Aluno: Eric Assis

Resumo da aula 11/03

1. **Git e GitHub**
    - **Por que usar**: controle de versão (histórico de alterações), colaboração em equipe e portfólio público.
    - **Criação de conta**: escolher login, e-mail e senha; confirmar por e-mail e configurar perfil.
    - **Configuração local**: instalar Git, definir user.name e user.email, (opcional) gerar chave SSH para evitar autenticação em cada push.
    - **Fluxo básico**: criar repositório remoto → clonar para máquina local → editar/commitar/commit → push para o repositório no GitHub.

---

2. **Componentes Arquiteturais (Hardware)**
    - **CPU (Processador)**: unidade de controle (decodifica instruções) + ALU (operações lógicas e aritméticas) + registradores.
    - **Memória RAM**: armazenamento volátil de dados e instruções em execução; variáveis locais (stack) e alocações dinâmicas (heap) residem nela.
    - **Armazenamento Permanente (HDD/SSD)**: guarda sistema operacional, arquivos e código-fonte; SSDs usam memória flash, HDDs usam discos magnéticos.
    - **Placa-mãe e Barramentos**: interligam CPU, memória e periféricos (PCIe, SATA, USB); permitem transferência de dados entre componentes.
    - **Dispositivos de E/S**: teclado, mouse, monitores e interfaces de rede; o sistema operacional gerencia drivers que traduzem comandos de hardware para software.

---

3. **Unidades de Informação e Performance**
    - **Bit e Byte**: menor unidade de informação (bit), 8 bits = 1 byte.
    - **KB, MB, GB, TB**: múltiplos de byte (base 2: 1 KB = 2^{10} bytes, 1 MB = 2^{20} bytes etc.).
    - **GHz**: frequência de clock do processador (ciclos por segundo).
    - **Mbps/Gbps**: velocidade de transferência em redes.
    - Esses conceitos influenciam limites de memória, tamanho de dados e performance de programas em C.

---

4. **Memória e Endereçamento em C**

- A memória é vista como um grande vetor de células numeradas (endereços em hexadecimal).
- **Stack**: variáveis locais de funções; **Heap**: alocações dinâmicas (malloc).
- Ponteiro (int *p) armazena o endereço de memória de outra variável; aritmética de ponteiros permite navegar em arrays.
- Entender endereçamento é crucial para manipulação direta de dados e para estruturas como listas ligadas.

---

5. **Algoritmo, Programa e Software**
   - **Algoritmo**: sequência finita, ordenada e não ambígua de passos para resolver um problema (definição de entrada, processamento e saída).
   - **Programa**: implementação em C de um ou mais algoritmos, obedecendo rigorosamente a sintaxe da linguagem.
   - **Software**: conjunto de programas (de sistema, de aplicativo ou de desenvolvimento) que permite ao usuário realizar tarefas específicas; C e GCC compõem o software de desenvolvimento.

---

6. **Tradutores: Compilador vs. Interpretador**
   - **Compilador (C)**: pré-processamento, tradução para Assembly, assemelhamento (código objeto) e linkedição (criação do executável). Gera binário nativo para a arquitetura, permitindo alta performance.
   - **Interpretador**: lê e executa o código-fonte em tempo de execução; comum em linguagens como Python e JavaScript. Em C, normalmente emprega-se compilador, pois interpretadores para C são menos comuns e menos performáticos.
   - **Vantagens do compilador**: detecção de erros de sintaxe antes da execução e otimizações de performance. **Desvantagens**: necessidade de processo de compilação a cada alteração.

---

7. **Tipos de Erros em C**
   - **Erro de Sintaxe**: falha nas regras gramaticais (ex.: falta de ponto e vírgula, chaves fora de lugar); o compilador indica a linha aproximada.
   - **Erro Lógico**: compilação bem-sucedida, mas resultado incorreto ou diferente do esperado (ex.: usar divisor errado ao calcular uma média); detecta-se testando diferentes entradas e analisando valores intermediários.

---

8. **Processo de Compilação e Linkedição (Etapas)**
   1. **Pré-processamento**: expansão de diretivas (#include, #define), proteção de múltiplas inclusões e remoção de comentários.

2. **Compilação**: conversão de código-fonte C em Assembly específico da arquitetura alvo.
3. **Assemblação**: transformação de Assembly em código de máquina imbutido em arquivos-objeto (.o).
4. **Linkedição**: junção de vários arquivos-objeto e bibliotecas (biblioteca padrão e bibliotecas adicionais) em um executável final; resolve referências externas (funções, variáveis globais).

---

9. **Bibliotecas em C**
    - **Biblioteca-padrão**:
        - <stdio.h> (entrada e saída padrão: printf, scanf).
        - <stdlib.h> (alocação dinâmica, controle de processo, conversões de string).
        - <math.h> (funções matemáticas; requer linkedição com a biblioteca matemática).
        - <string.h> (manipulação de strings: concatenação, comparação, cópia).
    - **Bibliotecas externas**: podem ser adicionadas para funcionalidades específicas (ex.: libcurl, SDL, GTK); exigem incluir cabeçalhos e informar ao compilador onde encontrá-las (flags -I, -L e -l).

---

10. **Estratégias de Planejamento de Algoritmos**
    1. **Solução Narrativa**
        - Descrição em texto corrido, passo a passo, do procedimental que o programa seguirá; ajuda a consolidar a lógica antes de diagramar ou codificar.
    2. **Fluxograma**
        - Representação gráfica usando símbolos padronizados (retângulo para processos, losango para decisões, paralelogramo para I/O); facilita visualizar o fluxo de execução e identificar loops/condicionais.
    3. **Diagrama de Chapin**
        - Macro-diagrama que mostra interações entre módulos, entidades externas e arquivos, evidenciando fluxo de dados e responsabilidades entre processos.
    4. **Pseudocódigo**
        - Texto estruturado com comandos semelhantes a “INÍCIO”, “LEIA”, “ESCREVA”, “SE … ENTÃO … FIMSE” e “ENQUANTO … FIMENQUANTO”; serve de ponte entre a narrativa em linguagem natural e o código C real.

11. **IDE (Ambiente de Desenvolvimento Integrado)**
    - Ferramenta que reúne editor com realce de sintaxe, compilador integrado, depurador (breakpoints, inspeção de variáveis) e gerenciador de projetos/build system (Make, CMake).
    - Exemplos comuns: VS Code (com extensão C/C++), Code::Blocks, Dev-C++, CLion (JetBrains).
    - **Vantagens principais**: autocompletar, realce de erros em tempo real, depuração passo a passo e integração direta com Git/GitHub.

Conclusão
A aula de 11/03 apresentou desde a configuração inicial de GitHub e Git (account, SSH, fluxo de commits) até noções de arquitetura de hardware (CPU, memória, barramentos), medidas de informação (bits, bytes, GB, GHz), e conceitos fundamentais de programação em C (memória, ponteiros, compilação, bibliotecas). Também detalhou as diferenças entre algoritmo, programa e software, o fluxo completo de compilação/linkedição, os tipos de erro (sintaxe vs. lógico) e formas de planejar algoritmos (narrativa, fluxograma, diagrama de Chapin, pseudocódigo). Por fim, destacou a relevância de IDEs para desenvolvimento produtivo e mostrou exemplos básicos de variáveis, I/O e operações aritméticas, cimentando a base para avançar em estruturas de controle nas próximas aulas.