DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM INFORMÁTICA
SISTEMAS OPERACIONAIS I
2º SEM/2011
Trabalho em Grupo – Nr 2

ESTUDO SOBRE THREADS

---

1. Objetivo do Trabalho

Estimular a capacidade do aluno de trabalhar em equipe para organizar, projetar e desenvolver soluções para problemas formulados que envolvam o estudo e o conhecimento sobre threads.

2. Escopo do Trabalho

- ✓ Estudar comandos indicados.
- ✓ Conceber e implementar os algoritmos conforme as questões apresentadas.
- ✓ Preparar um relatório em Word.
- ✓ Entregar todo o material elaborado (códigos fontes, executáveis e relatório) em meio magnético (CDROM). O relatório deve também ser entregue impresso.

3. Equipes de Trabalho

Devem ser formadas com 3 alunos cada. Excepcionalmente pode haver uma equipe com 2 alunos tendo em vista o número de inscritos.

4. Prazo de Entrega do Trabalho

O material deverá ser entregue na aula do dia **31/10**

5. Penalidades

Caso o grupo atrase a entrega do resumo seu grau final sofrerá um decréscimo na razão de 0,5 pontos por dia.

6. Avaliação

Serão considerados os seguintes aspectos: estética da apresentação do material escrito e conteúdo.

7. Temas para Desenvolvimento

a. Prog1 – Threads Cooperativas

Construa o mesmo programa do trabalho 1, agora num ambiente de múltiplas threads em nível de kernel, de forma a poder comparar as diferentes abordagens

a) Gere aleatoriamente uma matriz de dimensões “m” linhas e “k” colunas (fornecidos como parâmetros em tempo de execução);
b) Encontre o menor e o maior valor da matriz e suas respectivas coordenadas;

c) Calcule os “m” produtos internos conforme a fórmula

$$PI_i = \sum_{j=1}^{k} A_{i,j} * A'_{j,i}$$

d) Encontre o maior e o menor produto interno gerado e suas respectivas coordenadas.

(*) Obs:

1. Em todas as versões do programa, o tempo total de execução deve ser computado e apresentado na tela ao final da execução.
2. O programa deve permanecer em “loop” até que seja fornecido um valor zero para “m” e “k”.

- Rode o programa em uma máquina com apenas um processador (desabilite o outro ou outros possivelmente existentes) e avalie o desempenho alcançado.

- Habilite agora o segundo processador, execute novamente o programa, avalie o desempenho e o compare com o obtido no item anterior.

- Compare os dois resultados acima com os obtidos com as versões de 1 e vários subprocessos para os mesmos tamanhos da matriz A.

Teste e compare o desempenho da segunda versão (2 ou 4 cores) para diferentes valores de “m” e “k” (varie-os de pequenos – da ordem de dezenas (m = 10 e 20, k = 20, 30, 100, 1000, 10000 até m = 10, 20, 100, 1000 e 10000 e k = 20, 30 e 100, por exemplo).

b. Prog2 – Threads

Construa um programa multithread que implemente o contexto clássico conhecido como produtor / consumidor.

a) Simule um ambiente assíncrono com 3 produtores e 5 consumidores. Deverão ser produzidos 1000 produtos ao todo. Assuma que os produtos produzidos são inseridos em uma fila circular com 50 posições. Os produtores inserem ao final da fila e os consumidores consomem do início da fila. Construa uma interface gráfica para mostrar a situação: da fila de produtos; do estado das threads ativas (produtores e consumidores) – em execução, pronto ou sleeping (paradas por sincronismo ou exclusão mútua); e a situação de cada semáforo existente (tipo, valor atual, threads por ventura presas na fila)

b) Crie uma nova versão do programa acima incluindo as seguintes regras de negócios:

- ✓ Os produtores têm prioridade no acesso à fila circular (se produtores e consumidores estiverem aguardando acesso à fila, os produtores seguem primeiro);

- ✓ Consumidores podem acessar a fila circular conjuntamente (2 ou mais consumidores podem estar acessando a fila. (cuidado para não gerar inconsistência no acesso).

BOM TRABALHO

Prog1.c

```
#include    <stdio.h>
#include    <wait.h>
#include    <unistd.h>

int main(void)
{
    int       status, id;

    ********* Qual o PID deste processo?

    if (fork() != 0)
    {
            ***** Quem executa este trecho de código?
            execl("/Bin/ls", "ls", NULL);

            ***** O que acontece após este comando?
            ***** O que pode acontecer?

    } else
    {
            ***** Quem executa este trecho de código?
    }

    wait(&status);

            ***** Quem executa este trecho de código?

    if (status == 0)

            ***** Quem executa este trecho de código?
    else

            ***** Quando este trecho de código é executado?
}
```

Prog2.c

```
#include    <stdio.h>
#include    <wait.h>
#include    <sys/types.h>

#define     x       m;

int         i, j, k, id, d1, d2;
int         rc;

int main(void)
{
    d1 = 0; d2 = 0;

    j = 0;

    for (i = 0; i <= x; i++)

    {
            ***** mostre quem executa este trecho.

            ***** mostre o valor atual de d1 e d2.

            rc = fork();

            if (rc)
            {

                ***** verifique quem executa este trecho.

                d1 = d1 + (i + 1);

                d2 = d2 + d1 * 3;

                ***** verifique o valor e o escopo de d1 e d2.

            } else
            {

                ***** verifique quem executa este trecho.

                j = i + 1;

                d1= d1 + 10;

                d2 = 1.5 * d1;

                ***** verifique o valor e o escopo de d1 e d2 e os compare com os valores no
                outro trecho.
            }
    }
```

```
                ***** verifique quem executa este trecho.

        if (rc != 0)
        {
                ***** verifique quem executa este trecho.

                for (i = j; i == x; i++)
                {
                        **** Verifique se o comando "for" está correto, de forma a aguardar todos
                        os subprocessos criados

                        wait(&rc);

                        if (rc == 0)

                         **** O que ocorre quando este trecho é executado?
                        else

                         **** O que ocorre quando este trecho é executado?
                }
        }
        exit(0);
}
```

Rio de Janeiro, 13 de abril de 2011