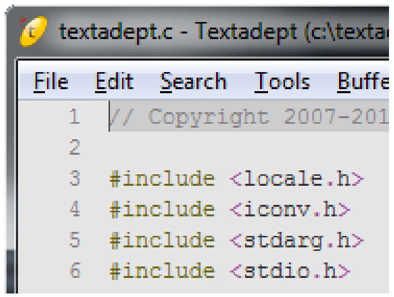
Textadept

Behind the Scenes

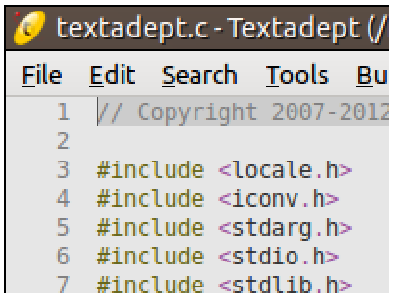
By Mitchell
Lua Workshop 2012

Outline

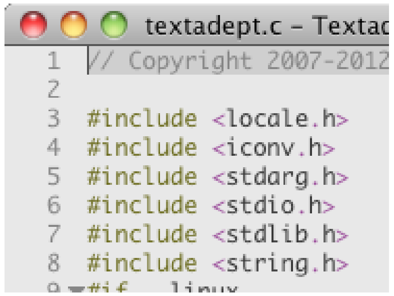
- Introduction
- Lua in Textadept
  - Syntax highlighting
  - Code completion
  - UI Scripting
    - Editing component
- Q & A



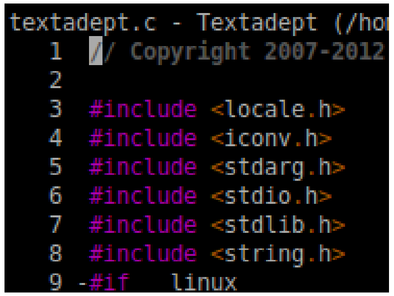
```
// Copyright 2007-201

#include <locale.h>
#include <iconv.h>
#include <stdarg.h>
#include <stdio.h>
```



```
// Copyright 2007-2012

#include <locale.h>
#include <iconv.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
```



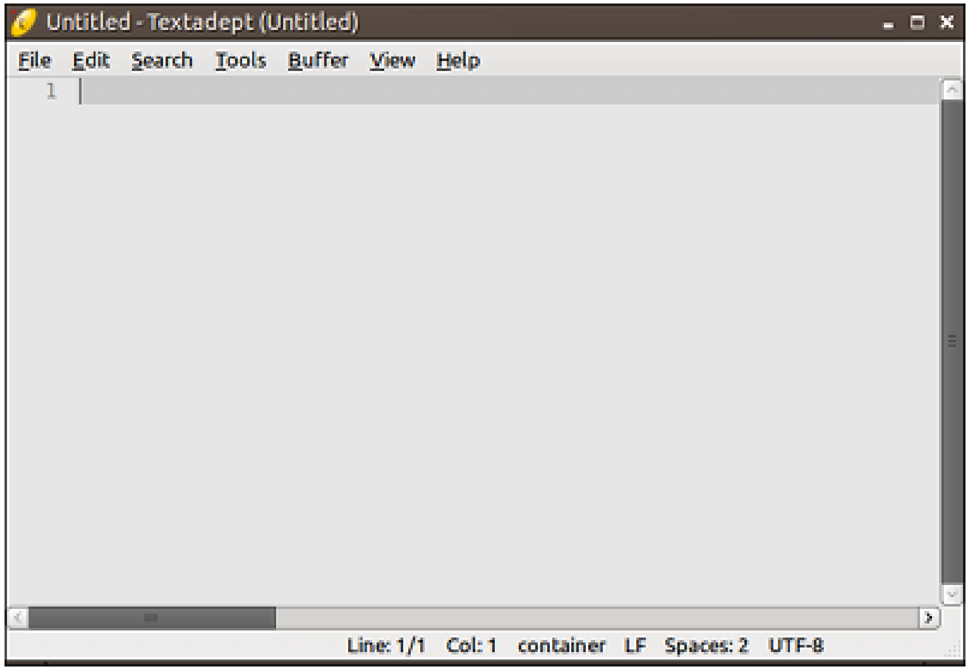
```
// Copyright 2007-2012

#include <locale.h>
#include <iconv.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#if   linux
```



```
// Copyright 2007-2012

#include <locale.h>
#include <iconv.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
-#if   linux
```

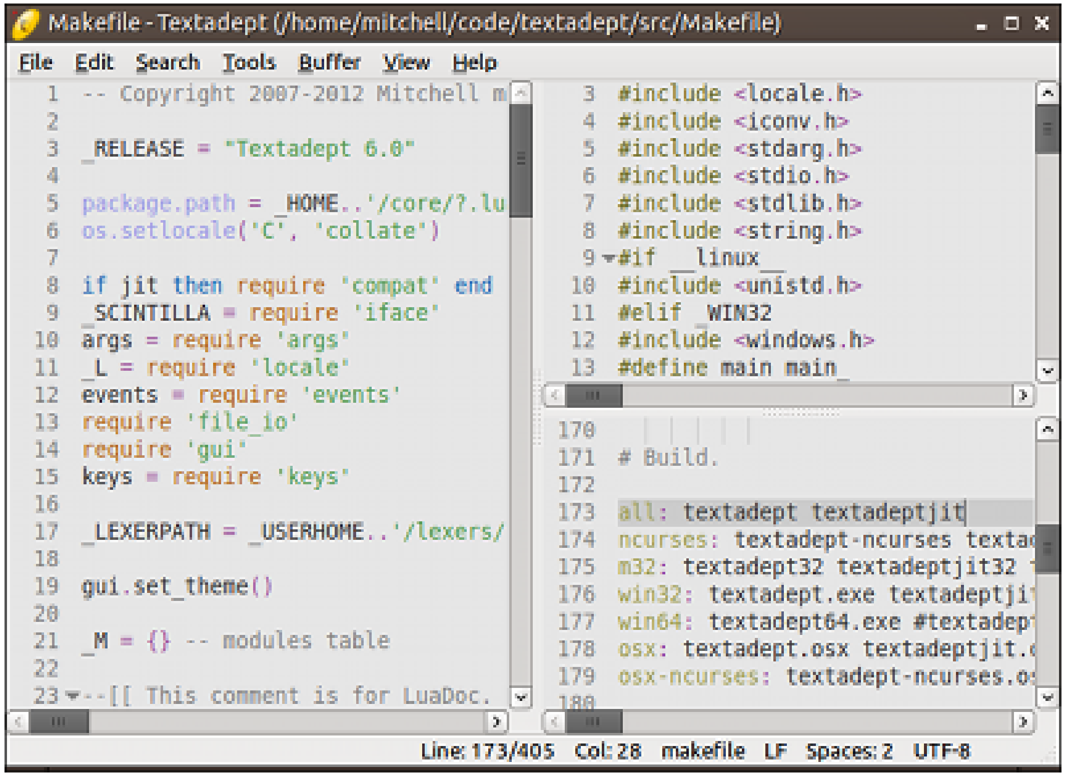
Introduction

- Why Textadept?
- Why Lua?
- Editor design


Untitled - Textadept (Untitled)
File Edit Search Tools Buffer View Help
Line: 1/1 Col: 1 container LF Spaces: 2 UTF-8

Syntax Highlighting

- Pattern matching
  - Regex
  - Character iteration
  - LPeg


Makefile - Textadept (/home/mitchell/code/textadept/src/Makefile)
File
Edit
Search
Tools
Buffer
View
Help

```
-- Copyright 2007-2012 Mitchell m

_RELEASE = "Textadept 6.0"

package.path = _HOME..'/core/?.lu
os.setlocale('C', 'collate')

if jit then require 'compat' end
_SCINTILLA = require 'iface'
args = require 'args'
_L = require 'locale'
events = require 'events'
require 'file_io'
require 'gui'
keys = require 'keys'

_LEXERPATH = _USERHOME..'/lexers/

gui.set_theme()

_M = {} -- modules table

--[[ This comment is for LuaDoc.
```

```
#include <locale.h>
#include <iconv.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#if __linux__
#include <unistd.h>
#elif _WIN32
#include <windows.h>
#define main main_
```

```

# Build.

all: textadept textadeptjit
ncurses: textadept-ncurses textad
m32: textadept32 textadeptjit32 t
win32: textadept.exe textadeptji
win64: textadept64.exe #textadep
osx: textadept.osx textadeptjit.
osx-ncurses: textadept-ncurses.o
```

Line: 173/405 Col: 28 makefile LF Spaces: 2 UTF-8

Syntax Highlighting with LPeg

- Tokens
  - Whitespace
  - Comments
  - Strings
  - Etc.
- Rules
- Grammars

```
l = lexer -- lexer module

ws = l.token(l.WHITESPACE, l.space^1)

ls = [...] -- long string pattern

lc = '--' * l.nonnewline^0
bc = '--' * ls
comment = l.token(l.COMMENT, bc + lc)

sq = l.delimited_range("'", '\\', true)
dq = l.delimited_range('"', '\\', true)
string = l.token(l.STRING, sq + dq + ls)

_rules = {
  {'whitespace', ws},
  [...], -- keywords, functions, etc.
  {'string', string},
  {'comment', comment},
  [...], -- numbers, labels, operators
} --> compiles to a grammar
```

Behind the Scenes

- Load lexer
- Build grammar
- Call `lpeg.match`
- Highlight text

```
lpeg.match(lua._GRAMMAR, [[
-- comment
local foo=10
print('foo')]]) -->

{'comment',    10, -- "-- comment"
 'whitespace', 11, -- newline
 'keyword',    16, -- "local"
 'whitespace', 17, -- space
 'identifier', 20, -- "foo"
 'operator',   21, -- "="
 'number',     23, -- "10"
 'whitespace', 24, -- newline
 'function',   29, -- "print"
 'operator',   30, -- "("
 'string',     35, -- "'foo'"
 'operator',   36} -- ")"
```

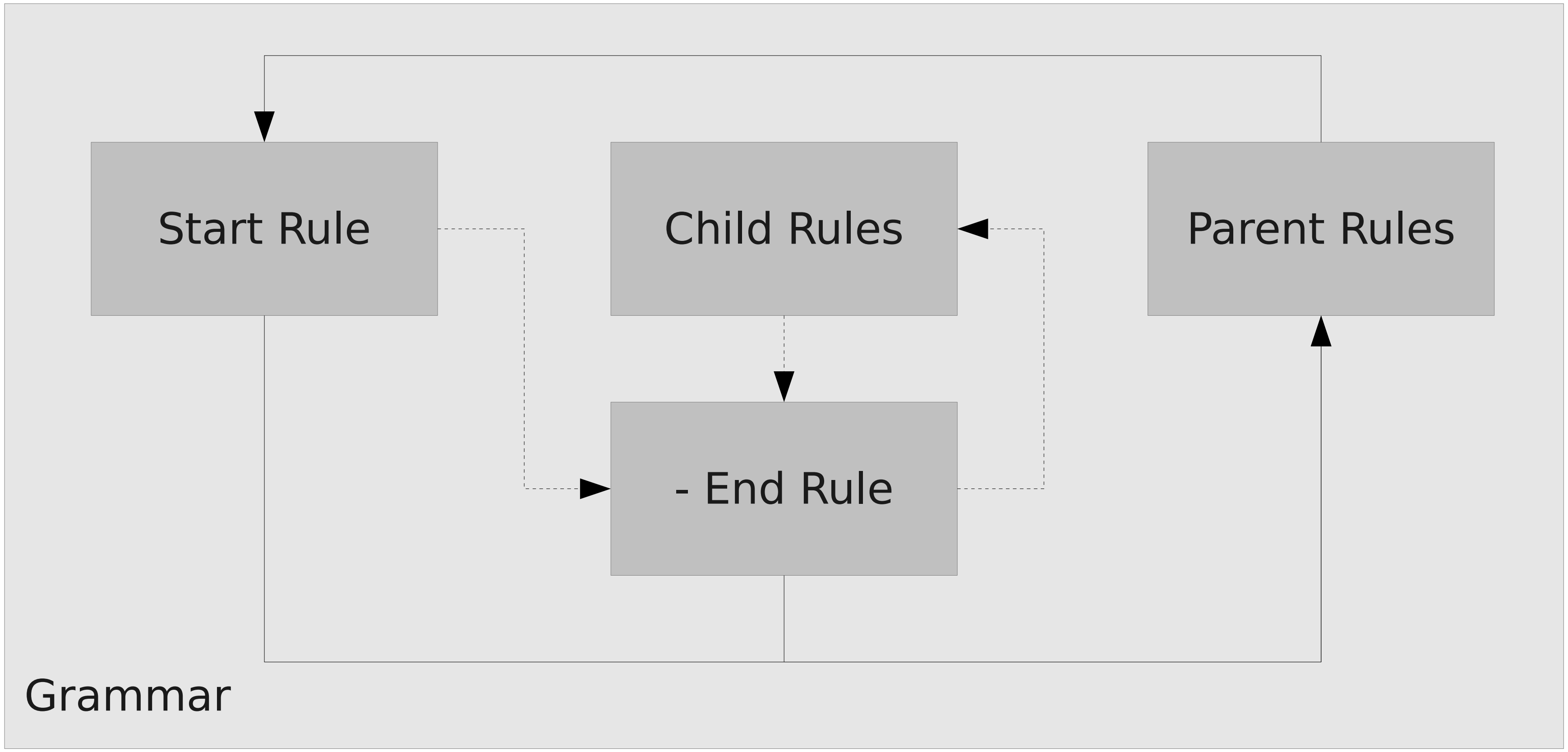
Embedded Languages

- Load lexer
- Start/end rules
- Call one function

```
-- html.lua lexer

l = lexer -- lexer module

[...] -- html patterns, rules, etc.

-- Embedded Lua.
lua = l.load('lua')
s_tag = lpeg.P('<?lua') * l.space^1
e_tag = lpeg.P('?>')
start = l.token(l.TAG, s_tag)
stop = l.token(l.TAG, e_tag)
l.embed_lexer(M, lua, start, stop)
```

Behind the Scenes

```
l.embed_lexer(parent, child, start_rule, end_rule)
```

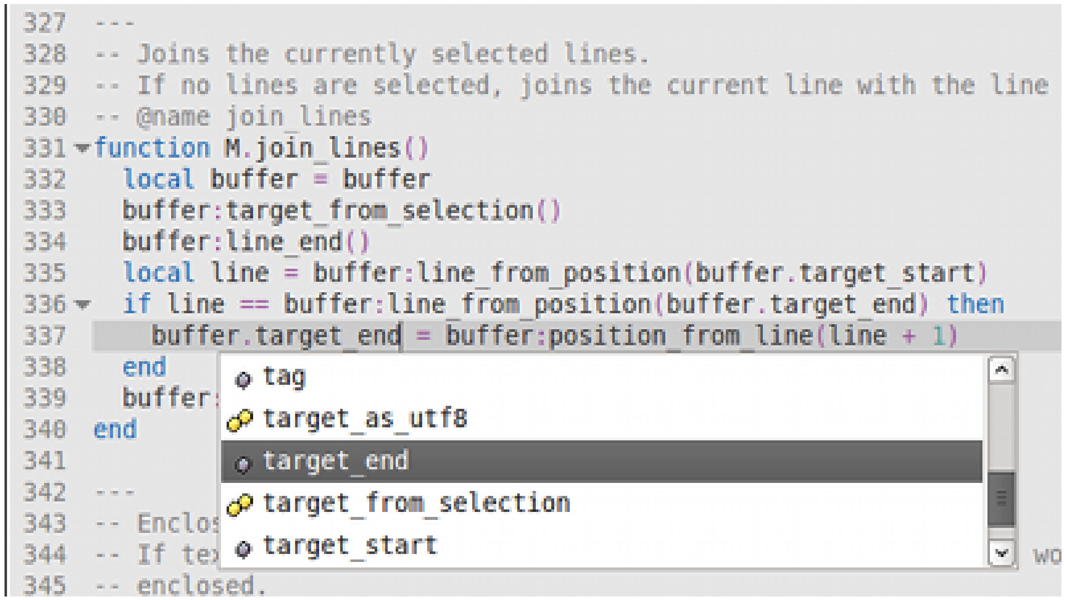


More LPeg

- Common language syntax patterns
  - Delimited ranges with escape, balanced, and forbidden characters
    ```
    l.delimited_range('()', '\\', false, true, '\n')
    ```
  - Nested pairs
    ```
    l.nested_pair('/+', '+/', true)
    ```
  - Beginning of lines
    ```
    l.starts_line('#' * l.nonnewline^0)
    ```
  - Words in a list
    ```
    l.word_match{'foo', 'bar', 'baz'}
    -- vs. lpeg.P('foo') + lpeg.P('bar') + lpeg.P('baz')
    ```

Code Completion

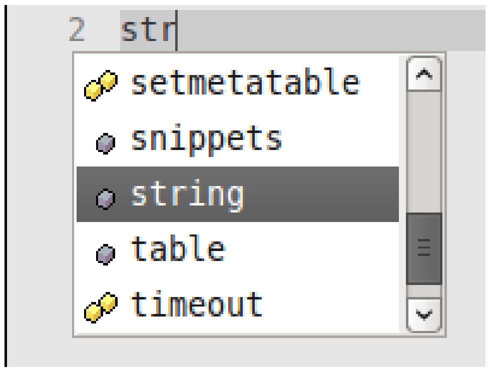
- Ctags
- Introspection
- Parser/AST
- Hybrid

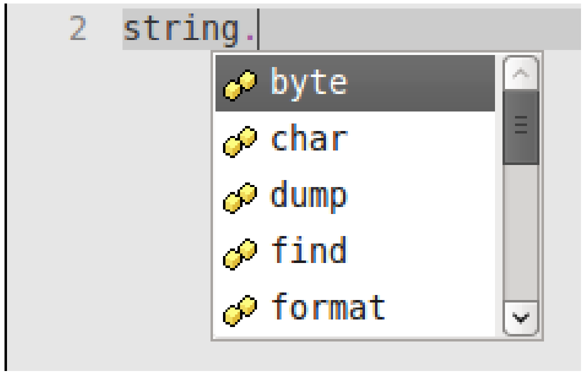


```
---
-- Joins the currently selected lines.
-- If no lines are selected, joins the current line with the line
-- @name join_lines
function M.join_lines()
  local buffer = buffer
  buffer:target_from_selection()
  buffer:line_end()
  local line = buffer:line_from_position(buffer.target_start)
  if line == buffer:line_from_position(buffer.target_end) then
    buffer.target_end = buffer:position_from_line(line + 1)
  end
  buffer:
end

---
-- Enclos
-- If tex                                                          wo
-- enclosed.
```

Adeptsense

- Ctags-ish + type/class inference with pattern matching
- Classes
  - Functions
  - Fields





Behind the Scenes

```
# Ctags-like tags
string _ 0;" m
byte _ 0;" f class:string
char _ 0;" f class:string
dump _ 0;" f class:string
find _ 0;" f class:string
format _ 0;" f class:string
gmatch _ 0;" f class:string
gsub _ 0;" f class:string
len _ 0;" f class:string
lower _ 0;" f class:string
match _ 0;" f class:string
rep _ 0;" f class:string
reverse _ 0;" f class:string
sub _ 0;" f class:string
upper _ 0;" f class:string
table _ 0;" m
...
```

```
-- Lua Adeptsense
sense.completions = {
  ['string'] = {
    functions = {
      'byte', 'char', 'dump',
      'find', 'format', 'gmatch',
      'gsub', 'len', 'lower',
      'match', 'rep', 'reverse',
      'sub', 'upper'
    },
    fields = {}
  },
  ['table'] = {[...]}, -- etc.
  [...] -- etc.
}
```

Type/Class Inference Patterns

- Class definition + “self”

```
# Ruby
class String
  def foo
    self.    #=> string completions
```

- Type declaration

```
// Java
String foo;
foo.         //> string completions
```

- Type assignment

```
-- Lua
foo = "foo"
foo:         --> string completions
```

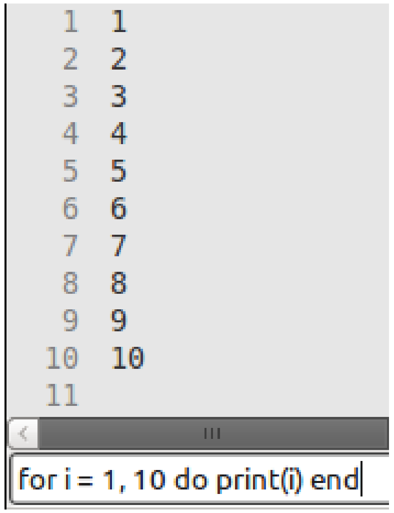
Behind the Scenes

- Simple pattern matching upwards
  - False positives
  - No return type inference
- Can subclass Adeptsense methods

```lua
function sense:get_class(symbol)
  if condition then
    return self.super.get_class(self, symbol) -- default behavior
  else
    -- different behavior
  end
end
```

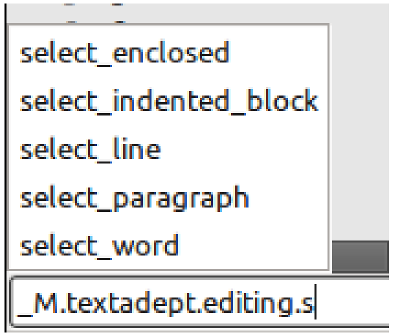
C ↔ Lua

- Most C is Lua interface code
  - UI scripting
    - Metatables and callbacks
      - Text fields
      - Menus
      - Events
      - Etc.
    - Editing component



```
1
2
3
4
5
6
7
8
9
10
```

```
for i = 1, 10 do print(i) end
```



```
select_enclosed
select_indented_block
select_line
select_paragraph
select_word
```

```
_M.textadept.editing.s
```

Editing Component

- Scintilla
- Lua-friendly API
  - `SCI_`*`MESSAGE`*`(lParam, wParam)`
    - `SCI_INSERTTEXT(int pos, char *text)`
      `→ buffer:insert_text(0, "foo")`
    - `SCI_GETCHARAT(int pos, void)`
      `→ buffer.char_at[0]`
    - `SCI_SETEOLMODE(int eolmode, void)`
      `→ buffer.eol_mode = 2`

Behind the Scenes

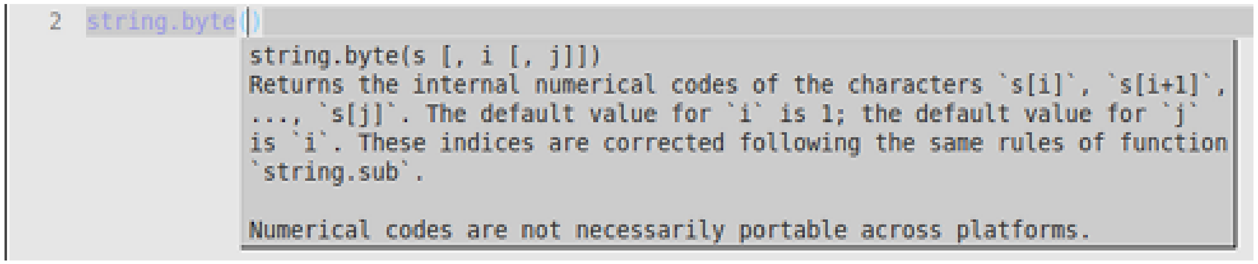
- “`buffer`” has `__index` and `__newindex`
- Scintilla messages have IDs
- Functions vs. Properties
  - `{id, return_type, wParam_type, lParam_type}`
  - `{get_id, set_id, return_type, wParam_type}`
- Functions are easy; return callable closure

Behind the Scenes Continued

- Properties are more difficult
  `{get_id, set_id, return_type, wParam_type}`
- `wParam_type == void` → simple property
- `wParam_type ~= void` → property table
  - Return table with `__index`, `__newindex`
- `__index` → use "`get_id`"; return value
- `__newindex` → use "`set_id`"; set value

Wrap Up

- Problems in Editor Design
  - Syntax highlighting
  - Code completion
  - UI scripting
- All solvable with Lua!

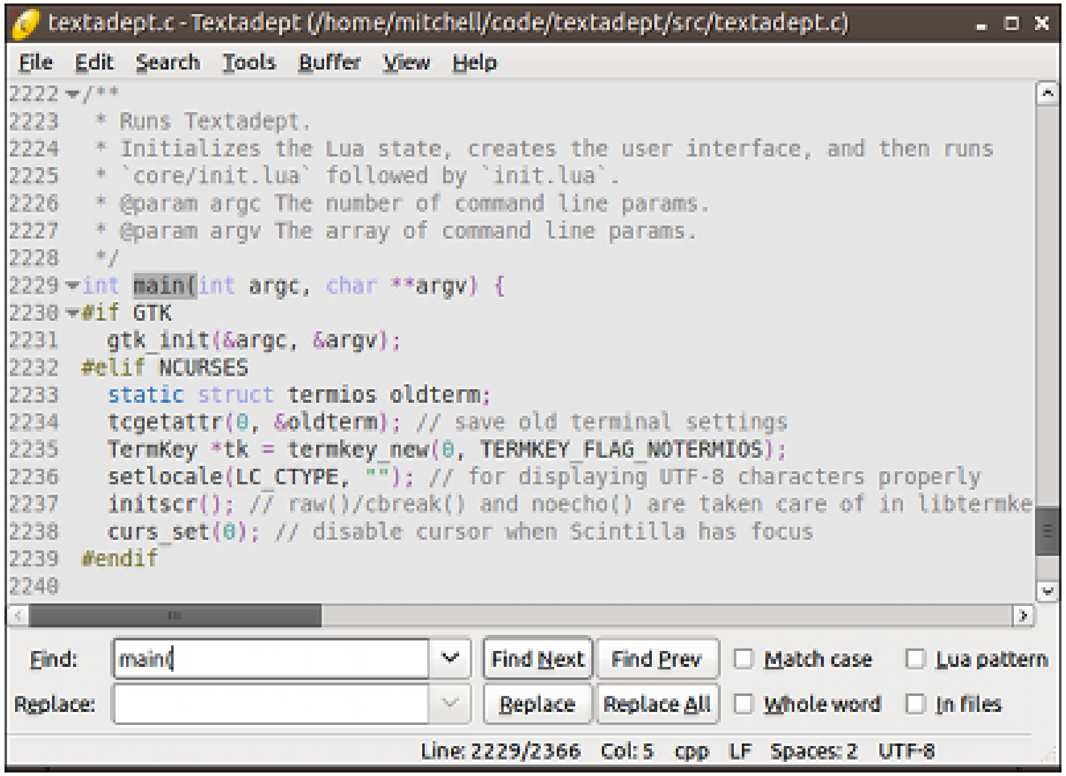


```
string.byte()
```

```
string.byte(s [, i [, j]])
Returns the internal numerical codes of the characters `s[i]`, `s[i+1]`,
..., `s[j]`. The default value for `i` is 1; the default value for `j`
is `i`. These indices are corrected following the same rules of function
`string.sub`.

Numerical codes are not necessarily portable across platforms.
```

Thank You

- Questions?



textadept.c - Textadept (/home/mitchell/code/textadept/src/textadept.c)

```
/**
 * Runs Textadept.
 * Initializes the Lua state, creates the user interface, and then runs
 * `core/init.lua` followed by `init.lua`.
 * @param argc The number of command line params.
 * @param argv The array of command line params.
 */
int main(int argc, char **argv) {
#if GTK
  gtk_init(&argc, &argv);
#elif NCURSES
  static struct termios oldterm;
  tcgetattr(0, &oldterm); // save old terminal settings
  TermKey *tk = termkey_new(0, TERMKEY_FLAG_NOTERMIOS);
  setlocale(LC_CTYPE, ""); // for displaying UTF-8 characters properly
  initscr(); // raw()/cbreak() and noecho() are taken care of in libtermke
  curs_set(0); // disable cursor when Scintilla has focus
#endif

```

Find: main(

Replace:

Line: 2229/2366 Col: 5 cpp LF Spaces: 2 UTF-8