

# Universidade Federal de Minas Gerais

## Redes de computadores

### Trabalho Prático 1 – Codificação de URLs



**Integrantes:** Matheus Ferreira Marques

**Professores:** José Marcos S. Nogueira, Daniel Fernandes Macedo

**Data:** 15/09/2014

# 1 Introdução

No mundo da computação é muito comum termos o problema de padronização. Seja ela para protocolos, algoritmos, ou até mesmo na escrita de códigos, pois nós somos seres humanos e nem todos pensam da mesma maneira.

Para esse Trabalho Prático vamos estudar e entender o porque de utilizar codificação e decodificação de URLs e implementar o algoritmo usado hoje.

Quando estamos utilizando a Internet precisamos de acessar páginas na web. Cada página recebe um endereço e cada endereço é batizado com um nome. Nasceu então a URL (Uniform Resource Locator). Elas são um padrão para cada endereço presente na web. A especificação completa de uma URL é:

*scheme://domain:port/path?query\_string#fragment\_id*

**Protocolo(scheme):** protocolo que definirá os padrões de sua URL. Dois muito conhecidos são o http e o ftp.

**Domínio(domain):** endereço da pagina web para acesso, podendo ser um nome ou endereço IP.

**Porta(port):** porta de seu endereço. Quando não especificado usa-se a porta padrão do protocolo. Para o http, usa-se a porta 80, por exemplo.

**Caminho(path):** caminho do arquivo destino. O arquivo pode ser uma pagina HTML, um link para download, dentre outros.

**Tabela(query\_string):** Em páginas na web que possuem essa parte é porque criamos alguma tabela de consulta. Uma lista de reprodução no Youtube é um bom exemplo.

**Índice(fragment\_id):** Já que temos uma tabela criada, podemos querer saber em qual índice, dos arquivos presente nessa tabela, nós estamos.

Note que alguns caracteres não foram explícitos. São eles: ':', '/', '?', '#'. Esses caracteres são especiais e reservados para o uso das URLs, sendo que cada um possui um objetivo. Um exemplo é o conjunto “://”. Ele tem como objetivo separar o protocolo do domínio. Em nenhuma outra parte da URL você encontrará esse conjunto. Visto que estamos padronizando as coisas e começando a classificar caracteres como especiais(reservados) ou não, precisamos de criar um algoritmo para que codifique todos os caracteres especiais e não especiais de uma maneira padrão. Isso, pois nem todos os teclados do mundo são únicos – os teclados americanos não possuem o 'ç', por exemplo – e as URLs poderiam ter problemas para serem acessadas de país para país se não for padrão para o mundo inteiro.

Foi necessário, então, a criação de um algoritmo chamado *Percent-Encoding* ou *URL Encoding*.

## 1.1 Percent-Encoding

O algoritmo é bastante simples e atende o problema ser solucionado: padronização de URLs para o mundo todo.

Foi criado uma estratégia para diferenciar caracteres especiais e não especiais para a construção de URLs.

RFC 3986 [section 2.2 Reserved Characters](#) (January 2005)

!	*	'	(	)	;	:	@	&	=	+	\$	,	/	?	#	[	]
---	---	---	---	---	---	---	---	---	---	---	----	---	---	---	---	---	---

RFC 3986 [section 2.3 Unreserved Characters](#) (January 2005)

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
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
0	1	2	3	4	5	6	7	8	9	-	_	.	~												

Figura 1. Tabela de caracteres reservados e não reservados.

## Algoritmo do *Percent-Encoding* para Codificação

A URL presente no arquivo de entrada é escrita em uma *string* e um iterador percorre caractere por caractere procurando por um que seja especial(reservado) de acordo com a Figura 1. Caso encontre algum, o caractere é substituído pelo símbolo '%' e o próprio caractere, porém no formato hexadecimal padrão ANSI. Forma-se então uma nova *string*, agora codificada no formato *percent-encoding*.

Exemplificando: caso o iterador encontre um ':', ele irá substituir esse caractere por "%3A".

## Algoritmo do *Percent-Encoding* para Decodificação

A URL presente no arquivo de entrada, agora codificada, é escrita em uma *string* e um iterados percorre caractere por caractere procurando por um que seja igual a '%'. Caso encontre algum, os próximos dois caracteres são, na verdade, o valor em hexadecimal padrão ANSI do caractere que havia sido codificado. Esse valor em hexadecimal é decodificado e substituído pelo caractere original para que a página na web possa ser acessada com a URL final montada.

Exemplificando: caso o iterador encontre um '%', ele irá substituir esse símbolo e os próximos dois caracteres pelo seu caractere que fora codificado.

Reserved characters after percent-encoding

!	#	\$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

Figura 2. Códigos equivalentes dos caracteres reservados.

## 2 Implementação

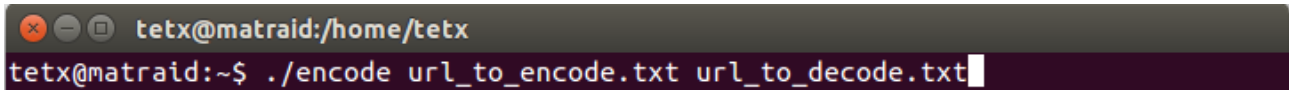
Para colocarmos o algoritmo *Percent-Encoding* “no papel” utilizamos dois códigos fonte: encode.c e decode.c. Cada um tem a sua função explícita de decodificar e codificar uma URL.

### 2.1 Códigos Fonte

#### Encode.c

Como o próprio nome já diz, esse código ficará encarregado de codificar uma URL. Cada URL a ser codificada estará num arquivo texto com seu nome de preferência – desde que seja escrito da mesma forma na linha de comando via terminal – separados por uma única quebra de linha. A saída do programa será outro arquivo texto com seu nome de preferência com todas as URLs codificadas. O formato do arquivo de saída também é uma URL por linha.

Para executar o código via terminal, devemos digitar esse comando mostrado na figura 3:

A terminal window with a dark background. The prompt is 'tetx@matraid:/home/tetx'. The command entered is './encode url\_to\_encode.txt url\_to\_decode.txt'.

```
tetx@matraid:/home/tetx
tetx@matraid:~$ ./encode url_to_encode.txt url_to_decode.txt
```

Figura 3. Exemplo de comando para execução do programa *encode*.

O nome do executável deverá ser, obrigatoriamente, *encode* e os próximos parâmetros com os nomes dos arquivos de entrada e saída no formato texto (.txt).

O arquivo foi escrito na linguagem C/C++ e consiste de duas funções: *url\_encode* e *main*.

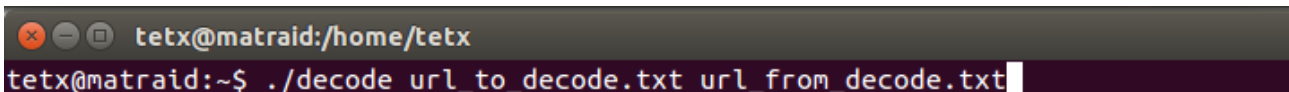
A função *url\_encode* tem o papel de codificar uma URL nos padrões do algoritmo *Percent-Encoding*.

A função *main* tem o papel de executar todo o programa, adquirindo as URLs do arquivo de entrada, codificando-as através da função *url\_encode* e escrevendo no arquivo de saída, atendendo todos os padrões.

## Decode.c

Como o próprio nome já diz, esse código ficará encarregado de decodificar uma URL. Cada URL a ser decodificada estará num arquivo texto com seu nome de preferência – desde que seja escrito da mesma forma na linha de comando via terminal – separados por uma única quebra de linha. A saída do programa será outro arquivo texto com seu nome de preferência com todas as URLs decodificadas. O formato do arquivo de saída também é uma URL por linha.

Para executar o código via terminal, devemos digitar esse comando mostrado na figura 4:

A terminal window with a dark background. The prompt is 'tetx@matraid:/home/tetx'. The command entered is './decode url\_to\_decode.txt url\_from\_decode.txt'.

```
tetx@matraid:/home/tetx
tetx@matraid:~$ ./decode url_to_decode.txt url_from_decode.txt
```

Figura 4. Exemplo de comando para execução do programa *decode*.

O nome do executável deverá ser, obrigatoriamente, *decode* e os próximos parâmetros com os nomes dos arquivos de entrada e saída no formato texto (.txt).

O arquivo foi escrito na linguagem C/C++ e consiste de duas funções: *url\_decode* e *main*.

A função *url\_decode* tem o papel de decodificar uma URL nos padrões do algoritmo *Percent-Encoding*.

A função *main* tem o papel de executar todo o programa, adquirindo as URLs do arquivo de entrada, decodificando-as através da função *url\_decode* e escrevendo no arquivo de saída, atendendo todos os padrões.

## Makefile

Código responsável por compilar todo o projeto e gerar os executáveis *encode* e *decode*. Os nomes deverão ser exclusivamente esses.

Para executar esse código via terminal, devemos digitar esse comando mostrado na figura 5:



```
tetx@matraid:/home/tetx
tetx@matraid:~$ make
```

Figura 5. Exemplo de comando para execução do código Makefile.

O comando make facilita bastante a compilação de vários códigos para o funcionamento de um único programa ou vários códigos que são parte de vários programas.

## 2.2 Informações Técnicas

Todo o Trabalho Prático foi desenvolvido no mesmo computador e escrito no editor de texto Sublime.

Sistema Operacional: Ubuntu 14.04 LTS 32-bits.

Processador: Intel Core i7-3612QM CPU @ 2.10GHz x 8

Gráficos: Intel Ivybridge Mobile x86/MMX/SSE2

Memória: 8GB

IDE utilizada: Nenhuma

Editor de texto: Sublime Text 3

## 3 Resultados

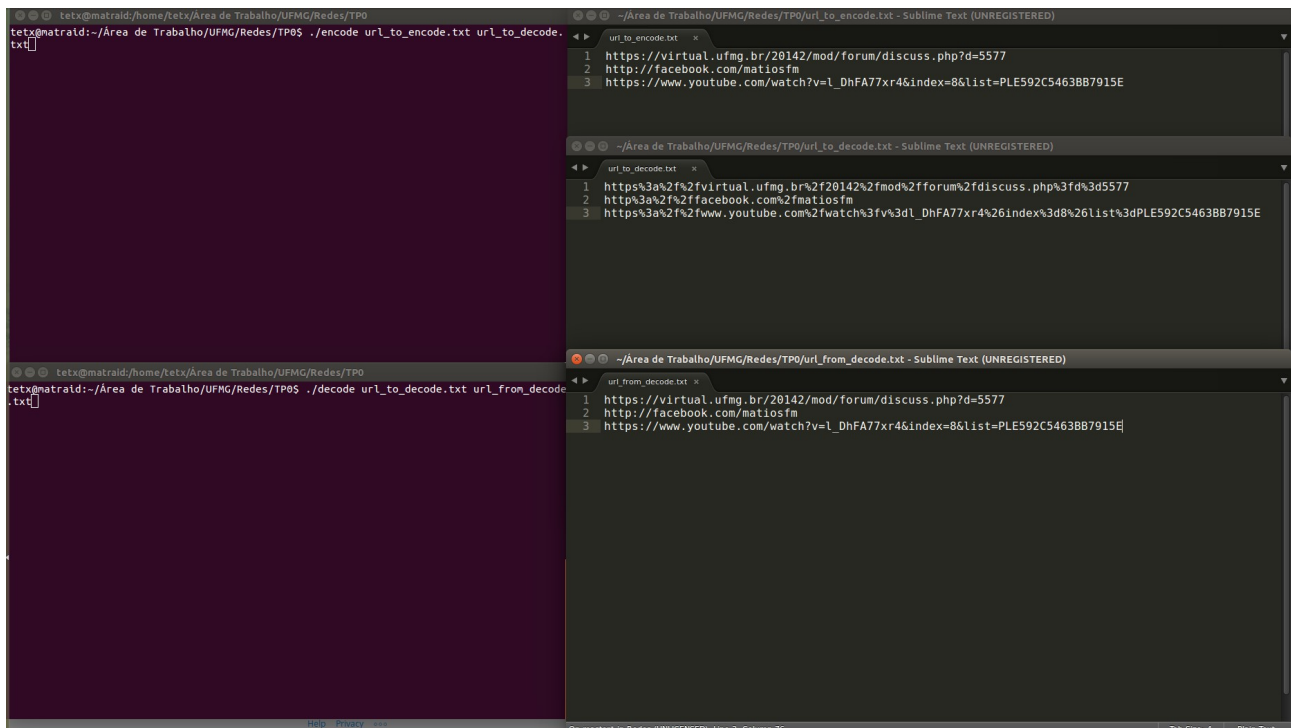


Figura 6. Tela dos resultados com a ajuda do Sublime Text 3.

Após a execução de encode e decode obtivemos esses três arquivos. O encode utilizou dos arquivos 1 e 2, sendo que o primeiro foi escrito previamente com alguns links comumente vistos na internet. O arquivo decode utilizou dos arquivos 2 e 3. O objetivo era os arquivos 1 e 3 estarem exatamente idênticos mostrando que o decode decodificou de maneira correta a codificação feita pelo encode.

## 4 Conclusão

O Trabalho Prático proporcionou uma experiência interessante mostrando como as URLs são montadas por trás de todo navegador web. Se não fossem os padrões da URL e do algoritmo de codificação/decodificação do *percent-encoding* seria difícil compartilhar links através da Internet, já que alguns computadores poderiam não entender todo o link.

O Trabalho ficou pequeno e escrito da forma mais simples e bem comentada em todos os códigos fonte. Isso só mostra que o algoritmo *percent-encoding* é bastante simples de implementar e muito eficaz.

Apesar do Trabalho ter sido escrito em C/C++ não foi utilizado nenhuma classe para não aumentar o grau de complexidade do código como um todo.

Poderia ter ficado mais modularizado e com um código mais legível, como são todos os códigos em C++ comparados com o C, mas conclui que seria melhor manter apenas dois códigos fontes, já que não foi necessário escrever muitas linhas de códigos para atingir o objetivo.

## 5 Bibliografia

1. <http://www.ietf.org/rfc/rfc3986.txt>
2. <http://en.wikipedia.org/wiki/Percent-encoding>
3. <http://tools.ietf.org/html/rfc1738>
4. [http://en.wikipedia.org/wiki/Uniform\\_resource\\_locator](http://en.wikipedia.org/wiki/Uniform_resource_locator)
5. PETERSON, L. L. & DAVIE, B. S. “Computer Networks: A Systems Approach”, Morgan Kaufman, San Francisco, CA, Fifth Edition, 2012. ISBN.