Regexes

March 23, 2015

Chapter 1

Introduction

1.1 Introduction

- Introduction
- What is a regex?
- Using regexes
- Conclusion

1.2 Goal

- Understand when you might want to use a regex tool/library
- Obtain an idea how to use such a tool in practice

Chapter 2

What is a regex?

2.1 Definition

- 'In computing, a regular expression is a specific pattern that provides concise and flexible means to "match" (specify and recognize) strings of text'[1]
- Example: a Dutch postal code consist of four characters, a space and two uppercase characters, for example '1234 AB'
  - Can be implemented with a for-loop
  - Regexes are more efficient to solve this type of problem

[1] http://en.wikipedia.org/wiki/Regular_expression

2.2 Example regexes[2][3]

My nickname	`(B|b)ilderbikkel`
Find out what I like	`I like (.*)\.`
Name of alcoholic beverage	`whisk(e)?y`
Request for help	`h(e)+lp!`
C++ filename	`.*\.(h|hpp|c|cpp)`
Dutch postal code #1	`\d\d\d\d\s[A–Z][A–Z]`
Dutch postal code #2	`\d{4}\s[A–Z]{2}`
IP address	`\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}`
URL	`http://(www\.)?(\w*\.)+\w*`

[2] using Boost.Regex
[3] non-idiot-proof versions

2.3 Regex operations

- Searching
- Replacing

```
sed "s/from/to/" test.txt
```

Chapter 3

Using regexes

3.1 Universality of regexes

- Different tools might use a different syntax
- Different libraries might use a different syntax
- Different library implementations might use a different syntax
- Syntaxes similar

3.2 Options to work with regexes

- Using command-line tools
- (Using shell scripts)
- Using C++ regex libraries

3.3 Command line from command line

- Finding:

```
egrep "[0-9]{4}\s?[A-Z][A-Z]" myfile.txt
```

- Displays:

```
1234 AB
1234AB
```

- Replacing:

```
sed "s/from/to/" myfile.txt
```

3.4 Command line from C++

- Calling egrep from C++

```
#include <cstdlib>

int main()
{
  std::system("egrep "
    "\"[0-9]{4}\\s?[A-Z][A-Z]\""
    "myfile.txt");
}
```

3.5 C++ regex libraries

TR1 (Visual Studio)	std::tr1::regex
TR1 (GCC)	(never implemented)
Boost	boost::regex
C++11	std::regex
Qt	QRegExp
C++/CLI	System::Text::RegularExpressions::Regex

3.6 Finding an exact match

```
#include <cassert>
#include <string>
#include <boost/regex.hpp>

int main()
{
  const boost::regex r("\\d{4}\\s[A-Z]{2}");
  assert(boost::regex_match("1234 AB",r));
}
```

3.7 Finding a match within

```
#include <cassert>
#include <string>
#include <boost/regex.hpp>

int main()
{
  const boost::regex r("\\d{4}\\s[A-Z]{2}");
  assert(!boost::regex_match( "Is 1234 AB valid?",r));
  assert( boost::regex_search("Is 1234 AB valid?",r));
}
```

3.8 Iterate over matches

```
#include <cassert>
#include <string>
#include <boost/regex.hpp>
int main() {
  const std::string s = "(1) 1234 AB (2) 2345 BC (3)";
  const boost::regex r("\\d{4} [A-Z]{2}");
  boost::sregex_iterator i(s.begin(),s.end(),r);
  assert(i != boost::sregex_iterator());
  assert(i->str()=="1234 AB");
  ++i;
  assert(i != boost::sregex_iterator());
  assert(i->str()=="2345 BC");
  ++i;
  assert(i == boost::sregex_iterator());
}
```

3.9 Replacing

```
#include <cassert>
#include <string>
#include <boost/regex.hpp>

int main()
{
  const std::string str = "The zip code 1234AB is correct";
  const boost::regex regex("(\\d{4})()([A-Z]{2})");
  const std::string fix("($1) ($3)");
  const std::string s
    = boost::regex_replace(str, regex, fix,
        boost::match_default | boost::format_all);
  assert(s == "The zip code 1234 AB is correct");
}
```

Chapter 4

Conclusion

4.1 Conclusion

- Regexes are very powerful tools
- Be aware that no two implementations are exactly different

Chapter 5

EOF