Exercises — Matching URI with grep

version #

Assistants C/Unix 2022 <assistants@tickets.assistants.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

Contents

*https://intra.assistants.epita.fr

1 Matching URI with grep

Files to submit:

- uri_grep/uri.grep

Authorized commands:

- grep

1.1 Goal

This exercise's goal is to match valid *URI*. A *URI* is a string of character used to identify a resource; A particular case of *URI* is the well-known *URL*.

1.2 URI definition

We are going to use a slightly modified version of *URI* to ease the subject.

A generic *URI* is of the form:

```
scheme:[//host[:port]][/path][?query]
```

Note that brackets '[' ']' means it is **optional**, thus imbricated brackets imply condition. For example a port can only exist if a hostname is present.

Scheme:

- Sequence of characters beginning with a lowercase letter, followed by a non-empty combinaison of lowercase letters, digits, plus '+', period '.', or hyphen '–'.
- Followed by a colon ':'.

```
ftp:
https:
my.scheme-+7:
```

Host:

- One or more sequence of lowercase letters separated by a dot '.'.

```
localhost
example.org
blog.xavierlogin.com
```

Port:

- Preceded by a colon ':'.
- A 1 to 5 digits number. It can begin by a 0.

```
:13
:1337
:65535
```

Path:

- Preceded by a slash '/'.
- A path is made of sequence separated by slashes '/'. A sequence can be:
  - 1 or more combinaisons of letters, digits, hyphens '–', and underscores '_' separated by a single dot '.'.
  - A double dot '..'.
  - A single dot '.'.
  - A path must not end with a /.

```
/../test/foo/bar.baz/Myf1.txt
/foo/./pikachu/../my.tra.la.la
```

Query:

- Several queries can exist at the same time.
- Each query is preceded by a question mark '?'.
- Each query is made of 2 sequences of letters and digits, separated by a equal sign '='. A sequence is one or more characters.

```
?acu=good?ing1=bad
?Tes7=K0k0
?a=b
```

This must only be done via a `grep` script. This exercise will make you love regexes!

Consider that the way you execute your scripts will always be similar to the following example, and will always be valid. We will test your script with the option `-E` that enables *extended regex*.

1.3 Example

```
42sh$ cat -e test.txt
https://example.org/absolute/URI/with/absolute/path/to/resource.txt$
https://example.org/absolute/URI/with/absolute/path/to/resource$
http://example.org/test.truc/lol_bob.vlc?good=true?awesome=verytrue$
ftp://localhost:1337?acu=false?ing1=true$
mftp://localhost$
mftp://localhost:1337$
mftp://localhost:1337/test.txt$
mftp://localhost:1337/test$
myscheme7:$
file:/test/truc.txt$
file:/../../test/./truc.txt$
file:/./../test/./truc.txt$
file/./../test/./truc.txt$
/test.txt$
http://hello.txt?why.not>nope$
ftp://local:nope/test.txt/ah?good=false$
42sh$ grep -Ef uri.grep test.txt
```

(continues on next page)

(continued from previous page)

```
https://example.org/absolute/URI/with/absolute/path/to/resource.txt
https://example.org/absolute/URI/with/absolute/path/to/resource
http://example.org/test.truc/lol_bob.vlc?good=true?awesome=verytrue
ftp://localhost:1337?acu=false?ing1=true
mftp://localhost
mftp://localhost:1337
mftp://localhost:1337/test.txt
mftp://localhost:1337/test
myscheme7:
file:/test/truc.txt
file:/../../test/./truc.txt
file:/./../test/./truc.txt
```

It is my job to make sure you do yours.