Fxtran

- Software
- Round tripping
- XML
- Editing (XML DOM)
- Searching (XML XPath)

Software

- Written in C
- GPL
- Embed cpp
- Fast; parse 3 millions lines in 2 minutes
- Most of Fortran 2003/2008
- No reserved keywords, fixed format

Round tripping

Fortran document

→ XML document

→ Fortran document

Final document = initial document

XML

```
PROGRAM MAIN

REAL :: X, Y, Z

X = Y + Z

CALL SUB (X, Y, Z)

END
```

XML

```
<?xml version="1.0"?><?xml-stylesheet type="text/css" href="fxtran.css"?
><object xmlns="http://fxtran.net/#syntax" source-form="FREE" source-
width="132" openmp="0"><file name="toto.F90"><program-unit><program-
stmt>PROGRAM <program-N><N><n>MAIN</n></N></program-N></program-stmt>

<T-decl-stmt><_T-spec_><intrinsic-T-spec><T-N>REAL</T-N></intrinsic-T-
spec></_T-spec_> :: <EN-decl-LT><EN-decl><EN-N><N><n>X</n></N></EN-N></EN-
decl>, <EN-decl><EN-N><N><n>Y</n></N></EN-N></EN-decl>, <EN-decl><EN-
N><N><n>Z</n></N></EN-N></EN-decl></EN-decl-LT></T-decl-stmt>

<a-stmt><E-1><named-E><N><n>X</n></N></named-E></E-1> <a>=</a> <E-2><op-
E><named-E><N><n>Y</n></N></named-E> <op><o>+</o></op>
<named-E><N><n>Z</n></N></named-E></op-E></E-2></a-stmt>

<call-stmt>CALL <procedure-designator><named-E><N><n>SUB</n></N></named-
E></procedure-designator> (<arg-spec><arg><named-E><N><n>X</n></N></named-
E></arg>, <arg><named-E><N><n>Y</n></N></named-E></arg>, <arg><named-
E><N><n>Z</n></N></named-E></arg></arg-spec>)</call-stmt>

<end-program-stmt>END</end-program-stmt></program-unit>
```

Editing with XML DOM

```
$ fxtran main.F90

my $doc = 'XML::LibXML'->load_xml (location =>
'main.F90.xml');
```

- parentNode
- nextSibling
- unbindNode
- replaceNode
- firstChild
- cloneNode
- insertBefore
- ...

Searching with XML XPath

```
$ fxtran main.F90


$doc = 'XML::LibXML'->load_xml (location => 'main.F90.xml');

# SUBROUTINE SHALLOW_MF (KLON,KIDIA,KFDIA,KLEV,KSV,KKA,...)
($pu) = &f ('./f:object/f:file/f:program-unit', $doc);

$stmt = $pu→firstChild;

# KLON,KIDIA,KFDIA,...

($darglt) = &f ('./f:dummy-arg-LT', $stmt);

@darglt = map { $_->textContent } &f ('./f:arg-N', $darglt);
```

```
# INTEGER, INTENT (IN) :: KLON

($stmt) = &f ('.//f:T-decl-stmt'
            . '[.//f:EN-decl/f:EN-N/f:N/f:n/text ()="KLON"]',
            $pu);

# Find end of line

($cr) = &f ('following::text ()'
          . '[contains (., "' . "\n" . '")]', $stmt);

# Find loop where index is JLON and ZGAGT0M is modified

my ($do) = &f ('./f:do-construct'
    . '[./f:do-stmt/f:do-V/f:named-E/f:N/f:n/text()="JLON"]'
    . '[.//f:a-stmt/f:E-1/f:named-E/f:N/f:n/text()="ZGAGT0M"]',
              $doc);
```