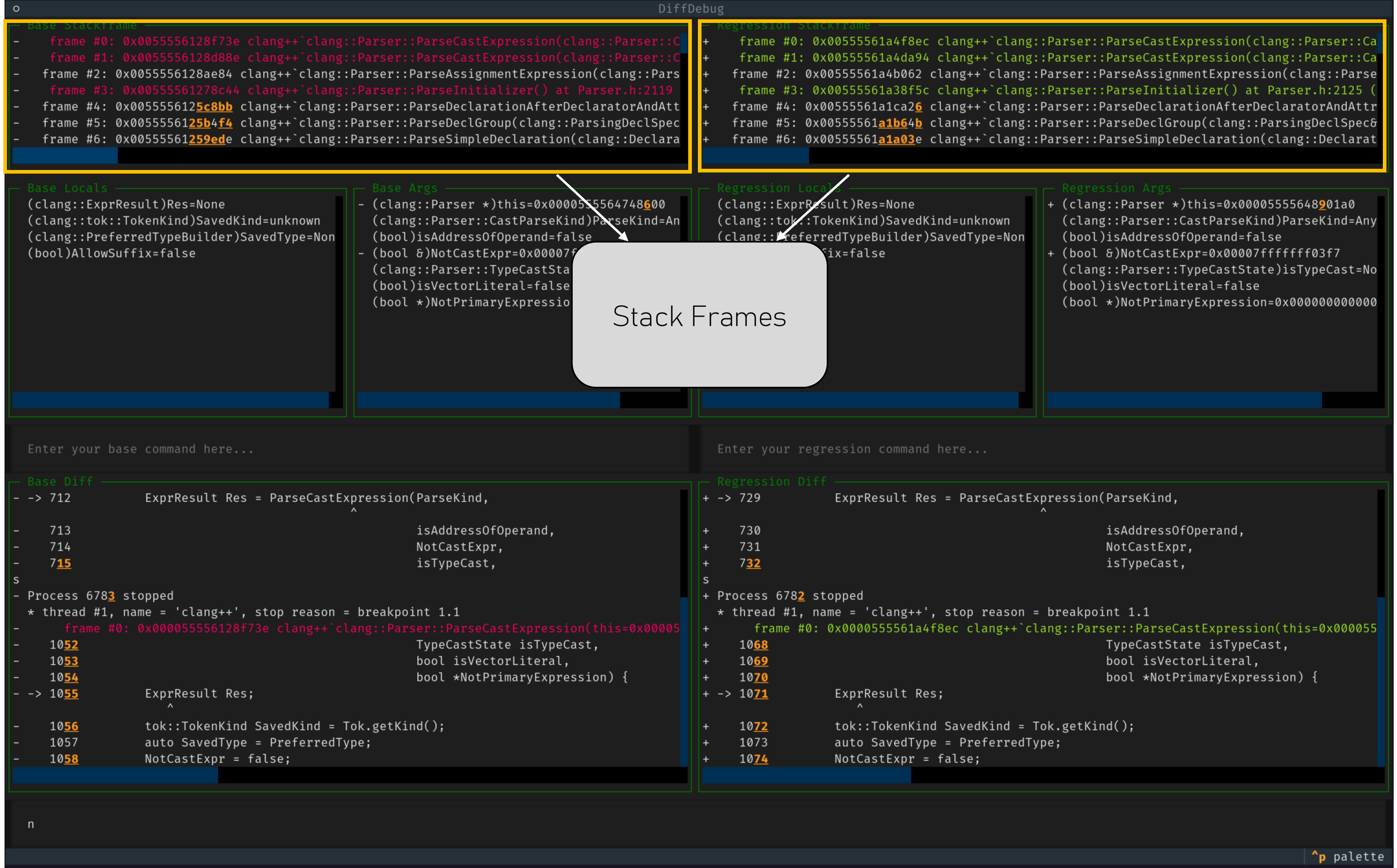
- Common place for data exchange between two debuggers. Works with either lldb or gdb.
  - Using lldb python API to steer the lldb debugger.
  - Similar but less powerful approach with gdb due to the limited API supporting tools.
- Git style difference viewer to easily recognize differences between two versions.
- Organizable UI with CSS to concentrate on what matter to you.



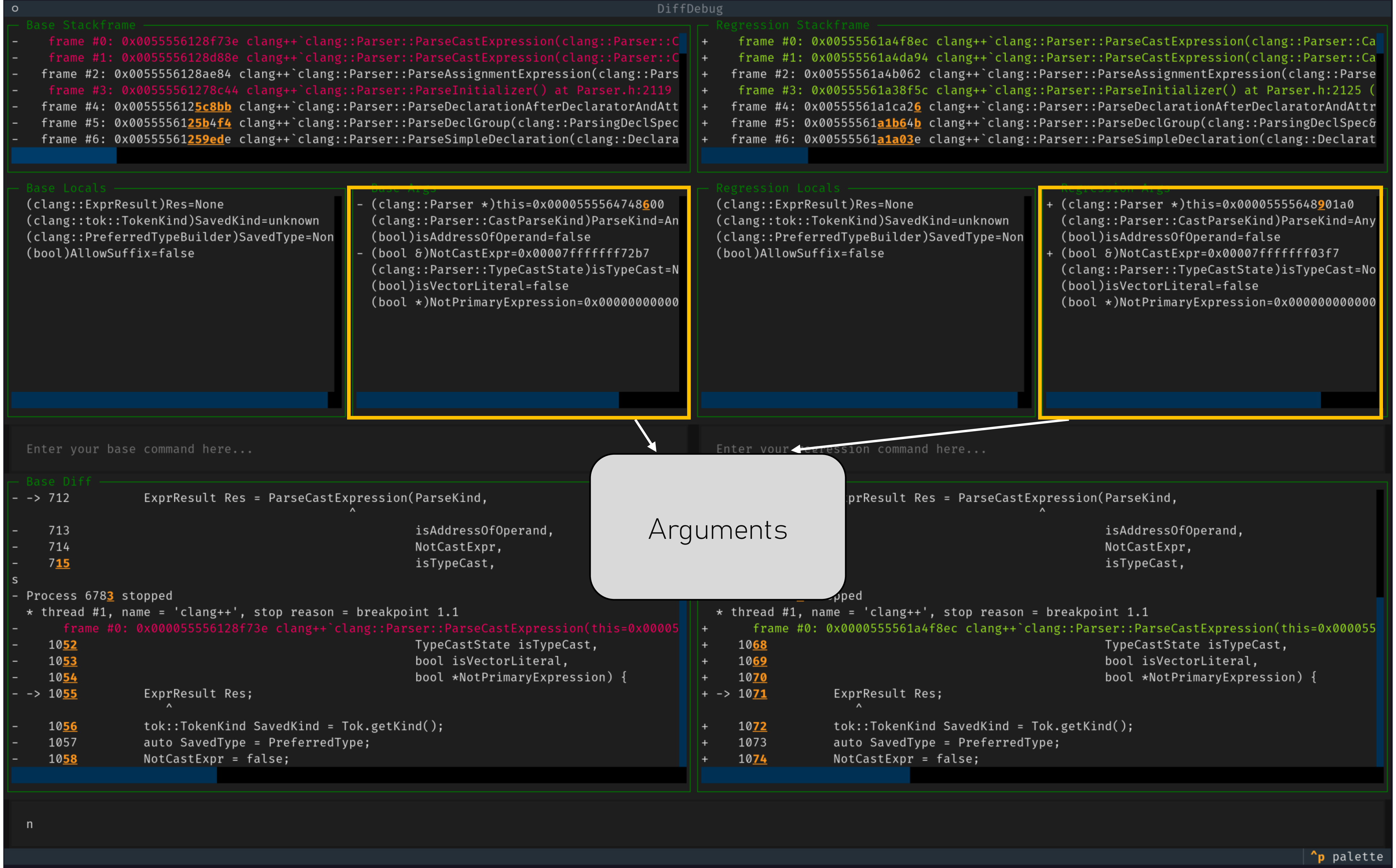
```
Base Stackframe
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x005555612 5c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x00555561 25b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x00555561 259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara

Base Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBu
  (bool)AllowSuffix=false

Base Args
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  )isAddressOfOperand=false
  &)NotCastExpr=0x00007fffffff72b7
  ::Parser::TypeCastState)isTypeCast=N
  )isVectorLiteral=false
  *)NotPrimaryExpression=0x00000000000

  Enter your base command here...

Base Diff
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                                  ^

-    713                                     isAddressOfOperand,
-    714                                     NotCastExpr,
-    715                                     isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                    TypeCastState isTypeCast,
-    1053                                    bool isVectorLiteral,
-    1054                                    bool *NotPrimaryExpression) {
- -> 1055          ExprResult Res;
                   ^

-    1056          tok::TokenKind SavedKind = Tok.getKind();
-    1057          auto SavedType = PreferredType;
-    1058          NotCastExpr = false;
```

```
Regression Stackframe
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat

Regression Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilde
  (bool)AllowSuffix=false

Regression Args
+ (clang::Parser *)this=0x00005555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  AddressOfOperand=false
  otCastExpr=0x00007fffffff03f7
  arser::TypeCastState)isTypeCast=No
  ectorLiteral=false
  otPrimaryExpression=0x000000000000

  Enter your regression command here...

Regression Diff
+ -> 729          ExprResult Res = ParseCastExpression(ParseKind,
                                                  ^

+    730                                     isAddressOfOperand,
+    731                                     NotCastExpr,
+    732                                     isTypeCast,
s
+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                    TypeCastState isTypeCast,
+    1069                                    bool isVectorLiteral,
+    1070                                    bool *NotPrimaryExpression) {
+ -> 1071          ExprResult Res;
                   ^

+    1072          tok::TokenKind SavedKind = Tok.getKind();
+    1073          auto SavedType = PreferredType;
+    1074          NotCastExpr = false;
```

```
  n
```



```
DiffDebug

Base StackFrame
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x005555612 5c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x0055556125b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara

Regression StackFrame
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat

Base Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false

Base Args
- (clang::Parser *)this=0x00005555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007f
  (clang::Parser::TypeCastSta
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpressio

Regression Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (                     fix=false

Regression Args
+ (clang::Parser *)this=0x0000555555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000

Enter your base command here...          Enter your regression command here...

Base Diff
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                          ^

-    713                                     isAddressOfOperand,
-    714                                     NotCastExpr,
-    715                                     isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                    TypeCastState isTypeCast,
-    1053                                    bool isVectorLiteral,
-    1054                                    bool *NotPrimaryExpression) {
- -> 1055         ExprResult Res;
                      ^

-    1056         tok::TokenKind SavedKind = Tok.getKind();
-    1057         auto SavedType = PreferredType;
-    1058         NotCastExpr = false;

Regression Diff
+ -> 729          ExprResult Res = ParseCastExpression(ParseKind,
                                          ^

+    730                                     isAddressOfOperand,
+    731                                     NotCastExpr,
+    732                                     isTypeCast,
s
+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                    TypeCastState isTypeCast,
+    1069                                    bool isVectorLiteral,
+    1070                                    bool *NotPrimaryExpression) {
+ -> 1071         ExprResult Res;
                      ^

+    1072         tok::TokenKind SavedKind = Tok.getKind();
+    1073         auto SavedType = PreferredType;
+    1074         NotCastExpr = false;

n
                                                                  ^p palette
```

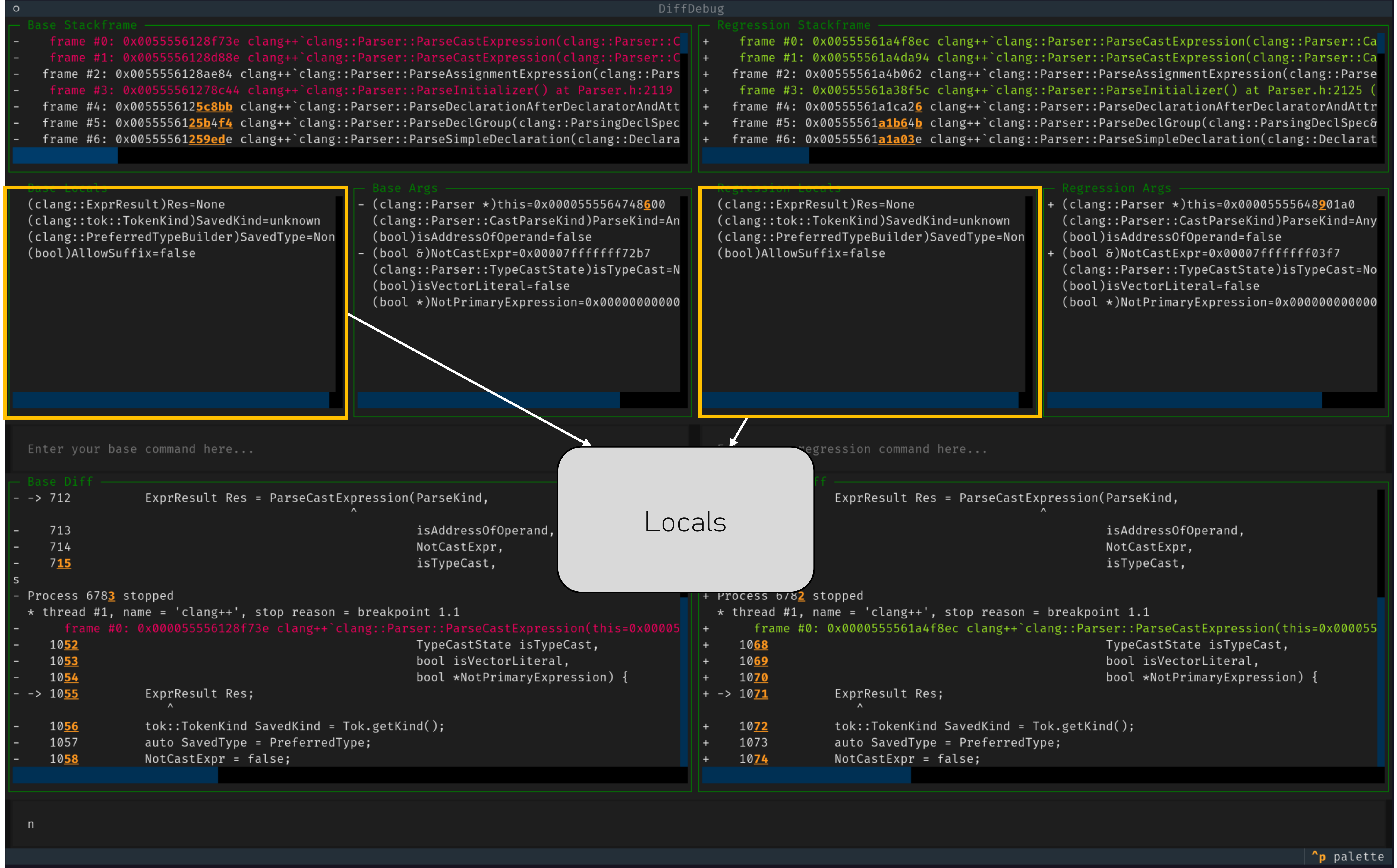


```
DiffDebug

Base Stackframe
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x0055556125c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x0055556125b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara

Regression Stackframe
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat

Base Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false

Base Args
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007fffffff72b7
  (clang::Parser::TypeCastState)isTypeCast=N
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x00000000000

Regression Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false

Regression Args
+ (clang::Parser *)this=0x00005555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000

Enter your base command here...
Enter your regression command here...
```

Arguments

```
Base Diff
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                             ^
-    713                                    isAddressOfOperand,
-    714                                    NotCastExpr,
-    715                                    isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                   TypeCastState isTypeCast,
-    1053                                   bool isVectorLiteral,
-    1054                                   bool *NotPrimaryExpression) {
- -> 1055         ExprResult Res;
                   ^
-    1056         tok::TokenKind SavedKind = Tok.getKind();
-    1057         auto SavedType = PreferredType;
-    1058         NotCastExpr = false;
```

```
prResult Res = ParseCastExpression(ParseKind,
                          ^
                                    isAddressOfOperand,
                                    NotCastExpr,
                                    isTypeCast,

pped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                   TypeCastState isTypeCast,
+    1069                                   bool isVectorLiteral,
+    1070                                   bool *NotPrimaryExpression) {
+ -> 1071         ExprResult Res;
                   ^
+    1072         tok::TokenKind SavedKind = Tok.getKind();
+    1073         auto SavedType = PreferredType;
+    1074         NotCastExpr = false;

n
                                                                              ^p palette
```



```
DiffDebug
```

Base Stackframe

```
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x005555612 5c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x005555612 5b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x005555612 59ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara
```

Regression Stackframe

```
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat
```

Base Locals

```
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false
```

Base Args

```
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007fffffff72b7
  (clang::Parser::TypeCastState)isTypeCast=N
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x00000000000
```

Regression Locals

```
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false
```

Regression Args

```
+ (clang::Parser *)this=0x00005555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000
```

```
Enter your base command here...
```

```
regression command here...
```

Locals

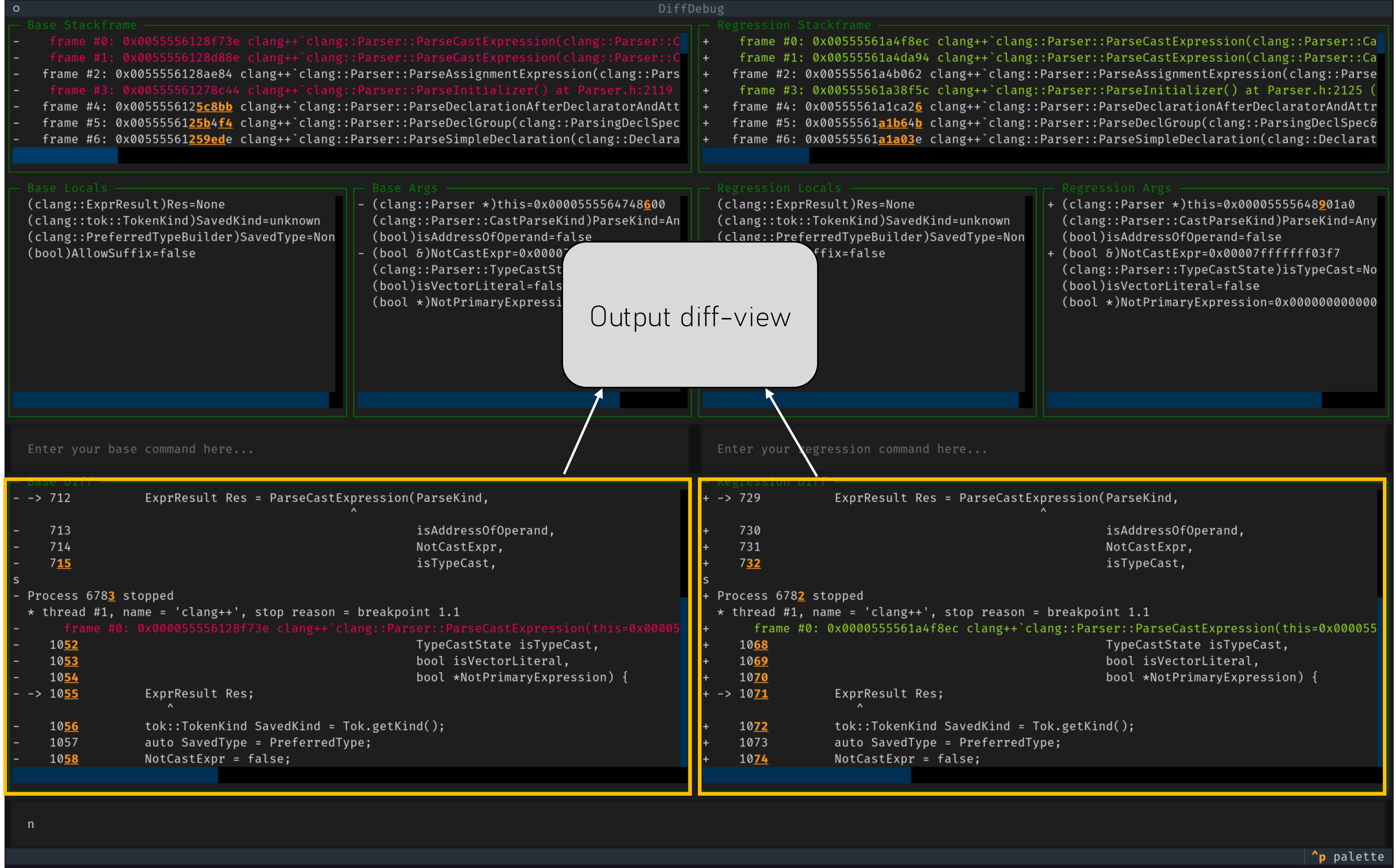
Base Diff

```
- -> 712         ExprResult Res = ParseCastExpression(ParseKind,
                                                     ^

-    713                                   isAddressOfOperand,
-    714                                   NotCastExpr,
-    715                                   isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                  TypeCastState isTypeCast,
-    1053                                  bool isVectorLiteral,
-    1054                                  bool *NotPrimaryExpression) {
- -> 1055        ExprResult Res;
                            ^

-    1056        tok::TokenKind SavedKind = Tok.getKind();
-    1057        auto SavedType = PreferredType;
-    1058        NotCastExpr = false;
```

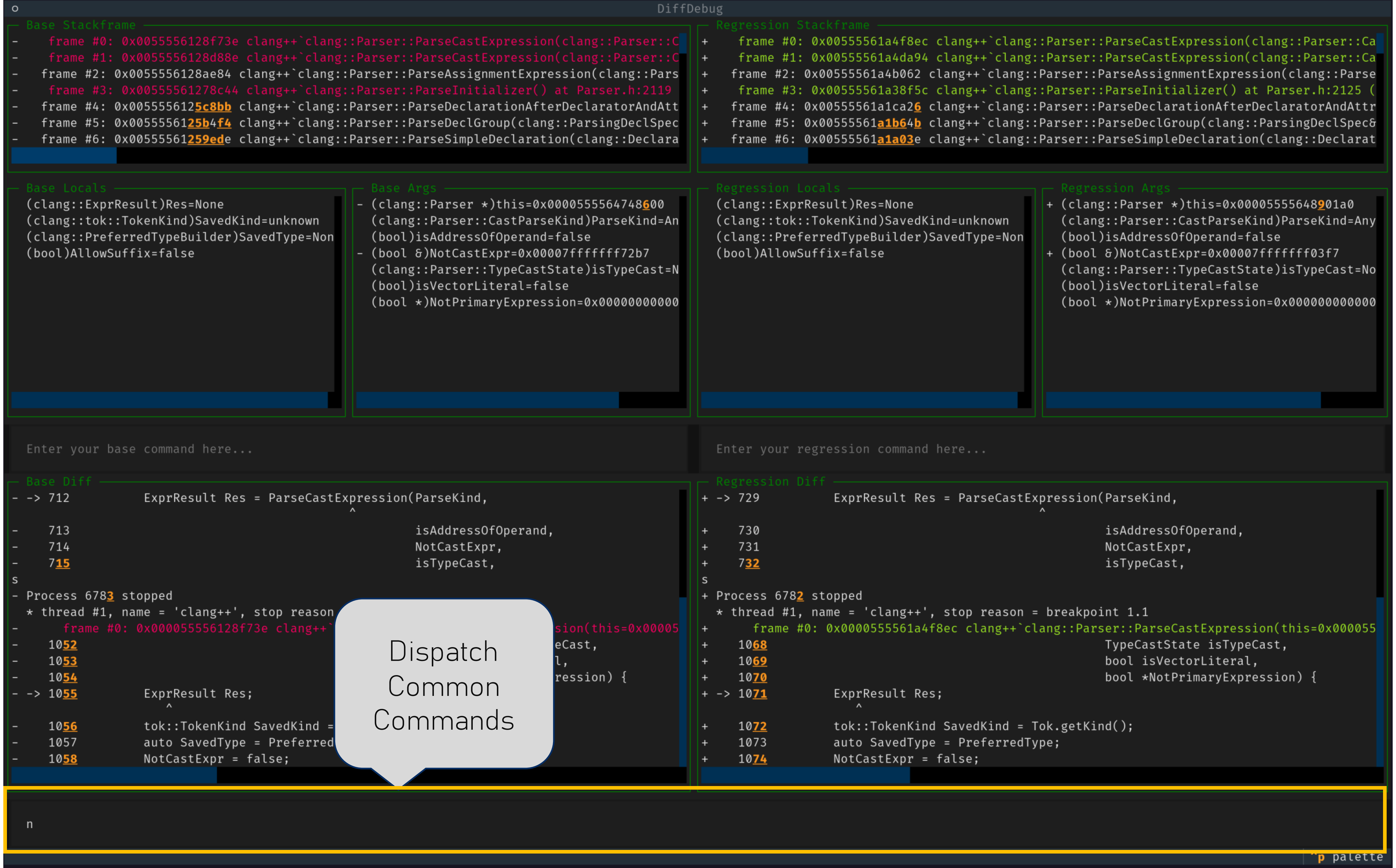
```
ff
             ExprResult Res = ParseCastExpression(ParseKind,
                                                     ^

                                   isAddressOfOperand,
                                   NotCastExpr,
                                   isTypeCast,

+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                  TypeCastState isTypeCast,
+    1069                                  bool isVectorLiteral,
+    1070                                  bool *NotPrimaryExpression) {
+ -> 1071        ExprResult Res;
                            ^

+    1072        tok::TokenKind SavedKind = Tok.getKind();
+    1073        auto SavedType = PreferredType;
+    1074        NotCastExpr = false;
```

```
n
```

```
^p palette
```



```
DiffDebug

Base Stackframe
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x005555612 5c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x0055556125b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara

Regression Stackframe
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat

Base Locals
(clang::ExprResult)Res=None
(clang::tok::TokenKind)SavedKind=unknown
(clang::PreferredTypeBuilder)SavedType=Non
(bool)AllowSuffix=false

Base Args
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007
  (clang::Parser::TypeCastSt
  (bool)isVectorLiteral=fals
  (bool *)NotPrimaryExpressi

Regression Locals
(clang::ExprResult)Res=None
(clang::tok::TokenKind)SavedKind=unknown
(clang::PreferredTypeBuilder)SavedType=Non
                  ffix=false

Regression Args
+ (clang::Parser *)this=0x00005555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000

Enter your base command here...            Enter your regression command here...

Base Diff
- -> 712         ExprResult Res = ParseCastExpression(ParseKind,
                                          ^

-    713                                    isAddressOfOperand,
-    714                                    NotCastExpr,
-    715                                    isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-     frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpression(this=0x00005
-    1052                                   TypeCastState isTypeCast,
-    1053                                   bool isVectorLiteral,
-    1054                                   bool *NotPrimaryExpression) {
- -> 1055        ExprResult Res;
                            ^

-    1056        tok::TokenKind SavedKind = Tok.getKind();
-    1057        auto SavedType = PreferredType;
-    1058        NotCastExpr = false;

Regression Diff
+ -> 729         ExprResult Res = ParseCastExpression(ParseKind,
                                          ^

+    730                                    isAddressOfOperand,
+    731                                    NotCastExpr,
+    732                                    isTypeCast,
s
+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+     frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                   TypeCastState isTypeCast,
+    1069                                   bool isVectorLiteral,
+    1070                                   bool *NotPrimaryExpression) {
+ -> 1071        ExprResult Res;
                            ^

+    1072        tok::TokenKind SavedKind = Tok.getKind();
+    1073        auto SavedType = PreferredType;
+    1074        NotCastExpr = false;

n
                                                                       ^p palette
```



```
DiffDebug
```

```
Base Stackframe
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x0055556125c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x00555556125b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x005555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara
```

```
Regression Stackframe
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat
```

```
Base Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false
```

```
Base Args
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007fffffff72b7
  (clang::Parser::TypeCastState)isTypeCast=N
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x00000000000
```

```
Regression Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false
```

```
Regression Args
+ (clang::Parser *)this=0x00005555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000
```

```
Enter your base command here...
```

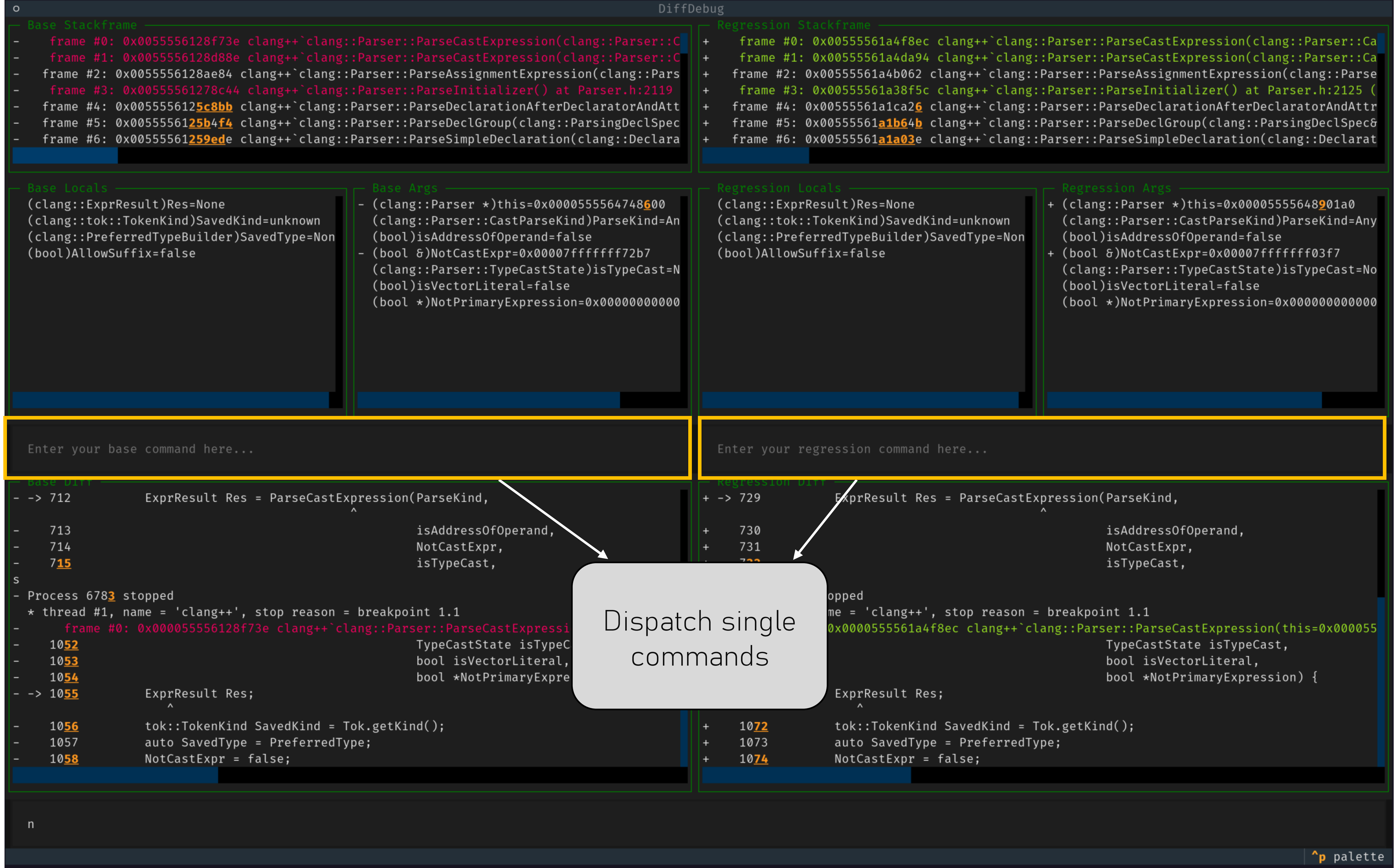
```
Enter your regression command here...
```

```
Base Diff
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                                ^

-    713                                     isAddressOfOperand,
-    714                                     NotCastExpr,
-    715                                     isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason
-      frame #0: 0x000055556128f73e clang++`                              sion(this=0x00005
-    1052                                                                  eCast,
-    1053                                                                  l,
-    1054                                                                  ression) {
- -> 1055          ExprResult Res;
                    ^

-    1056          tok::TokenKind SavedKind =
-    1057          auto SavedType = Preferred
-    1058          NotCastExpr = false;
```

```
Regression Diff
+ -> 729          ExprResult Res = ParseCastExpression(ParseKind,
                                                ^

+    730                                     isAddressOfOperand,
+    731                                     NotCastExpr,
+    732                                     isTypeCast,
s
+ Process 6782 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
+      frame #0: 0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
+    1068                                     TypeCastState isTypeCast,
+    1069                                     bool isVectorLiteral,
+    1070                                     bool *NotPrimaryExpression) {
+ -> 1071          ExprResult Res;
                    ^

+    1072          tok::TokenKind SavedKind = Tok.getKind();
+    1073          auto SavedType = PreferredType;
+    1074          NotCastExpr = false;
```

```
n
```

```
^p palette
```

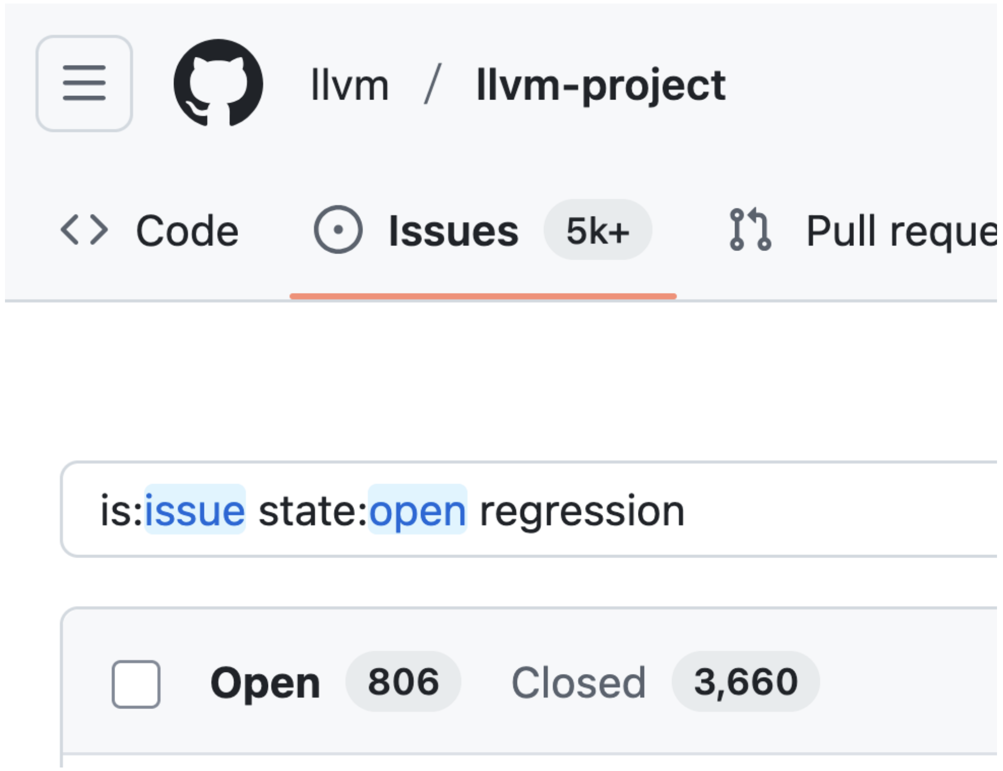


```
DiffDebug
Base Stackframe
-    frame #0: 0x0055556128f73e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-    frame #1: 0x0055556128d88e clang++`clang::Parser::ParseCastExpression(clang::Parser::C
-   frame #2: 0x0055556128ae84 clang++`clang::Parser::ParseAssignmentExpression(clang::Pars
-    frame #3: 0x00555561278c44 clang++`clang::Parser::ParseInitializer() at Parser.h:2119
-   frame #4: 0x005555612 5c8bb clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAtt
-   frame #5: 0x005555612 5b4f4 clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec
-   frame #6: 0x00555561259ede clang++`clang::Parser::ParseSimpleDeclaration(clang::Declara

Regression Stackframe
+    frame #0: 0x00555561a4f8ec clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+    frame #1: 0x00555561a4da94 clang++`clang::Parser::ParseCastExpression(clang::Parser::Ca
+   frame #2: 0x00555561a4b062 clang++`clang::Parser::ParseAssignmentExpression(clang::Parse
+    frame #3: 0x00555561a38f5c clang++`clang::Parser::ParseInitializer() at Parser.h:2125 (
+   frame #4: 0x00555561a1ca26 clang++`clang::Parser::ParseDeclarationAfterDeclaratorAndAttr
+   frame #5: 0x00555561a1b64b clang++`clang::Parser::ParseDeclGroup(clang::ParsingDeclSpec&
+   frame #6: 0x00555561a1a03e clang++`clang::Parser::ParseSimpleDeclaration(clang::Declarat

Base Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false

Base Args
- (clang::Parser *)this=0x0000555564748600
  (clang::Parser::CastParseKind)ParseKind=An
  (bool)isAddressOfOperand=false
- (bool &)NotCastExpr=0x00007fffffff72b7
  (clang::Parser::TypeCastState)isTypeCast=N
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000

Regression Locals
  (clang::ExprResult)Res=None
  (clang::tok::TokenKind)SavedKind=unknown
  (clang::PreferredTypeBuilder)SavedType=Non
  (bool)AllowSuffix=false

Regression Args
+ (clang::Parser *)this=0x000055555648901a0
  (clang::Parser::CastParseKind)ParseKind=Any
  (bool)isAddressOfOperand=false
+ (bool &)NotCastExpr=0x00007fffffff03f7
  (clang::Parser::TypeCastState)isTypeCast=No
  (bool)isVectorLiteral=false
  (bool *)NotPrimaryExpression=0x000000000000

Enter your base command here...

Enter your regression command here...

Base Diff
- -> 712          ExprResult Res = ParseCastExpression(ParseKind,
                                        ^

-    713                                  isAddressOfOperand,
-    714                                  NotCastExpr,
-    715                                  isTypeCast,
s
- Process 6783 stopped
  * thread #1, name = 'clang++', stop reason = breakpoint 1.1
-      frame #0: 0x000055556128f73e clang++`clang::Parser::ParseCastExpressi
-    1052                                 TypeCastState isTypeC
-    1053                                 bool isVectorLiteral,
-    1054                                 bool *NotPrimaryExpre
- -> 1055         ExprResult Res;
                             ^

-    1056         tok::TokenKind SavedKind = Tok.getKind();
-    1057         auto SavedType = PreferredType;
-    1058         NotCastExpr = false;

Regression Diff
+ -> 729          ExprResult Res = ParseCastExpression(ParseKind,
                                        ^

+    730                                  isAddressOfOperand,
+    731                                  NotCastExpr,
                                          isTypeCast,

opped
me = 'clang++', stop reason = breakpoint 1.1
0x0000555561a4f8ec clang++`clang::Parser::ParseCastExpression(this=0x000055
                                          TypeCastState isTypeCast,
                                          bool isVectorLiteral,
                                          bool *NotPrimaryExpression) {
           ExprResult Res;
                    ^

+    1072         tok::TokenKind SavedKind = Tok.getKind();
+    1073         auto SavedType = PreferredType;
+    1074         NotCastExpr = false;

n
                                                                    ^p palette
```

Advantages

Differential Debugging is not restricted to finding regressions in the codebase.

- Bug Localization in Regression Analysis
- Migration and Third-Party Library Updates
- Debugging Across Compiler Optimizations



llvm / llvm-project

Code Issues 5k+ Pull reque

is:issue state:open regression

Open 806 Closed 3,660

Time for Demonstration

Future Work

- Improved semantic diff. E.g. Address Space Randomization (ALSR)
- Automatically halt execution at diverging stack frames
- Watchpoints for diverging variables of interest

These enhancements would reduce manual effort, accelerate bug localization, and improve the overall user experience of IDD.

Thank You

Any Questions?

- 🔗 GitHub: github.com/compiler-research/idd
- 📦 PyPI: pypi.org/project/idd
- 📌 Install via: **pip install idd**