UNIOSTE – Universidade Estadual do Oeste do Paraná
Campus de Foz do Iguaçu/PR
Centro de Engenharias e Ciências Exatas
Curso de Ciência da Computação

Disciplina de Sistemas Operacionais – Trabalho Prático – Simulador de sistema operacional

Data da Entrega: 04/12/2018, até as 23:59h.

Valor do trabalho: 100 pontos, sujeito à penalidade de 50 pontos por dia de atraso na entrega.

Nota: a entrega do trabalho vale como presença para as duas aulas extra-classe do dia 21/11/2018.

Especificação do trabalho:

- O trabalho deve ser realizado em grupo de um a três alunos (preferencialmente 2 a 3 alunos).
- Como parte do trabalho, implemente um simulador de componentes de hardware, os quais operam independentemente, em paralelo [Prof. Johann, UFRGS]. É recomendado o uso de *threads* em uma linguagem como C ou Java para simular o paralelismo.
    1. Os seguintes componentes devem ser modelados: Processador, Disco e Teclado. Cada um desses componentes deve ser implementado em um arquivo de código separado, contendo suas funcionalidades e seus dados.
    2. O objetivo desse trabalho será a simulação de parte de um pequeno sistema operacional com funções mínimas. Em particular, você deve criar as *threads* que representam três componentes que funcionam independentemente: Processador, Disco e Teclado. Cada componente é representado por uma *thread*.
    3. Todas as *threads* iniciam do mesmo modo, executando um laço sem fim e repetindo a mesma sequência de operações:
        i. Imprime uma mensagem na tela;
        ii. Espera um pequeno intervalo (função *sleep*).
    4. No processador, a mensagem lançada na tela é “executei uma instrução”, e a espera é pequena.
    5. No teclado, a mensagem emitida para a tela é “usuário digitou algo”, e somente é emitida a cada vez que é digitado “ENTER” no terminal (use *getchar* do C ou similar para saber quando essa tecla foi digitada). Diferentemente dos demais componentes, o teclado dispensa a espera por um pequeno intervalo. Convém ressaltar que o teclado ilustra o funcionamento de um dispositivo de entrada e saída no simulador. A tela é outro dispositivo, não modelado diretamente neste trabalho mas usada para acompanhamento da execução do simulador.
    6. Finalmente, o disco imprime na tela “dei uma volta”, e espera um tempo muito maior que o processador. Para maior realismo, além do tempo de espera, a *thread* correspondente ao disco deve abrir um arquivo texto, imprimir um número randômico dentro desse arquivo, e fechar o mesmo. Para facilitar, abra o arquivo em modo “*append*”, ou seja, cada operação de escrita adiciona um número ao fim do arquivo.
    7. Se a implementação das *threads* estiver correta, você deve ver que, ao rodar o programa, ele mostrará as mensagens intercaladas das três *threads*, com cada uma imprimindo repetidamente a mesma mensagem.

8. Para aproximar mais o simulador de componentes de um computador real, assuma que a memória é representada por uma variável vetor com vários elementos, sendo acessada pela *thread* representando o processador. Esse vetor deve ser previamente preenchido com valores arbitrários. Na prática, cada um desses valores poderia ser um dado ou uma instrução. Considere também um endereço de leitura (índice do vetor) com um valor inicial correspondente ao endereço (índice) do primeiro elemento do vetor.
9. Assim, o processador deverá, além da mensagem e da espera solicitadas anteriormente, ler um valor da memória (variável vetor), imprimir na tela o valor que leu, e avançar o endereço de leitura (índice do vetor) a cada iteração do seu laço. Convém ressaltar que se o vetor representasse o segmento de código de um processo em execução, com cada elemento armazenando a sequência de bits referente a uma instrução, o endereço de leitura estaria aproximando o papel do registrador "PC". Nesse caso, o processador estaria buscando instruções da memória sequencialmente para executá-las.
10. O disco, apesar de poder funcionar independentemente dos outros componentes, não deve ficar fazendo operações sem ser requisitado. Ele deve aguardar um pedido vindo do processador. Para isso, ele terá uma função que pode ser chamada pelo processador, e um semáforo interno, iniciado com 0. O disco executa *down*() nesse semáforo como primeira instrução dentro do seu laço (a cada iteração), de modo que ficará bloqueado aguardando logo no início da iteração. A função disponibilizada para o processador, quando chamada, deve executar um *up*() nesse semáforo, liberando o disco para dar uma volta – lembre que a espera do disco é grande, e nesse caso você pode imprimir uma mensagem antes e outra depois da espera para poder observar o assincronismo quando roda o programa. Tendo realizado isso, você pode verificar o funcionamento fazendo com que o processador chame essa função de pedido de disco a cada vez que ele encontrar um determinado valor na memória. Assim se simula o funcionamento de uma instrução de acesso a disco.
11. Convém ressaltar que o processador não deve parar após lançar *up()* no semáforo do disco. Adicionalmente, caso o processador tenha chegado ao final da memória, i.e., o último elemento do vetor, ele somente deve realizar as duas operações básicas solicitadas (impressão de mensagem e espera). Para que o processador realize outras tarefas, seria necessário realizar uma troca de contexto, um tema para trabalho futuro que poderia se beneficiar do PCB criado no primeiro bimestre.

- O trabalho deve ser entregue para o email (newtonsp.unioeste@gmail.com) com um arquivo zipado contendo:
  - Todo o código fonte correspondente ao simulador, com comentários descrevendo o propósito de cada função, procedimento, classe e método.
  - Arquivo README.txt com informações básicas sobre pré-requisitos do simulador (sistema operacional, versão da máquina virtual Java, compilador C...), além de configurações necessárias e instruções para execução do programa. O arquivo deve permitir que um programador consiga instalar e executar seu simulador com autonomia.

    - Arquivo com nomes dos alunos que fizeram o trabalho e todas as referências utilizadas (livros, relatórios, artigos, sites).
- O simulador deve ser apresentado ao professor no dia 05/12/2018. A ordem de apresentação é inversa à ordem de entrega de trabalho (*deadline*: 04/12). Assim, o grupo que entregar o trabalho primeiro será o último a apresentar. A apresentação deve durar no máximo 10 minutos e conter no mínimo três itens, com uso opcional de slides:
    1. Informações básicas do código implementado: linguagem utilizada, tecnologias consideradas, bibliotecas e recursos adicionais empregados, pré-requisitos necessários.
    2. Descrição de como cada membro da equipe contribuiu no desenvolvimento do trabalho.
    3. Execução do simulador em um computador/*laptop* trazido pelo grupo.
- Após a simulação, o professor fará perguntas ao grupo sobre o trabalho.