Universidade Federal de Ouro Preto - UFOP
Disciplina de Programação de Computadores I - CSI030
Trabalho Prático: Gerenciamento de Listas de Tarefas
Professor: Vinicius Dias
Valor: 1.0 ponto (10% da nota final)

Leia atentamente a especificação a seguir:

Visão geral

Neste trabalho, vocês devem implementar um gerenciador de listas de tarefas (conhecidas como *TODO lists*). A ideia é bem simples: uma **lista de tarefas** é uma lista ordenada de coisas para se fazer: **as tarefas**. Tarefas podem ser adicionadas, excluídas, marcadas como feita ou marcadas como não-feitas. Listas de tarefas podem ser adicionadas, excluídas ou visualizadas. Uma lista de tarefa também possui um nome que a identifica. Por exemplo, uma lista de tarefas com nome "Compras" e outra lista de tarefas com o nome "TCC" podem facilitar a ida a um supermercado ou mesmo tornar a sua vida acadêmica mais organizada.

Descrição

Inicialmente, vamos definir o que compõe cada elemento deste trabalho.

- Lista de tarefas: deve conter um **nome**, uma **descrição** e **tarefas**.
- Tarefa: deve conter uma **descrição**, uma **data** e um **estado**. O estado da tarefa deve indicar se a mesma se encontra feita ou não-feita.

O sistema de gerenciamento de listas de tarefas deve suportar as seguintes funcionalidades, organizadas em dois níveis hierárquicos:

1. Adicionar Lista de Tarefa
O usuário é capaz de cadastrar uma nova lista de tarefas vazia. Nesse caso, o usuário entra com as informações de uma lista (nome e descrição) e o sistema cria uma lista vazia.

2. Remover Lista de Tarefa
O usuário é capaz de remover uma lista de tarefas existente. Nesse caso, o sistema mostra uma lista das listas cadastradas e o usuário escolhe qual delas deseja remover.

3. Abrir Lista de Tarefa
O usuário é capaz de abrir uma lista de tarefas existente para visualização e edição. Nesse caso, o sistema mostra uma lista das listas cadastradas e o usuário escolhe qual delas deseja visualizar/editar. Em seguida, o sistema mostra o conteúdo da lista de tarefas (nome, descrição e tarefas) e fornece as seguintes funcionalidades para o usuário.

> ***1. Adicionar Tarefa***
> O usuário é capaz de adicionar uma tarefa à lista de tarefas. Nesse caso, o sistema pergunta os campos da tarefa ao usuário (descrição e estado) e a nova tarefa é criada com o estado de não-feita. Além disso, a nova tarefa é adicionada com a prioridade mais baixa até então dentre as tarefas existentes na lista.
>
> ***2. Remover Tarefa***
> O usuário é capaz de remover uma tarefa da lista de tarefas. Nesse caso, remover uma tarefa implica em reorganizar a lista de tarefas. A decisão de como fazer isso é parte de seu trabalho.

3. Marcar Tarefa como Feita
O usuário é capaz de marcar uma tarefa como feita.

4. Marcar Tarefa como Não-Feita
O usuário é capaz de marcar uma tarefa como não-feita.

4. Mudar Prioridade do Item
O usuário é capaz de alterar a prioridade (posição) de uma tarefa.

5. Voltar ao Início
O usuário é capaz de voltar às funcionalidades da hierarquia anterior.

4. Listar Tarefas de um Período
O usuário é capaz de visualizar as tarefas existentes dado um período: dia/mês/ano de início e dia/mês/ano de término. Nesse caso, o sistema deve buscar as **tarefas que se enquadram nesse período de todas as listas de tarefas** e exibir o resultado para o usuário.

Detalhes de funcionamento

O usuário interage com o sistema na forma de menus de opções, seguindo a hierarquia apresentada na seção anterior. Você pode representar a escolha do usuário por uma sequência de números. Assim, considerando o menu inicial:

```
1 - Adicionar Lista de Tarefas
2 - Remover Lista de Tarefas
3 - Abrir Lista de Tarefas
4 - Listar Tarefas de um Período
```

ao pressionar '1', o sistema ativa a inclusão de uma lista de tarefas que espera a entrada por parte do usuário dos respectivos nome e descrição da lista. Da mesma forma, ao pressionar '3', o sistema mostra uma lista de listas existentes que podem ser escolhidas e gerenciadas através do novo menu de funcionalidades para tarefas abertas, construído de forma similar.

A prioridade das tarefas também pode ser representada por números. Assim como descrito anteriormente, as tarefas são adicionadas com a prioridade mais baixa até então na lista. Entretanto, como a prioridade das tarefas podem mudar ou podem ser excluídas, a ordem de aparição delas também deve ser alterada de acordo.

Detalhes técnicos

Seu sistema será testado em ambiente Linux, com linguagem de terminal Bash. A compilação deve ser através de um Makefile:

```
$ make
```

Makefiles são recursos padrão de compilação de programas em linguagem C/C++, facilitando o processo de construção de projetos complexos (ferramenta de *build*). Em ambientes baseados em UNIX, Makefiles são especialmente conhecidos como meios de se construir um software/pacote a partir do código-fonte e de forma manual. O funcionamento de um Makefile[1] é baseado em objetivos (targets) e dependências. Por exemplo, para se obter o executável de um programa em C (objetivo), precisamos do arquivo fonte escrito em

[1] https://www.cs.swarthmore.edu/~newhall/unixhelp/howto_makefiles.html

linguagem C (dependência). Por sua vez, para obter o executável a partir do código-fonte, basta utilizarmos um compilador da linguagem (por exemplo, o GCC).

Em seguida, o seu sistema deve ser iniciado com o seguinte comando:

```
$ ./todolist
```

Observe que o nome do executável **deve ser "todolist"**. Lembre-se de que a adequação à especificação faz parte da avaliação do trabalho.

Pontos extras

Receberão pontos extras os alunos que:

1. Persistirem as informações do gerenciador em armazenamento secundário (arquivos). A biblioteca padrão de C inclui ferramentas para manipulação correta de arquivos[2].

O que deve ser entregue

1. Implementação:
    a. Códigos-fonte em linguagem C devidamente indentados e comentados.
    b. *Makefile* que compila seu sistema.
    c. Tipos de arquivos aceitos: código-fonte C (.c) e *Makefile,* apenas.
2. Documentação:
    a. **Introdução.** Descreva o problema que você está resolvendo
    b. **Implementação.** Descreva sua solução: quais estruturas decidiu usar, como elas são usadas para implementar as funcionalidades exigidas e quaisquer suposições que você usou e que julgue pertinente para o entendimento da sua solução
    c. **Manual de utilização.** Descreva os passos e comandos (em Linux) necessários para compilar, executar e interagir com o seu software.
    d. **Estudo de caso.** Elabore um exemplo (walkthrough) que motive e justifique a utilização de seu software.
    e. **Considerações finais.** Realize uma reflexão sobre as dificuldades e sobre os resultados que você encontrou/descobriu ao longo do caminho.
    f. Tipo de arquivo aceito: PDF apenas.

Como deve ser entregue

Os trabalhos devem ser entregues pelo Moodle (moodlepresencial.ufop.br) na forma de um único arquivo zipado (formato `.zip`) contendo implementação e documentação, organizadas em um diretório chamado "todolist":

```
todolist/
  |-todolist.c          // código-fonte (pode conter mais de um arquivo .c)
  |-Makefile            // makefile que compila seu sistema
  |-documentacao.pdf    // documentação em PDF
```

Você deve compactar ("zipar") esse diretório e seu conteúdo em um arquivo único chamado "**`todolist-<seu-nome>-<seu-sobrenome>.zip`**". Arquivos em outros formatos (RAR, por exemplo) que não `.zip` não serão aceitos.

Avaliação

Este trabalho será avaliado de acordo com os seguintes critérios:

---

[2] http://fig.if.usp.br/~esdobay/c/arquivos.pdf

- O aluno seguiu a especificação do trabalho;
- Qualidade da documentação;
- Qualidade da implementação;
- (Ponto extra) O aluno implementou a funcionalidade de persistência em arquivos.

Observações importantes

- Leia esta especificação com cuidado;
- Comece o trabalho o quanto antes;
- Tire dúvidas com o professor;
- O trabalho deve ser implementado em linguagem C. Trabalhos em outras linguagens (C++, Java, Python, etc.) não serão aceitos;
- Siga **exatamente as especificações** de formato de entrega, compilação e execução descritas neste documento;
- Apenas um arquivo deve ser entregue, compactado e no formato `.zip`;
- Seu trabalho deve ser compatível com ambiente Linux;
- O trabalho é individual. Cópias detectadas serão avaliadas com nota zero;
- Trabalhos atrasados não serão aceitos.