DCC031 – Redes de computadores

Trabalho prático 0 – Codificação de URLs

Professores: José Marcos S. Nogueira, Daniel Fernandes Macedo

Data de entrega: 15/09/2014

Trabalho em grupos de dois alunos (ou individual).

Valor do trabalho: 10 pontos

1. Introdução

Na Web, muitas vezes temos formulários onde enviamos um texto para o servidor. Uma forma de enviar o texto é codificando-o no endereço da solicitação HTTP, o URL (Uniform Resource Locator). O URL é uma string que indica como um programa pode acessar um certo recurso, e possui: *(i)* o protocolo a ser empregado; *(ii)* um identificador do computador a ser acessado; *(iii)* a porta, um atributo que nem sempre é indicado; *(iii)* o caminho para acessar o recurso; e *(iv)* dados para estabelecer a comunicação, tais como usuário e senha.

Para identificar cada um destes campos, empregamos caracteres ou sequências de caracteres delimitadores. Por exemplo, a seguinte URL é válida para acessar dados em um servidor de arquivos baseado em AFS (protocolo de compartilhamento de arquivos do MacOS):

afs://redes:senha@servidor.dcc.ufmg.br:/dir1/dir2/arquivo.bin

Neste exemplo, `afs` indica o protocolo, `redes:senha` indica o login e a senha. O caractere arroba separa as credenciais do endereço do servidor (`servidor.dcc.ufmg.br`), e finalmente o caminho do arquivo é dado por `/dir1/dir2/arquivo.bin`.

Um aspecto crucial em qualquer sistema de codificação é a existência de caracteres reservados. Por exemplo, os caracteres ":" e "/" são reservados para separar campos com funções diferentes no URL. Entretanto, surge a seguinte situação: e se a minha senha utilizar estes caracteres reservados?

Neste trabalho vocês vão implementar uma possível forma de solucionar este problema, que é a forma definida na regra de codificação de URLs da Internet.

2. A codificação Percent-encoding

A Internet resolve o problema da codificação de *strings* utilizando um sistema conhecido como "*Percent encoding*" ou "*URL Encoding*". Este sistema permite a codificação de dados em ASCII, bem como dados binários, de forma que os

mesmos podem ser escritos em uma URL. A definição completa do sistema é dada na RFC 3986, entretanto uma síntese da codificação pode ser encontrada na página http://en.wikipedia.org/wiki/Percent-encoding.

O trabalho prático não irá implementar o padrão completo. Vocês deverão suportar somente a codificação e decodificação de caracteres ASCII.

3. Programas a serem desenvolvidos

O trabalho prático consistirá na implementação de dois programas, o codificador e o decodificador. Ambos vão receber dois parâmetros de entrada, como mostrado a seguir:

```
encode <arquivo_entrada> <arquivo_saida>

decode <arquivo_entrada> <arquivo_saida>
```

O primeiro programa, o encode, irá ler um arquivo de texto, e gravar o conteúdo deste arquivo, em forma codificada, no arquivo de saída. Se o arquivo de saída já existir, ele deverá ser sobrescrito.

Já o decode irá ler o conteúdo de um arquivo em forma codificada, e escrever o texto correspondente, já decodificado, no arquivo de saída. Se o arquivo de saída já existir, ele deverá ser sobrescrito.

4. Entrega do código

O código e a documentação devem ser entregues em um arquivo Zip (não pode ser RAR nem .tgz nem .tar.gz) no Moodle, contendo um arquivo **readme.txt** com o nome dos integrantes, e um arquivo PDF da documentação. Incluam todos os arquivos (.c, .h, makefile, não incluam executáveis ou arquivos objeto) EM UM ÚNICO DIRETÓRIO. Um makefile deve ser fornecido para a compilação do código.

Parte desse trabalho envolve o aprendizado de como construir um makefile e utilizar a ferramenta Make. Este makefile, quando executado sem parâmetros, irá gerar os dois programas, *encode* e *decode*, EXATAMENTE com esses nomes. Submissões onde os programas não seguem as especificações de parâmetros e nomes, makefiles não funcionando ou arquivos necessários faltando não serão corrigidos. Programas que não compilarem também não serão corrigidos. O julgamento do trabalho será feito em um computador rodando o SO Linux.

5. Documentação

A documentação, além de ser entregue com o Zip junto ao código, deve ser entregue impressa ao professor até o dia seguinte à entrega, na sala de aula. Caso não tenhamos aula no dia seguinte ao prazo para envio, iremos combinar antecipadamente qual é a forma mais conveniente de entrega da documentação. Trabalhos que forem entregues no Moodle mas sem a documentação impressa **não serão** corrigidos.

O texto da documentação deve ser breve, de forma que o corretor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado
- Uma descrição sucinta dos algoritmos e dos tipos abstratos de dados, das principais funções, e procedimentos e as decisões de implementação
- Decisões de implementação que porventura estejam omissos na especificação
- Testes, mostrando que o programa está funcionando de acordo com a especificação, seguidos da sua análise
- Conclusão e referências bibliográficas

6. Avaliação

A avaliação do trabalho será composta pela execução dos programas desenvolvidos e pela análise da documentação. Os seguintes itens serão avaliados:

- A qualidade do código (código bem organizado, com comentários explicativos, variáveis com nomes intuitivos, modularidade, etc).
- Se o código executa as funções esperadas e da forma definida na especificação do trabalho.
- Execução correta do código em entradas de testes, a serem definidas no momento da avaliação. As entradas de teste irão exercitar a funcionalidade completa do código e testar casos especiais ou de maior dificuldade de implementação, mas que devem ser tratados por um programa correto.
- Aderência ao protocolo especificado. Iremos usar ferramentas de captura do tráfego da rede (tcpdump, wireshark) e iremos analisar o código para verificar se a comunicação está implementada da forma definida na documentação.
- Conteúdo da documentação, que deve conter os itens mencionados anteriormente.
- Coerência e coesão da documentação (apresentação visual e organização, uso correto da língua, qualidade textual e facilidade de compreensão).

Como mencionado anteriormente, trabalhos fora da especificação (por exemplo, sem makefile, que não gerem programas com os nomes especificados, que não compilem ou não possuam os parâmetros esperados) não serão corrigidos. Trabalhos fora da especificação tomam muito trabalho do corretor, que deve

entender como compilar e rodar **cada programa avaliado**, tempo esse que deveria ser empregado para avaliar a execução e corretude do código.

Qualquer elemento que não esteja especificado neste documento, mas que tenha que ser inserido para que o protocolo funcione, deve ser descrito na documentação de forma explícita.

7. Desconto de nota por atraso

Os trabalhos poderão ser entregues até a meia-noite do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema Moodle, ou seja, a partir de 00:01 do dia seguinte à entrega no relógio do Moodle, os trabalhos já estarão sujeitos a penalidades. O valor do trabalho irá decrementar 5% a cada hora de atraso. Para um aluno que entregou o trabalho à 1:38, ou seja, com 1:38 de atraso, iremos descontar 10%, empregando a fórmula a seguir:

```
Nota = valor_correção * (1-0.05*horas_atraso)
```

8. Referências

Programação em C:

- http://www.dcc.ufla.br/~giacomin/Textos/tutorialc.pdf
- ftp://ftp.unicamp.br/pub/apoio/treinamentos/linguagens/c.pdf
- http://www.cs.cf.ac.uk/Dave/C/CE.html
- http://www2.its.strath.ac.uk/courses/c/
- http://www.lysator.liu.se/c/bwk-tutor.html

Uso do programa make e escrita de Makefiles:

- http://comp.ist.utl.pt/ec-aed/PDFs/make.pdf
- http://haxent.com.br/people/ruda/make.html
- http://informatica.hsw.uol.com.br/programacao-em-c16.htm
- http://www.gnu.org/software/make/manual/make.html
- http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/