Fundamentos de Sistemas Operacionais
Trabalho 01

Prof. Tiago Alves

Introdução aos Sistemas Operacionais

Introdução

A disciplina de Fundamentos de Sistemas Operacionais trata de diversos tópicos desses sistemas que provêem uma forma intuitiva de se utilizar as funcionalidades de computadores digitais sem que seja necessário ao usuário ou ao programador ter profundo conhecimento das interações entre os diferentes *hardwares* que compõem um computador.

Para construir ou adicionar funcionalidades a esses sistemas computacionais, porém, é necessário conhecimento de linguagens de programação e ferramentas de desenvolvimento. Em nosso curso, o domínio da linguagem C é um pré-requisito para o devido acompanhamento das atividades da disciplina.

Objetivos

1) Exercitar conceitos da linguagem de programação C, especialmente aqueles referentes à programação de sistemas operacionais.
2) Interagir com ferramentas de desenvolvimento para criação, gerenciamento, depuração e testes de projeto de aplicações.

Referências Teóricas

Capítulos 1 e 2 de Mitchell, Mark, Jeffrey Oldham, and Alex Samuel. Advanced linux programming. New Riders, 2001.

Material Necessário

- Computador com sistema operacional programável
- Ferramentas de desenvolvimento GNU/Linux ou similares: compilador GCC, depurador, editor de texto.

Roteiro

1) Revisão de técnicas e ferramentas de desenvolvimento usando linguagem C.

Colete o material acompanhante do roteiro do trabalho a partir do Moodle da disciplina e estude os princípios e técnicas de desenvolvimento de aplicações usando linguagem C e sistema operacional Linux.

2) Realizar as implementações solicitadas no questionário do trabalho.

Nas referências bibliográficas que acompanham o trabalho há uma breve apresentação de ferramentas de depuração para o ambiente Linux.

Implementações e Questões para Estudo

1) Escreva um programa em C que, em tempo de execução, receba do usuário **4 coordenadas cartesianas** dos vértices de um **quadrilátero**, e, como saída, indique se o quadrilátero é convexo. Caso seja, realmente, um polígono convexo, o programa deverá calcular e imprimir sua área.
    - Na sua implementação, deverá constar, no mínimo, três arquivos **.h** e três arquivos **.c**.
        - Utilize um arquivo **.h** para definir os seguintes tipos compostos: ponto no plano cartesiano e quadrilátero, com quatro pontos no plano cartesiano e outros membros que o estudante julgar conveniente.
        - Utilize um arquivo **.c/.h** para realizar as operações geométricas: cálculo do comprimento do lado do polígono, cálculo da condição de convexidade do quadrilátero e cálculo da área do quadrilátero.
        - Utilize um arquivo **.c/.h** com as funções de entrada e saída.
        - Utilize um arquivo **.c** com a função **main()**.
    - A entradas dos pontos do quadrilátero no plano cartesiano deverá seguir uma ordem de conectividade no sentido horário.
        - Tome como exemplo o quadrado unitário nas coordenadas (0,0), (0,1), (1,1), (1,0).
            - O ponto (0,0) é conectado com o ponto (0,1), resultando em lado de comprimento 1.
            - O ponto (0,1) é conectado com o ponto (1,1), resultando em lado de comprimento 1.
            - O ponto (1,1) é conectado com o ponto (1,0), resultando em lado de comprimento 1.
            - E, por conseguinte, conclui-se que o ponto (1,0) se conecte ao ponto (0,0), o que resulta em comprimento 1.
        - Os valores das coordenadas deverão ser manipulado como `double`.
    - Sua implementação deverá ser construída a partir de um **Makefile**.
    - Exemplo de invocação 1:

      `$./meu_programa`
      0 0
      0 1
      1 1
      1 0

Quadrilatero convexo.
Area: 1.

- Exemplo de invocação 2:

```
$./meu_programa
0 0
0 1
0.3 0.3
1 0
Quadrilatero nao convexo.
```

2) Escreva um programa que processe os parâmetros recebidos pela linha de comando.
   - O programa deverá imprimir a quantidade de parâmetros recebidos pela linha de comando e, também, o conteúdo dos parâmetros.
   - Exemplo de invocação:

```
$./meu_programa -a bababa -30 33 21
# de parametros: 5
Nome do executavel: meu_programa
Parametro #1: -a
Parametro #2: bababa
Parametro #3: -30
Parametro #4: 33
Parametro #5: 21
```

3) Escreva um programa que ordene uma lista de inteiros **não-negativos**. A ordem de ordenação deverá ser informada como **parâmetros da linha de comando**. Na ausência de uma *switch* (opção de linha de comando), a ordem deverá ser crescente; caso o usuário indique a opção **-d**, a ordem deverá ser **crescente** e, se indicar **-r**, a ordem deverá ser decrescente.
   - Na sua implementação deverá constar, no mínimo, dois arquivos **.c** e dois arquivos **.h**.
   - Sua implementação deverá ser construída a partir de um **Makefile**.
   - O vetor usado para armazenar temporariamente os inteiros recebidos pela linha de comando deverá ser **alocado dinamicamente** e com quantidade de células adequada para a execução do programa.
   - A entrada do inteiro **-1** deverá sinalizar que o programa já recebeu todos os inteiros e, dessa forma, deverá realizar a ordenação de acordo com o que foi sinalizado na linha de comando.
   - Exemplo de invocação:

```
$./meu_programa -r
20
1
33
44
37
-1
Saida decrescente:
44 37 33 30 1
```

Instruções e Recomendações

A submissão das respostas aos problemas dos trabalhos deverá ser feita através do Moodle da disciplina.

O compilador usado na implementação das soluções deverá ser o GCC.

As soluções do Trabalho 01 deverá ser entregue em um pacote ZIP. A dupla de alunos deverá nomear o pacote ZIP da seguinte forma: nome_sobrenome_matricula_nome_sobrenome_matricula_**trab01.ZIP**. Dentro desse pacote, deverá existir os diretórios q01, q02 e q03, que conterão, respectivamente, as soluções às Questões 1, 2 e 3.

Entre os artefatos esperados, listam-se:

- códigos-fonte C das soluções dos problemas;
- documentação mínima da aplicação:
  - qual sistema operacional foi usado na construção do sistema;
  - qual ambiente de desenvolvimento foi usado;
  - quais são as telas (instruções de uso);
  - quais são as limitações conhecidas;
  - casos de testes que demonstram que a aplicação contempla o comando do trabalho.

Não devem ser submetidos executáveis.

Códigos-fonte C com erros de compilação serão desconsiderados (anulados).

Os trabalhos poderão ser realizados em duplas; a identificação de cópia ou plágio irá provocar anulação de todos os artefatos em recorrência.