Trabalho 1 - Gerência e 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 gerência e sincronização de processos em sistemas com *time-sharing*.

2 Descrição do Trabalho

Considere o programa servidor TCP disponível na página do curso. Trata-se de um programa multithread cujo objetivo é implementar um servidor que retorna frases de um dicionário de ditados populares. O programa implementa um protocolo para diversas funcionalidades:

1. GETR: retorna um ditado randômico.
2. GETN: retorna um ditado específico.
3. REPLACE: substitui um ditado por outro.
4. VER: retorna a versão do programa.
5. FIM: encerra a conexão.

O primeiro passo é compilar e testar o programa servidor. Isto pode ser feito com os comandos a seguir:

```
$ gcc -o servidorD servidorD.c -lpthread
$ ./servidorD 9551
```

abra um terminal em outro computador e teste o servidor usando como cliente o programa *telnet*:

```
$ telnet IP-DO-SERVIDOR 9551
Connected to localhost.
Escape character is '^]'.
Requisição 1
GETR

Ditado 1: Água mole em pedra dura, tanto bate até que fura.
Requisição 2
GETN
4
Casamento e mortalha no céu se talha.
Requisição 3
REPLACE
5

OKTeste

OKRequisição 4
GETN
5
Teste
Requisição 5
FIM
```

É necessário incrementar as funcionalidades do servidor incluindo:

1. DEL: apaga um ditado.
2. ROTATE: troca a posição de um ditado com outro no vetor de ditados.
3. SEARCH: procura a ocorrência de um termo e retorna todos os ditados com esta ocorrência
4. PALAVRAS-D: retorna o número total de palavras de um ditado
5. PALAVRAS-T: retorna o número total de palavras de todos os ditados do dicionário
6. ALTERACOES: retorna o número de alterações realizadas na base de dados de dicionário.
7. GRAVA: servidor deve gravar o dicionário em arquivo.
8. LE: servidor deve carregar um arquivo de dicionários para memória.

Cada equipe deve implementar as funcionalidades listadas acima a partir do código base. A partir disso, a equipe deverá:

2.1 Parte 1

- Implementar vários programa de teste (cliente TCP) que solicita várias operações ao servidor. Os programas de teste devem ser lançados simultaneamente utilizando um *shell script*. Devem ser implementados pelo menos 10 programas monothread realizando 1000 operações diferentes.

- Identificar os erros causados por problemas de concorrência.

- Resolver os problemas de concorrência utilizando semáforos. Deseja-se manter o máximo de paralelismo possível na execução das threads.

2.2 Parte 2

Lançar na máquina que o servidor está sendo executado processos consumidores de CPU de forma a ocupar 100% do processador.

- Anote os tempos de resposta para as sequências de testes do servidor.

- Altere a prioridade do processo consumidor de CPU e do processo servidor (comando *nice*) para maior e menor possível analise se houve alteração no tempo de resposta. Mantenha a política de time-sharing padrão (SCHED_OTHER). Use as prioridades base -20, -10, 0, 10 e 20 e execute combinações em número suficiente para permitir determinar se há um padrão de comportamento no tempo de resposta..

- Altere a política de escalonamento do processo consumidor de CPU e do processo servidor. No Linux a política padrão de escalonamento é a realização de time-sharing (SCHED_OTHER). Veja explicações em (https://askubuntu.com/questions/51283/how-to-run-a-program-with-sched-rr-policy-from-command-line e www.informit.com/articles/article.aspx?p=101760&seqNum=4). Você deve alterar para SCHED_RR que é a política de tempo real. Configure os processos consumidores de CPU como *real-time* e depois faça o mesmo com o servidor - use as combinações possíveis e analise o tempo de resposta do servidor.

2.3 O que deve ser entregue

1. Código fonte com a solução para compilação em sistemas Unix.

2. Código fonte do programa de teste.

3. 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.

4. Comparação de tempo de execução do sistema nas diversas situações solicitadas. 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).

3 Critérios de avaliação

O trabalho deve ser implementado em equipes de até 3 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: 40 pontos;
3. Realização dos testes e apresentação de resultados: 50 pontos;

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