Universidade Federal de Pernambuco
Centro de Informática

IF767 - Processamento de Cadeias de Caracteres 2014.2
Prof. Paulo Fonseca

Projeto 1 — versão 2 (19/11/2014)

- Este documento contém as regras e diretrizes para o primeiro projeto. Leia com atenção todo o conteúdo do documento e tente ater-se às orientações o mais fielmente possível.
- As regras abaixo podem ser modificadas a qualquer tempo pelo professor no melhor interesse acadêmico e didático. As modificações serão comunicadas em tempo útil através do grupo de discussão da disciplina.
- Eventuais omissões serão tratadas de maneira discricionária pelo professor, levando-se em conta o bom senso, a praxe acadêmica e os interesses didáticos.

Objetivo

Neste projeto deve ser desenvolvida uma ferramenta para busca de padrões num arquivo ou conjunto de arquivos, similar ao GNU grep [1]. O objetivo é de consolidar o conhecimento dos algoritmos vistos no curso através da implementação de um software com correção, documentação e escalabilidade em nível de produção.

No que se segue, chamaremos a ferramenta de **`pmt`**.

Funcionamento básico

A ferramenta deve ter uma interface em linha de comando (*command line interface*—*CLI*) seguindo as diretrizes GNU/POSIX [2]. A sintaxe básica deve ser

`$ pmt [options] pattern textfile [textfile...]`

que fará com que o padrão *`pattern`* seja procurado no(s) arquivo(s) *`textfile`*. Múltiplos arquivos de texto podem ser indicados utilizando-se *wildcards* (e.g. `livro*.txt`).

A ferramenta deve suportar dois modos:

- Busca exata (default)
- Busca aproximada.

A busca aproximada deve ser indicada pela opção:

`-e, --edit` e_{max} **:** Localiza todas as ocorrências aproximadas do padrão a uma distância de edição máxima e_{max}

A ferramenta também poderá receber um conjunto de padrões a serem procurados num arquivo, sendo um padrão por linha, o que deve ser feito através da opção

[1] `http://www.gnu.org/software/grep/`
[2] `https://www.gnu.org/prep/standards/html_node/Command_002dLine-Interfaces.html`

`-p, --pattern patternfile:` Realiza a busca de todos os padrões contidos no arquivo *patternfile*.

Formato de saída

A ferramenta deve imprimir na saída padrão, para cada arquivo pesquisado, as linhas do arquivo do texto contendo as ocorrências de todos os padrões pesquisados, de maneira similar ao GNU grep.

```
$ pmt "Brasil"  hino*.txt
hinopt1.txt:Brasil, um sonho intenso, um raio vivido
hinopt1.txt:Entre outras mil Es tu, Brasil.
hinopt1.txt:Brasil!
hinopt2.txt:Fulguras, o Brasil, florao da America,
hinopt2.txt:Brasil, de amor eterno seja simbolo
hinopt2.txt:Entre outras mil Es tu, Brasil.
hinopt2.txt:Brasil!
```

Além dessa saída por default, também de forma similar ao GNU grep, a ferramenta deve suportar uma opção

`-c, --count` Realiza a busca por todos os padrões em todos os arquivos indicados e imprime na saída padrão apenas a quantidade total de ocorrências encontradas por arquivo.

```
$ pmt -c "Brasil"  hino*.txt
hinopt1.txt:3
hinopt2.txt:4
```

Implementação

A ferramenta deve ser implementada preferencialmente em C/C++. O objetivo é torná-la a mais eficiente possível. A ferramenta deve ser baseda na plataforma GNU/Linux. Deve-se tentar minimizar as depedências externas para torná-la facilmente portável entre plataformas.

Podem ser utilizadas APIs externas apenas para o *frontend* da ferramenta. Por exemplo, a GNU C Library (glibc)[3] contém funções para o parsing das opções de linha de comando (getopt), e para os nomes de arquivos com wildcards (fnmatch). Entretanto, o *backend* da ferramenta deve consistir *apenas de algoritmos vistos em aula* e (re-)implementados diretamente pelos alunos. *A detecção de cópia de partes substanciais do código desses algoritmos implicará na atribuição da nota 0.0 (zero) ao trabalho como um todo, independente de outras partes*.

Deverão ser implementados *pelo menos* dois algoritmos: um para a busca exata e outro para a busca aproximada.

Deliverables

Deve ser entregue uma tarball com nome no fomato *toolname.tgz*, onde *toolname* representa o nome da ferramenta (e.g. `pmt.tgz`). Este arquivo deve conter um diretório com o seguinte conteúdo *mínimo*

[3] `http://www.gnu.org/software/libc/`

(trocando-se, obviamente, `pmt` pelo nome escolhido).

```
pmt/
 |
 +-- doc/
 +-- src/
 +-- README.txt
```

O arquivo `README.txt` deve conter uma identificação da ferramenta, dos autores, e as instruções para compilação (vide seção abaixo). O conteúdo de cada diretório será especificado a seguir.

Código-fonte

Deve ser entegue o código fonte da ferramenta juntamente com um Makefile ou script para compilação no subdiretório `src/`. As instruções para o processo de compilação da ferramenta devem ser dadas no arquivo `README.txt`. Idealmente a compilação deveria consistir apenas na execução de um simples `make`.

O código deve ser o mais *limpo*[4] possível. Entretanto, os objetivos principais são 1) correção e 2) eficiência. Portanto, deve-se evitar o uso exagerado de modelagem por objetos, padrões de projetos, etc. que tornem o programa mais lento. Um programa bem estruturado, com nomes expressivos para funções e variáveis, e com uma separação clara entre interface e motor de busca, deve ser suficiente.

Após a compilação, o arquivo executável deve estar num diretório `bin`, criado dentro do diretório original, isto é, teremos

```
pmt/
 |
 +-- bin/    <=== executável aqui
 +-- doc/
(...)
```

Documentação

Conforme as diretrizes adotadas para a CLI, uma ajuda com as instruções para a utilização básica da ferramenta deve ser obtida através da execução da ferramenta com a opção

`-h, --help`

Além disso, deverá ser entregue um breve relatório ($\leq$ 4 págs) contendo:

- Nome da ferramenta
- Identificação da equipe
- Descrição do funcionamento da ferramenta, incuindo:

[4] *RC Martin*. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.

    - Algoritmos implementados
    - Situações nas quais cada algoritmo é empregado
- Detalhes de implementação relevantes, com impacto significativo para o desempenho da ferramenta, incluindo:
    - Estruturas de dados
    - Estratégia de leitura das entradas
    - Heurísticas para combinação do algoritmos
    - etc.
- Bugs conhecidos e limitações de desempenho notáveis. Se o trabalho não foi integralmente concluído, o que faltou deve ser explicitamente reportado aqui.

Esse relatório deve estar contido no subdiretório `doc/`, num arquivo chamado *toolname.ext*, onde *toolname* corresponde ao nome da ferramenta, e *ext* à extensão do formato do arquivo, que pode ser um `.pdf` ou um simples `.txt` (*Não* use MSWord ou qualquer formato proprietário).

Avaliação

A avaliação será feita com base nos seguintes critérios:

1. Implementação (peso 2). Inclui a correção e qualidade do código-fonte levando-se em conta a quantidade e dificuldade intrínseca dos algoritmos implementados.
2. Qualidade da documentação (peso 2). Inclui o relatório, o `README.txt` e a ajuda do programa.
3. Eficiência prática do programa (peso 3). As ferramentas serão submetidas a testes automáticos e os tempos de execução em diferentes entradas serão aferidos. Serão utilizadas entradas grandes, potencialmente capazes de exceder a capacidade de memória da máquina. É preciso atenção quanto à leitura dos arquivos.

   Os tempos dos experimentos serão medidos em vários cenários de busca exata ou aproximada de um ou vários padrões com diferentes limites de erro. Para cada caso, serão então calculadas a média μ e o desvio padrão σ dos tempos das ferramentas. As ferramentas com tempo $t \leq \mu + \sigma$ receberão a pontuação integral nesse teste. A partir desse limiar, a pontuação sofrerá uma penalização de $(1 + e^{-4(\delta - 1.5)})^{-1} \cdot 100\%$, onde $\delta := (t - \mu)/\sigma$.
4. Arguição (peso 3). Será agendada, posteriormente, uma breve arguição de cerca de 15min com cada equipe. Cada integrante deve ter participado de todas as atividades e, portanto, deve conhecer integralmente ser capaz de responder questões sobre qualquer aspecto do projeto.

Extras

Além desse conjunto mínimo de requisitos, cada equipe está livre para implementar recursos extras. Esses recursos devem ser assinalados no relatório e poderão receber, eventualmente, alguma bonificação.

Equipes

O projeto deve ser feito em duplas. Cada integrante deve participar de todas as atividades envolvidas (implementação, documentação e testes).

Data de entrega

O tarball deve ser enviado por e-mail até **30 de Novembro de 2014**.

Changelog

v.2

- Especificação do formato de saída
- Novo prazo de entrega