PUCRS/Escola Politécnica
Programação de Software Básico
Trabalho 1

Compreensão de software básico escrito em linguagem C: estudo de caso com um utilitário ou biblioteca

Cronograma:

- Escolha de grupos e programas: 03/04
- Entrega: 24/04

Objetivo:

Conhecer e apresentar idiomas e práticas da programação em **linguagem C** no contexto de **software básico**.
Atenção: não são aceitos trabalhos sobre programas escritos em linguagem C++, C\#, Objective-C ou similares.

Enunciado:

Software Básico ou Software de Sistema é uma categoria de programas que permitem o uso de equipamentos e oferecem apoio a outros programas. Sistemas Operacionais e Compiladores são exemplos de software básico.
Para iniciar o estudo de software básico, faremos um estudo de utilitários e bibliotecas de programação escritas em linguagem C. Serão utilizados projetos do GNU [1][2] e bibliotecas da própria linguagem C [3]. Utilitários e bibliotecas nem sempre são considerados software básico pela literatura. Utilitários apresentam interface com usuário e bibliotecas nem sempre oferecem funções de acesso específico ao hardware. Por outro lado, os utilitários e as bibliotecas deste estudo foram criados por programadores profissionais que atuam em software básico.
Para desenvolver software básico, é necessário compreender e adotar práticas que nem sempre são utilizadas no desenvolvimento de programas em outras categorias. Não é suficiente conhecer a linguagem de programação C, mas como ela pode ser utilizada pelos programadores profissionais.

Um idioma em uma linguagem de programação é uma maneira particular e recorrente de usar essa linguagem, incorporando padrões, práticas e convenções que são característicos da comunidade de programadores. No contexto da linguagem C, programadores experientes frequentemente empregam técnicas e recursos que, em outras linguagens, podem ser vistos como arriscados ou até mesmo desaconselháveis. Entre esses recursos, destacam-se: (i) **aritmética de ponteiros**, que permite a manipulação direta de endereços de memória para um controle preciso do desempenho; (ii) **desvio incondicional**, como o uso de `*goto*`, que pode complicar o fluxo do programa, mas é eficiente em situações específicas; (iii) **reutilização de variáveis**, que pode comprometer a legibilidade em troca da economia de memória; (iv) **contagens regressivas**, que, embora menos intuitivas, podem aumentar a eficiência; e (v) **blocos de código similares ou repetidos**, que podem elevar o risco de erros, mas são justificados quando a performance supera

a necessidade de modularidade. Embora essas práticas possam parecer "problemáticas" em outros contextos, elas são escolhas conscientes dos programadores de C para maximizar o desempenho e o controle sobre o software, especialmente em sistemas onde esses fatores são essenciais. Esses programadores possuem um domínio profundo desses recursos e compreendem plenamente o impacto que eles têm na eficiência de execução, mesmo que isso implique um sacrifício na legibilidade.

Neste trabalho, deve ser realizado o estudo, a compreensão e a apresentação de código fonte desenvolvido por programador da linguagem C profissional, no contexto de projetos como o compilador C e os utilitários do Linux.
O trabalho deve ser desenvolvido individualmente ou em duplas. A equipe deve escolher um conjunto de utilitários do GNU e/ou um conjunto de funções da biblioteca padrão da linguagem C, conforme implementada no compilador GNU C.
O trabalho deve cobrir no mínimo 600 linhas de código-fonte, incluindo comentários e linhas em branco, de acordo com o texto original nos repositórios escolhidos. Caso o programa tenha menos de 600 linhas, o grupo deve incluir novos programas ou funções até atingir o limite mínimo indicado.
Cada grupo deve reservar o programa ou função de biblioteca escolhidos. Somente um grupo para cada programa ou função principal. A reserva mais antiga prevalece. A reserva ocorre em área indicada no Moodle. O utilitário *echo* não pode ser reservado.

Entrega

Cada grupo deve realizar uma apresentação de até 10 minutos, com a apresentação dos itens solicitados nos critérios de avaliação indicados a seguir. O grupo deve realizar a gravação e entregar o endereço do vídeo hospedado em um sistema de compartilhamento de mídias (ex.: Zoom e YouTube).

Atenção: <u>todos os integrantes do grupo devem apresentar o vídeo, isto é, áudio e vídeo. Na ausência de imagem e/ou áudio o trabalho não será avaliado.</u>

Critérios de avaliação

<u>Identificar e destacar</u> na apresentação do trabalho:

- [1 ponto] Histórico: no caso de autores identificados no código, descrever a carreira desses programadores, publicações e suas contribuições em outros sistemas. Indicar a instituição e/ou projeto relacionado com o programa, ex. UNIX, Linux etc.

- [1 ponto] Convenções de codificação: convenções como alinhamento de blocos, margem, identificadores.

- [1 ponto] Manipulação de ponteiros e vetores: uso de aritmética de ponteiros.

- [2 pontos] Análise de blocos de código: as responsabilidades de cada bloco e de funções auxiliares.

- [3 pontos] Truques: codificação com uso de recursos característicos da linguagem C que podem reduzir o número de instruções geradas pelo compilador ou executadas pelo programa. Redução de ocupação de memória e redução de ciclos do processador. "Coisas de programador C".

- [1 ponto] Referência acadêmica: ao menos uma obra acadêmica relacionada com o programa ou seus autores.

- [1 ponto] Exemplo de uso do programa: um caso de teste com ilustração da execução do programa, consumo de memória (*stack* e *heap*), com uso de depurador ou diagrama.

- [1 ponto] Diagrama estático e dinâmico da UML ou notação similar.

- [1 ponto] Construção e testes automatizados: uso de utilitários como make, bibliotecas como *Boost* etc.

Referências:

[1] GNU Core Utils: https://github.com/coreutils/coreutils/tree/master/src
[2] GNU Savannah: https://git.savannah.gnu.org/cgit/
[3] GLIBC: https://github.com/lattera/glibc