Trabalho 1 - Sincronização de Processos

Carlos Marcelo Pedroso

Universidade Federal do Paraná,
Departamento de Engenharia Elétrica

E-mail: pedroso@eletrica.ufpr.br

Abstract: Este documento especifica atividades a serem desenvolvidas para disciplina de Sistemas Operacionais Embarcados do curso de Engenharia Elétrica com Ênfase em Sistemas Embarcados da Universidade Federal do Paraná. A atividade tem por objetivo compreender o problema da sincronização entre processos e utilizar semáforos para solucionar este tipo de problema. Deverá ser apresentado um relatório respondendo as questões solicitadas, bem como código fonte em linguagem C em sistemas Unix. O formato do relatório está disponível na página da disciplina, bem como um template em LaTeX.

Sumário

1 Objetivo

Compreender os fundamentos de sincronização de processos em sistemas com *time-sharing*.

2 Descrição do Trabalho

Escreva um programa com 5 threads:

Leitor: Lê um arquivo de entrada, uma linha de cada vez. O leitor disponibiliza a linha lida em uma de uma fila de mensagens.

Processador1: Lê uma linha da fila de mensagens e troca todos os espaços pelo caractere '*', atualizando a fila de mensagens.

Processador2: Lê uma linha da fila de mensagens e converte os caracteres para caixa alta, atualizando a fila de mensagens

Processador3: Lê uma linha da fila de mensagens e acrescenta no início da linha o número de caracteres da linha e a quantidade de vogais, entre os símbolos "[" "]".

Escritor: Lê uma linha *já processada* da fila de mensagens e grava em um arquivo de saída, eliminando a mensagem da fila.

ATENÇÃO: As threads Processador1, Processador2, Processador3 e Escritor devem ler a linha da fila de mensagens, fazer o processamento e depois atualizar as modificações na fila de mensagens. Desta forma, devem ser implementadas funções para leitura e atualização de mensagens na fila de mensagens que serão utilizadas por todas as threads. É **proibido** fazer modificações diretamente nas estruturas compartilhadas caracter-a-caracter sem o uso das funções de leitura e atualização mencionadas.

As threads devem ser sincronizadas utilizando semáforos. A sua solução deve permitir que as threads executem de forma paralela. O arquivo de saída deve preservar mesma ordem do arquivo de entrada. A implementação deve ser feita em linguagem C.

Atenção: soluções que transformam a execução das threads em uma simples sequência (ou seja, não permitem a execução das tarefas em paralelo) não serão consideradas válidas.

2.1 Detalhes de implementação

- Não é permitido o uso da chamada fork() ou exec().
- É obrigatório o uso da biblioteca pthread do Unix.
- A leitura do arquivo de entrada pode supor um tamanho máximo de linha de 256 caracteres. Linhas maiores podem ser truncadas, ou seja, o que exceder 256 caracteres na linha pode ser descartado.
- O nome do arquivo de entrada e de saída devem ser argumentos do programa.
- Dicas. Para implementar a thread Processador1, pesquise a função *index*. Para implementar a Thread Processador2, pesquise as funções *islower* e *toupper*.
- O programa deve ser finalizado quando não houver mais linhas no arquivo de entrada.

2.2 O que deve ser entregue

1. Código fonte com a solução para compilação em sistemas Unix.
2. Relatório explicando a solução adotada, na forma textual e em pseudo-código. Deve ser detalhado o uso de semáforos para exclusão mútua ou para sincronização.
3. Comparação de tempo de execução do sistema multithread e monothread. Para isso, deve ser implementada a solução sequencial, em que o processamento seguinte é realizada após o fim do anterior. Dica: registre a diferença de tempo entre o início e o fim do programa (ver exemplo em http://www.eletrica.ufpr.br/pedroso/2015/TE244/tempo-microsegundos.c). A comparação dos resultados deve utilizar resultados com diferentes tamanhos de arquivo de entrada e também o número de processadores disponíveis (note que o exercício exige que o programa seja executado em um sistema com mais de um processador disponível), bem como a presença ou não de outros processos consumindo CPU.
4. Comparação de tempo de execução de dois programas multithread simultâneos, mas com prioridades base diferentes. Execute simultaneamente dois programas e altere a prioridade base (ver comando nice). O arquivo de entrada deve ser grande o suficiente para que o tempo de execução seja na escala de segundos. Anote o tempo de execução de cada programa, e compare os resultados com uma diferença de prioridade base de 0, 1, 2, 3,..., 20 níveis, apresentando em forma gráfica. Analise o resultado considerando o mecanismo de atualização de prioridade dinâmica usado no Linux.

3 Critérios de avaliação

O trabalho deve ser implementado em equipes de até 2 pessoas. Não serão admitidos trabalhos desenvolvidos por equipes maiores. A avaliação será realizada da seguinte considerando os seguintes critérios:

1. Qualidade da apresentação: 10 pontos;
2. Emprego apropriado de semáforos: 50 pontos;
3. Realização dos testes e apresentação de resultados: 40 pontos;

Em caso de cópias, as equipes envolvidas terão grau zero.