Jancy

LLVM-based scripting language for IO and UI programming

Vladimir Gladkov
Tibbo Technology Inc

http://tibbo.com/jancy

Overview

- Why?
- 2 main Jancy features
- Compiler design and how we use LLVM
- Questions

Why?! Do we need more?

Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

nteraction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export

List of programming languages

From Wikipedia, the free encyclopedia

The aim of this **list of programming languages** is to include all notable programming languages in existence, both those in current use and historical ones, in alphabetical order, except for dialects of BASIC and esoteric programming languages.

***Note**: Dialects of BASIC have been moved to the separate List of BASIC dialects.*

***Note**: This page does not list esoteric programming languages.*

Programming language lists

- **Alphabetical**
- Categorical
- Chronological
- Generational

V·T·E

A · B · C · D · E · F · G · H · I · J · K · L · M · N · O · P · Q · R · S · T · U · V · W · X · Y · Z

Contents : See also

A [edit]

- A# .NET
- A# (Axiom)
- A-0 System
- A+
- A++
- ABAP
- ABC
- Ada
- Adenine
- Agda
- Agilent VEE
- Agora
- AIMMS
- Alef
- Apex (Salesforce.com)
- API
- ARexx

~700 already!!

Wanted! (for IO Ninja)

- IO
  - Safe pointer arithmetic
  - High level of source compatibility with C
  - Built-in incremental lexer generator

Wanted! (for IO Ninja)

- IO
  - Safe pointer arithmetic
  - High level of source compatibility with C
  - Built-in incremental lexer generator
- UI
  - Properties
  - Events
  - Excel-like “reactive“ evaluation

Jancy Design Goals

- Embedded scripting language
- Statically typed
- C-family language syntax
- ABI-compatible with C
- Garbage collected (accurate GC)
- LLVM as back-end

Other interesting features

- Const-correctness
- Multiple inheritance
- Partial application
- Schedulers
- Exception-style syntax over error code checks
- Dual type modifiers
- Bigendian integers
- Bitflag enums
- Break-n/Continue-n
- Hex literals

Handling binary data (wrong)

```
public class IPv4Packet {
    private static final int IP_TOS_POS = 1;  // type of service
    private static final int IP_LEN_POS = 2;  // total packet length
    private static final int IP_ID_POS = 4;   // the packet id
    private static final int IP_FRAG_POS = 6; // the frag flags and offset
    // ...
    public int getTypeOfService() {
        if (_isReadTOS == false) {
            myTOS = myPacket[myIPHdrOffset + IP_TOS_POS] & 0x0f;
            _isReadTOS = true;
        }
        return myTOS;
    }
    public int getFragmentFlags() {
        if (_isReadFragFlags == false) {
            _isReadFragFlags = true;
            myFragmentFlags = ByteUtils.getByteNetOrderTo_uint16(
                myPacket, myIPHdrOffset + IP_FRAG_POS) >> 13;
        }
        return myFragmentFlags;
    }
    // ...
}
```

Handling binary data (wrong)

```
public class IPv4Packet {
    private static final int IP_TOS_POS = 1;  // type of service
    private static final int IP_LEN_POS = 2;  // total packet length
    private static final int IP_ID_POS = 4;   // the packet id
    private static final int IP_FRAG_POS = 6; // the frag flags and offset
    // ...
    public int getTypeOfService() {
        if (_isReadTOS == false) {
            myTOS = myPacket[myIPHdrOffset + IP_TOS_POS] & 0x0f;
            _isReadTOS = true;
        }
        return myTOS;
    }
    public int getFragmentFlags() {
        if (_isReadFragFlags == false) {
            _isReadFragFlags = true;
            myFragmentFlags = ByteUtils.getByteNetOrderTo_uint16(
                myPacket, myIPHdrOffset + IP_FRAG_POS) >> 13;
        }
        return myFragmentFlags;
    }
    // ...
}
```

Handling binary data (wrong)

```
public class IPv4Packet {
    private static final int IP_TOS_POS = 1;  // type of service
    private static final int IP_LEN_POS = 2;  // total packet length
    private static final int IP_ID_POS = 4;   // the packet id
    private static final int IP_FRAG_POS = 6; // the frag flags and offset
    // ...
    public int getTypeOfService() {
        if (_isReadTOS == false) {
            myTOS = myPacket[myIPHdrOffset + IP_TOS_POS] & 0x0f;
            _isReadTOS = true;
        }
        return myTOS;
    }
    public int getFragmentFlags() {
        if (_isReadFragFlags == false) {
            _isReadFragFlags = true;
            myFragmentFlags = ByteUtils.getByteNetOrderTo_uint16(
                myPacket, myIPHdrOffset + IP_FRAG_POS) >> 13;
        }
        return myFragmentFlags;
    }
    // ...
}
```

Handling binary data (right)

Step #1 – Define data layout

```
struct IpHdr
{
    uint8_t m_headerLength : 4;
    uint8_t m_version      : 4;
    uint8_t m_typeOfService;
    // ...
}

struct IcmpHdr
{
    uint8_t m_type;
    uint8_t m_code;
    bigendian uint16_t m_checksum;
    // ...
}
```

Handling binary data (right)

Step #2 – Access buffer

```
printIpHdr (void const* buffer)
{
    IpHdr const* ipHdr = (IpHdr const*) buffer;

    print ($"IP version = $(ipHdr.m_version)\n“);
    // ...

    if (ipHdr.m_protocol == IPPROTO_ICMP)
    {
        buffer += ipHdr.m_headerLength * 4;
        IcmpHdr const* icmpHdr = (IcmpHdr const*) buffer;

        print ($“ICMP type = $(icmpHdr.m_type)\n“);
        // ...
    }
}
```

Handling binary data (right)

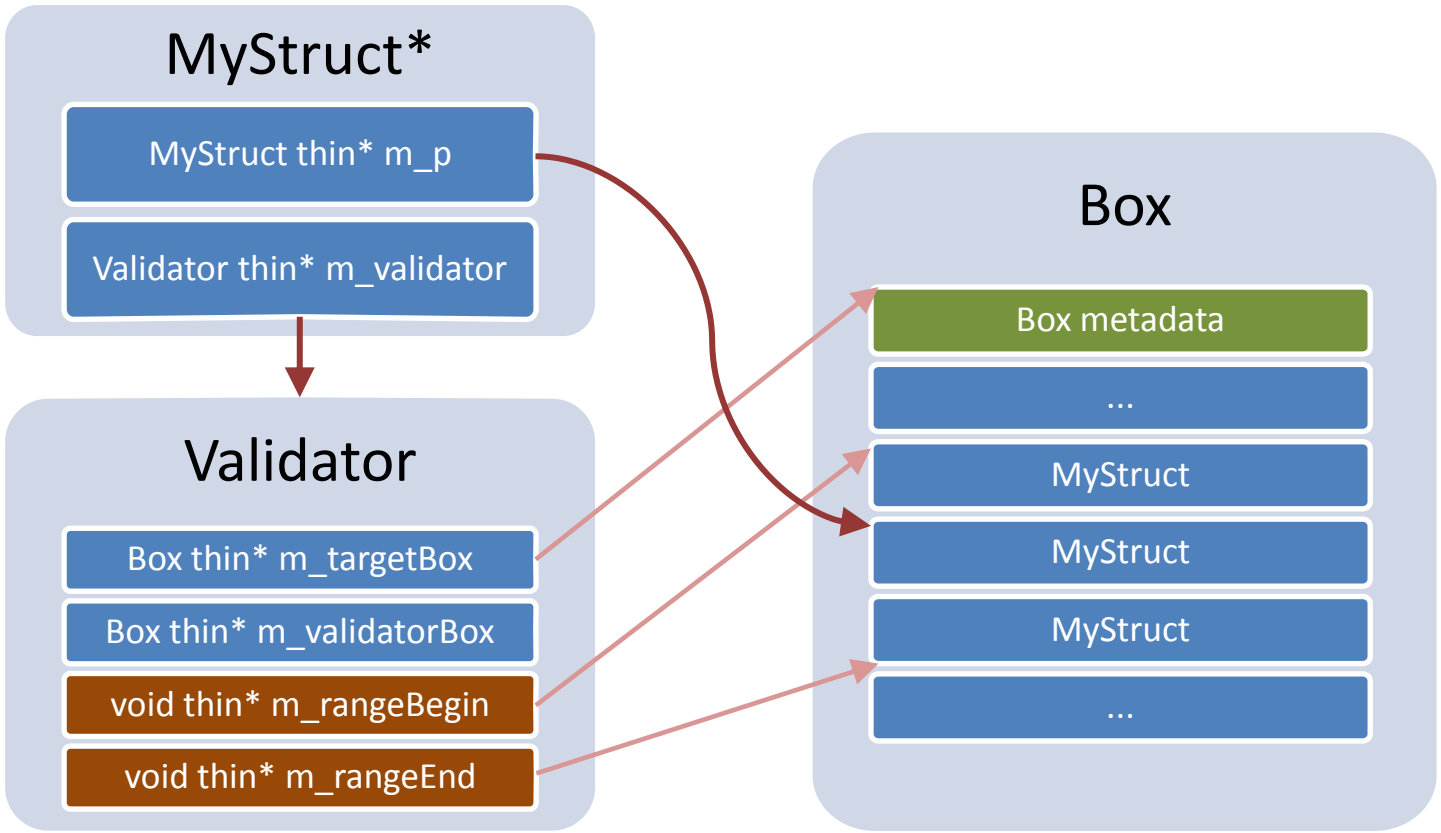
Step #2 – Access buffer

```
printIpHdr (void const* buffer)
{
    IpHdr const* ipHdr = (IpHdr const*) buffer;

    print ($"IP version = $(ipHdr.m_version)\n“);
    // ...

    if (ipHdr.m_protocol == IPPROTO_ICMP)
    {
        buffer += ipHdr.m_headerLength * 4;
        IcmpHdr const* icmpHdr = (IcmpHdr const*) buffer;

        print ($“ICMP type = $(icmpHdr.m_type)\n“);
        // ...
    }
}
```

How is pointer arithmetic safe?

Fat pointers, obviously

```
MyStruct* p;
```



Loads/stores are bounds checked

Pointer dereference

```
foo (char* p, size_t i)
{
    p += i;
    *p = 10; // <-- range is checked
}
```

Array indexing

```
bar (size_t i)
{
    static int a [] = { 10, 20, 30 };
    int x = a [i]; // <-- range is checked
}
```

Dynamic sizeof/countof

```
foo (int* a)
{
    size_t count = dynamic countof (a);
    for (size_t i = 0; i < count; i++)
    {
        // do something with a [i]
    }
}
```

Are bounds checks enough?

- Dangling pointers?
- Unions?
- Reinterpret casts?
- Pointer-to-fields increments?
- Downcasts?

Are bounds checks enough?

- Dangling pointers – impossible in Jancy
- Unions
- Reinterpret casts

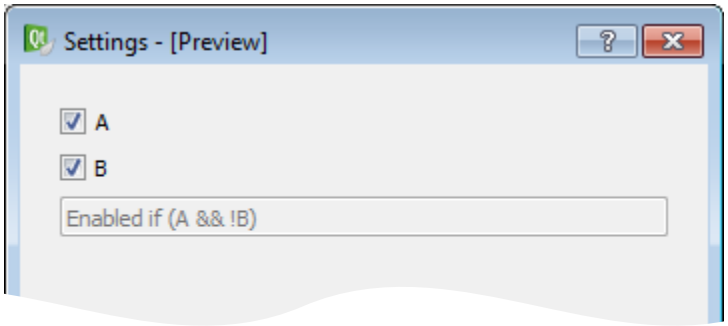
} only when safe

- Pointer-to-fields increments – range-controlled
- Downcasts – dynamic casts

```
foo (Parent* a)
{
    Child* c = dynamic (Child*) a;
    // ...
}
```

Reactive Programming for UI

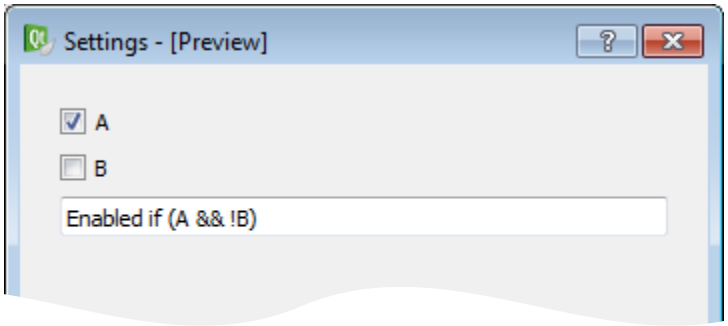
- Automatic propagation of changes
- Observer/Observable pattern
- Our goal: Excel-like re-evaluation for UI
- Our workhorses:
  - Multicasts & events
  - Properties




```
m_editBox.m_isEnabled =
    m_checkBoxA.m_isChecked &&
    !m_checkBoxB.m_isChecked;
```

Reactive Programming for UI

- Automatic propagation of changes
- Observer/Observable pattern
- Our goal: Excel-like re-evaluation for UI
- Our workhorses:
  - Multicasts & events
  - Properties




```
m_editBox.m_isEnabled =
    m_checkBoxA.m_isChecked &&
    !m_checkBoxB.m_isChecked;
```

Multicasts & events

```
class C1
{
    event m_onComplete ();

    work ()
    {
        // ...
        m_onComplete (); // OK, 'call' is accessible from C1
    }
}

foo (C1* c)
{
    multicast m (int);
    m += bar;
    m += baz;
    m (100);  // <-- foo (100); bar (100);

    c.m_onComplete (); // <-- error, 'call' is inaccessible
}
```

Bindable properties

```
int bindable property g_bindableProp;
```

```
g_bindableProp.set (int x)
{
    if (x == m_value)
        return;

    m_value = x;
    m_onChanged (); // compiler-generated event is 'm_onChanged'
}
```

```
onPropChanged ()
{
    // ...
}

foo ()
{
    bindingof (g_bindableProp) += onPropChanged;
    g_bindableProp = 100; // onPropChanged will be called
}
```

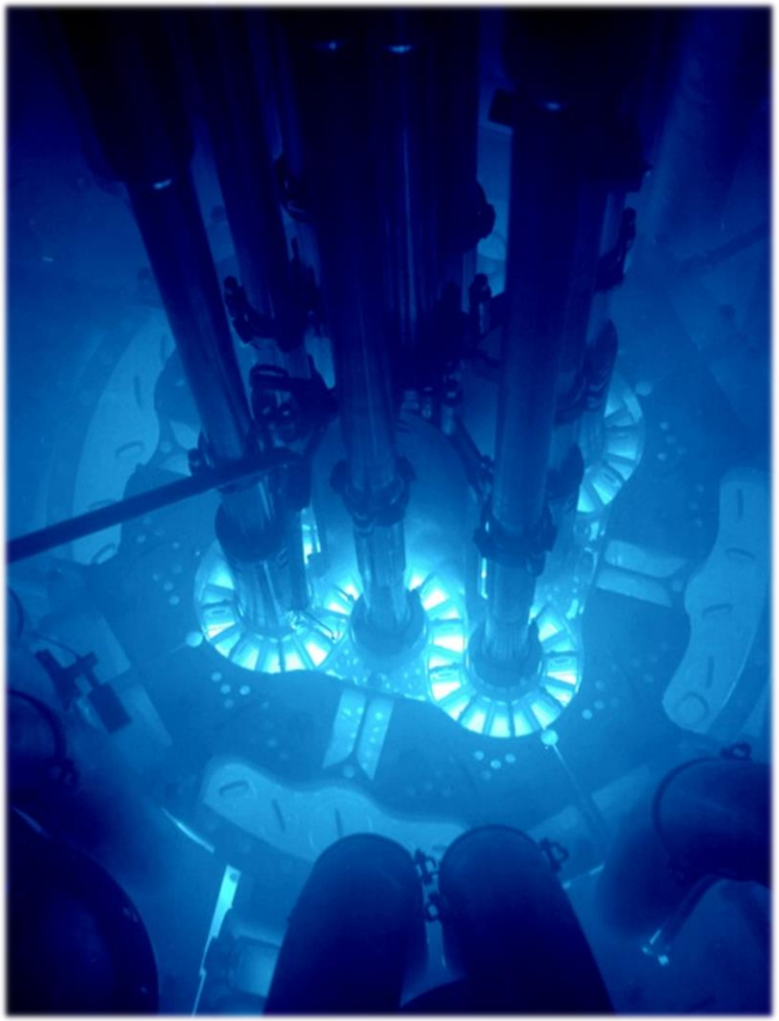
Dilemma

- We want Excel-like re-evaluation

Dilemma

- We want Excel-like re-evaluation
- Implicit observers are hard to control

Solution – reactors!



Solution – reactors!

```
reactor TcpConnectionSession.m_uiReactor ()
{
    m_title = $"TCP $(m_addressCombo.m_editText)";
    m_isTransmitEnabled = m_state == State.Connected;
    m_actionTable [ActionId.Disconnect].m_isEnabled = m_state != State.Closed;
    m_adapterProp.m_isEnabled = m_useLocalAddressProp.m_value;
    m_localPortProp.m_isEnabled = m_useLocalAddressProp.m_value;
}
```

Solution – reactors!

```
reactor TcpConnectionSession.m_uiReactor ()
{
    m_title = $"TCP $(m_addressCombo.m_editText)";
    m_isTransmitEnabled = m_state == State.Connected;
    m_actionTable [ActionId.Disconnect].m_isEnabled = m_state != State.Closed;
    m_adapterProp.m_isEnabled = m_useLocalAddressProp.m_value;
    m_localPortProp.m_isEnabled = m_useLocalAddressProp.m_value;
}
```

Automated, but controlled

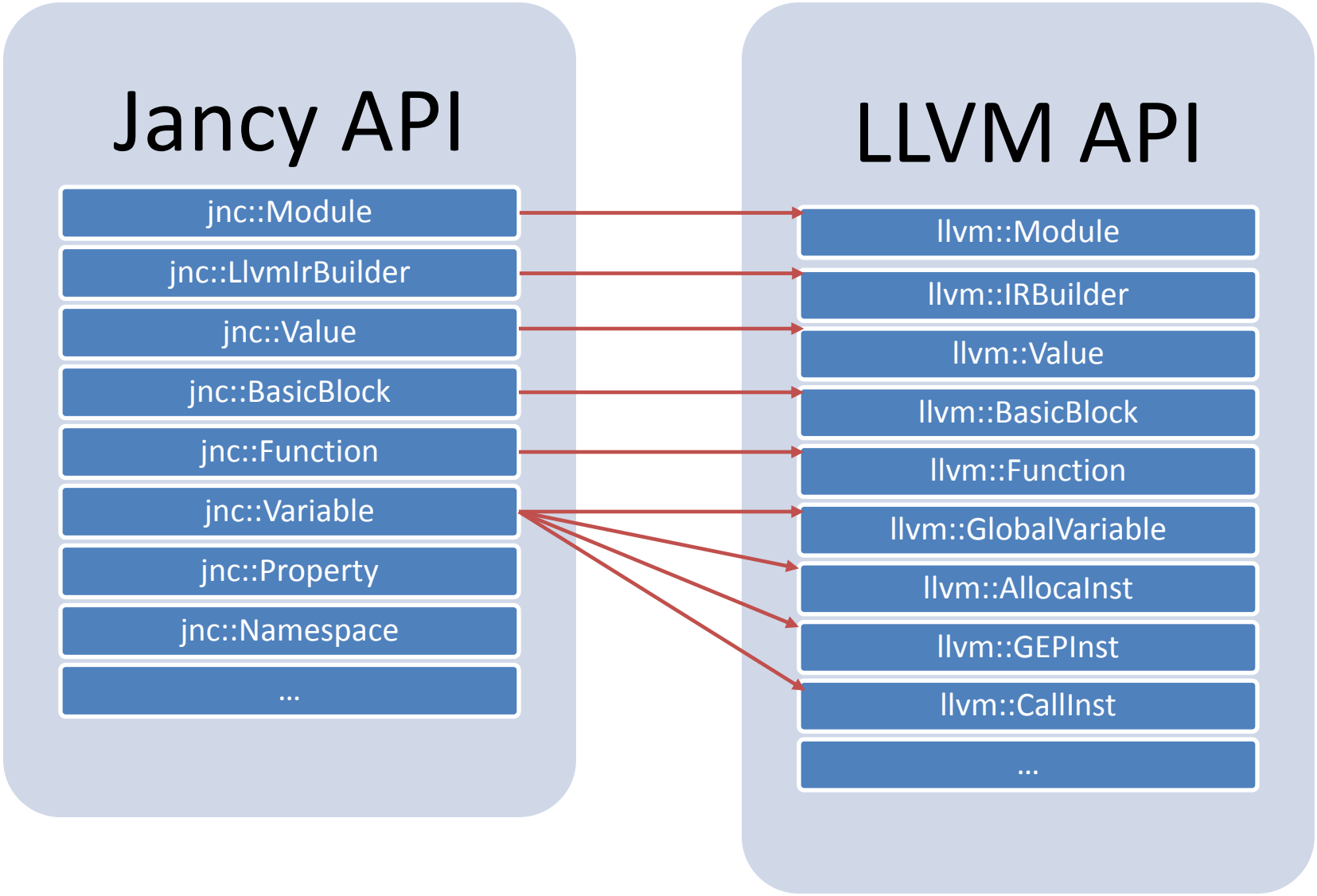
```
reactor m_uiReactor ()
{
    m_title = $"TCP $(m_addressCombo.m_editText)";
    m_isTransmitEnabled = m_state == State.Connected;
    // ...

    onevent m_transmitButton.m_onClicked ()
    {
        // handle start button click...
    }

    onevent (m_userEdit.m_onChanged, m_passwordEdit.m_onChanged) ()
    {
        // handle login change...
    }
}
```

```
m_uiReactor.start ();
// ...
m_uiReactor.stop ();
```

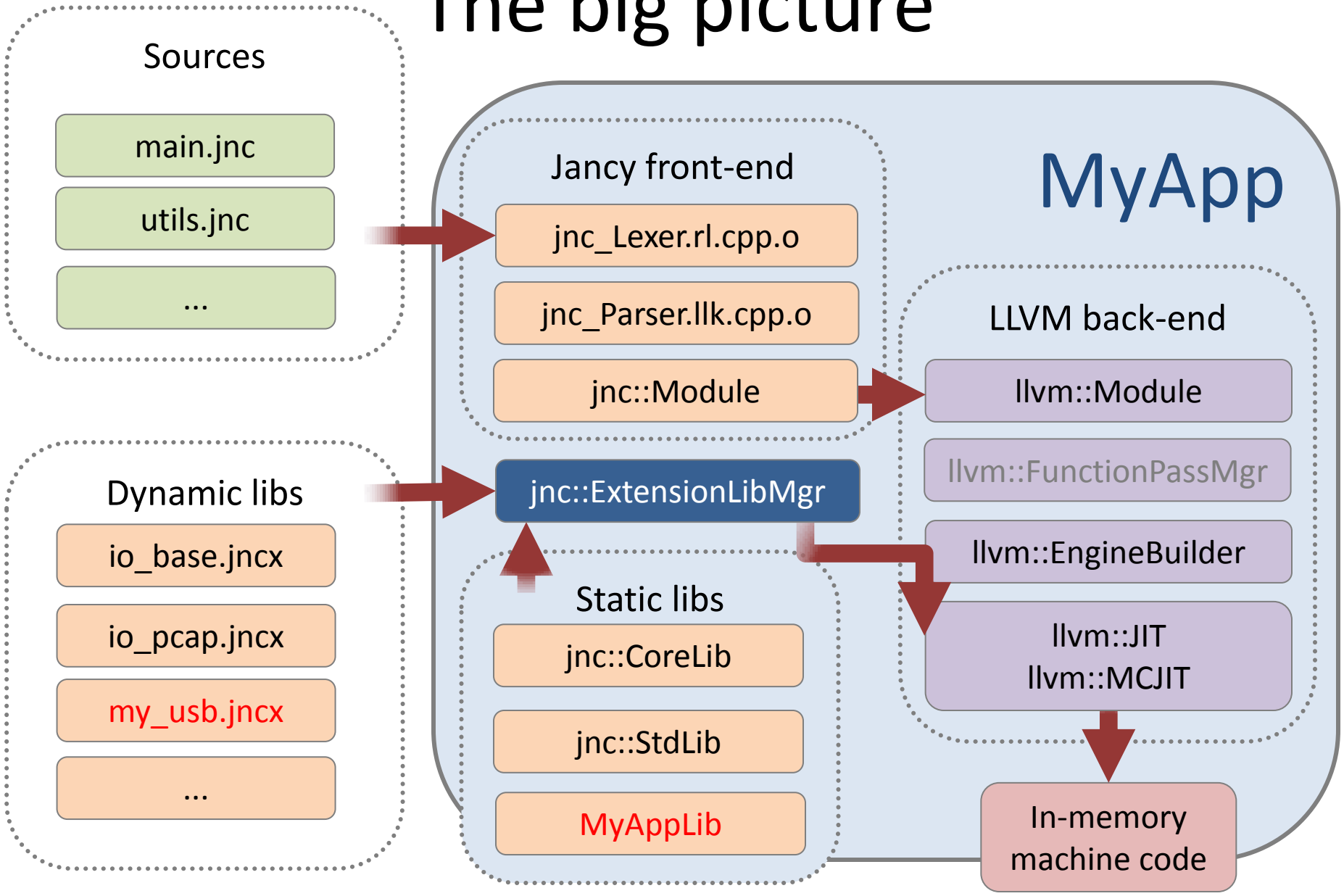
Implementation

- Main goal: embedded scripting
- Ragel-generated lexer as a front-end
- Table-driven generated top-down parser
- LLVM API to generate in-memory IR
- LLVM JIT to machine code
- Plugins for NetBeans IDE

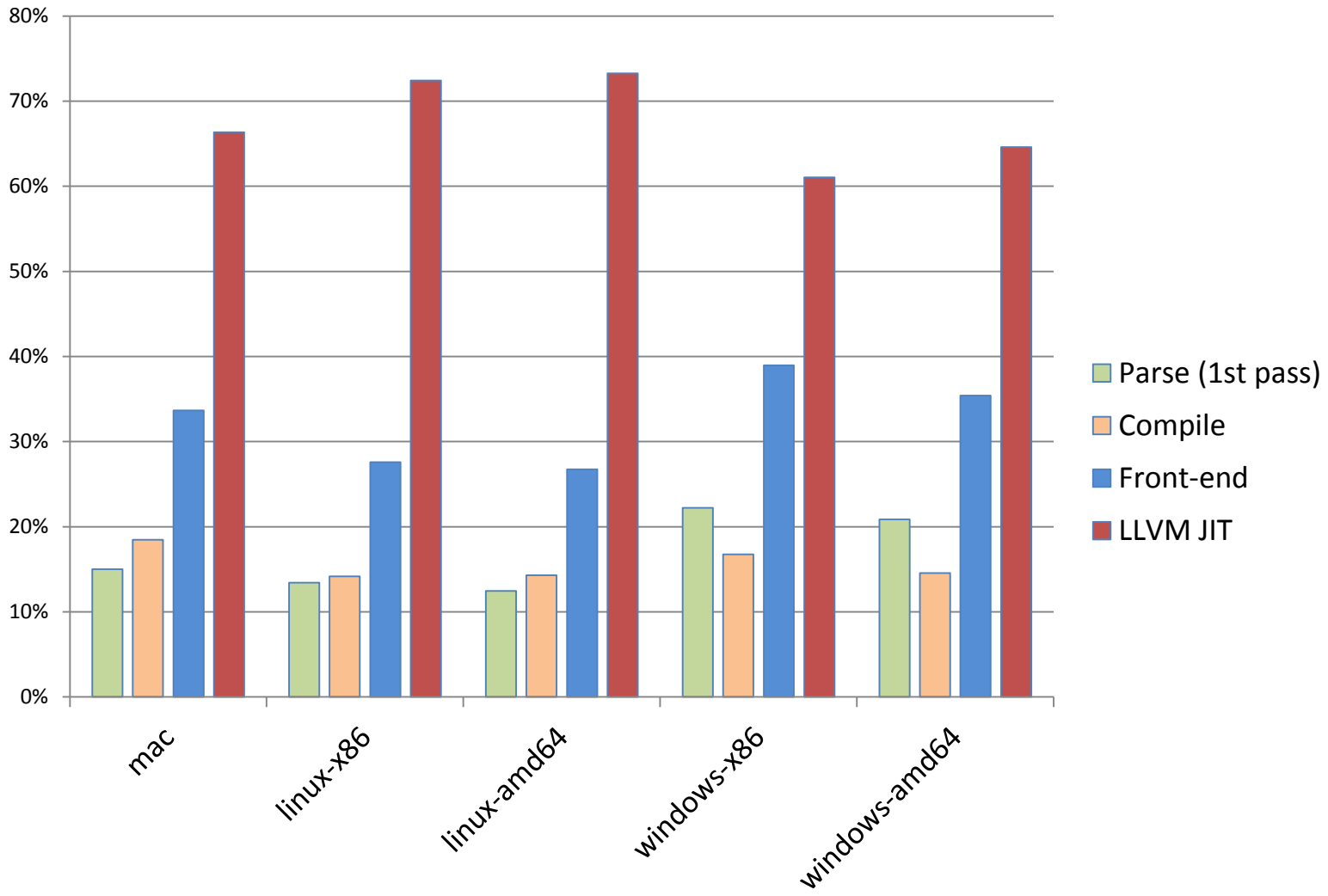
jnc::Module vs llvm::Module



The big picture



Where is time spent?



Summary

- Open source LLVM-based scripting language
- Offers unique features
- Used in a real-life product IO Ninja
- Comes with NetBeans-based IDE
- Live demo on the website
- Play, contribute, use in your projects

http://tibbo.com/jancy

vovkos@tibbo.com