UNIVERSIDADE PRESBITERIANA MACKENZIE
Decanato Acadêmico

UNIDADE UNIVERSITÁRIA: FACULDADE DE COMPUTAÇÃO E INFORMÁTICA – FCI
CURSO: CIÊNCIA DA COMPUTAÇÃO
DISCIPLINA: SISTEMAS OPERACIONAIS **ANO/SEMESTRE**: 2022/01
PROFESSOR: EDUARDO FERREIRA DOS SANTOS

LABORATÓRIO 04 – SISTEMAS OPERACIONAIS

RESUMO

Sistemas Operacionais são softwares abstratos que permitem a comunicação dos programas de usuário com os dispositivos de hardware conectados ao computador. Para realizar melhor uso dos recursos disponíveis, é papel do sistema operacional fornecer ferramentas aos programas do usuário que permitam a utilização dos recursos conectados de forma ótima. O Projeto da disciplina aborda o tema da utilização, disponibilização e acesso consumindo as chamadas disponibilizadas pelo sistema operacional.

OBJETIVOS

Objetivo Geral
Compreender o funcionamento do sistema operacional e sua comunicação com os dispositivos conectados.

Objetivos Específicos
1. Abordar o problema da concorrência;
2. Desenvolver um programa de computador que consuma as chamadas de sistema do SO;
3. Conhecer e utilizar de forma ótima as chamadas de sistema do SO.

ORGANIZAÇÃO DO TRABALHO

Grupos
- O trabalho poderá ser realizado em grupos de até **3 alunos**.
- Apesar do trabalho ser em grupo, a produção deve ser individual. Assim, deve ficar **claro e explícito** em todas as etapas do trabalho a contribuição individual de cada um.

METODOLOGIA

Requisitos
- Todos os fontes e artefatos produzidos para a disciplina devem ser disponibilizados em um repositório de acesso público, sem a necessidade de cadastrou e/ou senha para baixar os fontes. Alguns exemplos de repositório público, a saber:
  - https://gitlab.com
  - https://github.com/

METODOLOGIA

- http://bazaar.canonical.com/en/
- https://bitbucket.org/

- Não deve haver nenhuma dependência proprietária para execução e compilação do programa. **Importante**: programas compilados no Windows não devem ser realizados em ferramentas proprietárias, tais como Visual Studio ou Borland;
- Não existe requisito de linguagem de programação. Contudo, a solução proposta deve utilizar os mesmos conceitos apresentados no projeto e atender os mesmos requisitos, o que é facilitado na linguagem C.
- Não se pode fazer nenhuma relação lógica, sendo necessário o calculo massivo dos valores. Ou seja, não se pode igualar ao valor do ln(t) para se chegar a uma aproximação do resultado, nem utilizar nenhum meio de aproximações.

Documentação
Na raiz do Projeto no repositório devem possuir um arquivo README que contenha as instruções detalhadas para a execução do problema. Na construção da documentação devem ser disponibilizadas as seguintes questões:

- Como compilar o programa na plataforma disponibilizada;
- Como executar o programa;
- Como comprovar que os resultados propostos foram alcançados.

Avaliação
No dia da entrega será realizada uma avaliação individual dos trabalhos pelo professor com os grupos, onde a execução do programa com sucesso é um requisito para a avaliação. Caso o programa não seja capaz de cumprir o objetivo proposto, o grupo deve justificar os motivos da falha e explicar, mostrando o código-fonte, o quão perto conseguiram chegar da solução.

Entrega
O grupo deverá submeter o link para o repositório do projeto na tarefa do Moodle aberta para o tema. Deverão constar:

- Link público para o repositório;
- Nome dos componentes do grupo.

Observações de ordem geral:

- Os prazos para entrega não serão estendidos, então fiquem atentos às datas;
- Apesar de não haver restrição de linguagem, toda escolha tecnológica deve ser justificada durante o desenvolvimento do trabalho;
- Durante a apresentação cada grupo deve identificar a contribuição individual de cada componente no desenvolvimento do trabalho;
- Caso não seja possível identificar qual foi a contribuição individual de cada aluno, todos terão a nota 0 (zero) atribuída ao trabalho;
- Após a apresentação será disponibilizado um espaço de 15min para perguntas do professor e dos colegas.

PROJETO

Implementação

O exemplo a seguir, implementado na linguagem C[1], apresenta uma operação de transferência de fundos entre duas contas:

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <sched.h>
#include <stdio.h>

// 64kB stack
#define FIBER_STACK 1024*64

struct c {
  int saldo;
};
typedef struct c conta;

conta from, to;
int valor;

// The child thread will execute this function
int transferencia( void *arg)
{
    if (from.saldo >= valor){                  // 2
      from.saldo -= valor;
      to.saldo += valor;
    }
    printf("Transferência concluída com sucesso!\n");
    printf("Saldo de c1: %d\n", from.saldo);
    printf("Saldo de c2: %d\n", to.saldo);
    return 0;
}

int main()
{
    void* stack;
    pid_t pid;
    int i;


    // Allocate the stack
    stack = malloc( FIBER_STACK );
    if ( stack == 0 )
    {
      perror("malloc: could not allocate stack");
      exit(1);
    }

    // Todas as contas começam com saldo 100
```

1 Inspirado no exemplo disponibilizado em https://www.modernescpp.com/index.php/race-condition-versus-data-race

PROJETO

```
    from.saldo = 100;
    to.saldo = 100;

    printf( "Transferindo 10 para a conta c2\n" );
    valor = 10;

    for (i = 0; i < 10; i++) {
      // Call the clone system call to create the child thread
      pid = clone( &transferencia, (char*) stack + FIBER_STACK,
            SIGCHLD | CLONE_FS | CLONE_FILES | CLONE_SIGHAND | CLONE_VM, 0 );
      if ( pid == -1 )
      {
          perror( "clone" );
          exit(2);
      }
    }

    // Free the stack
    free( stack );
    printf("Transferências concluídas e memória liberada.\n");

    return 0;
}
```

O trecho de código tem como objetivo zerar o saldo de uma conta (*from*) através de várias transferências para a outra (*to*). Contudo, acontece uma falha em sua execução causada por um dos problemas relativos à concorrência apresentados em sala. Para que o código seja executado com sucesso, é necessário identificar o problema e apresentar uma solução, com base nas observações relativas aos problemas e suas soluções discutidos em sala.

Considere os seguintes requisitos para o problema acima:

1. A conta *to* pode receber mais de uma transferência simultânea;
2. A conta *from* pode enviar mais de uma transferência simultânea;
3. A conta *from* não pode enviar dinheiro se não tiver mais saldo;
4. A conta *to* pode trocar de ordem com a conta *from*, ou seja, a conta que enviava pode receber e a conta que recebia pode enviar;
5. Poderão ser realizadas até 100 transações simultâneas de transferência.

PROJETO: Utilizando o exemplo e os conceitos apresentados em sala, implemente uma solução para a condição de corrida apresentada no problema.

RECURSOS DIDÁTICOS

Comunicação entre aluno e professor

- E-mail: eduardo@eduardosan.com
- Whatsapp
- Moodle

RECURSOS DIDÁTICOS

- O Moodle será utilizado para comunicar informações sobre: datas das avaliações, plano de ensino, menção do aluno, faltas do aluno e possíveis ausências ou atrasos do professor.
- O fórum da disciplina poderá ser utilizado para comunicar informações sobre: plano de ensino, datas das avaliações, lista de exercícios, trabalhos, aplicativos e materiais de ensino em geral.

BIBLIOGRAFIA

BÁSICA

Tanenbaum, A. S. and Machado Filho, N. Sistemas Operacionais Modernos. PEARSON, 2010.
Galvin, P. B., Gagne, G., and Silberschatz, A. Operating System Concepts. John Wiley & Sons, Inc. (2013).

COMPLEMENTAR

DEITEL, Harvey M; Deitel. Sistemas operacionais. PEARSON PRENTICE HALL, 2005.
OLIVEIRA, Rômulo Silva De; Carissimi. Sistemas Operacionais. Bookman, 2010.
STALLINGS,. Operating Systems: internals and design principles. 7 ed. Pearson
TANENBAUM, Andrew S.; Steen. Sistemas distribuídos: princípios e paradigmas. Pearson Prentice Hall, 2008.
TANENBAUM, Andrew Stuart; Woodhull. Sistemas Operacionais: Projeto E Implementação [acompanha CD-rom]. Bookman, 2008.