DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM INFORMÁTICA
SISTEMAS OPERACIONAIS I
2º SEM/2013
Trabalho em Grupo – Nr 2

ESTUDO SOBRE SUBPROCESSOS E THREADS

---

1. Objetivo do Trabalho

Estimular a capacidade do aluno a trabalhar em equipe no desenvolvimento de soluções para problemas que envolvam o estudo e o conhecimento sobre subprocessos e threads.

2. Escopo do Trabalho

- ✓ Estudar comandos indicados.
- ✓ 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.
- ✓ Eventualmente, apresentar e discutir as soluções em sala de aula.

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 **29/10**.

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

Ambiente para desenvolvimento: Linux

a. Estudo de Comandos

Estude, numa plataforma Linux, os comandos: fork(); exec(); execl(); wait() e exit(); getpid(), getppid().

Leia o material sobre Comunicação entre Processos. Use a plataforma Linux para os itens a seguir, de “b” a “d”.

b. Prog1 (em anexo)

- ✓ Faça “m = 1”, complete o código em anexo conforme solicitado, execute o programa, analise e explique o que ocorre.
- ✓ Identifique a função da variável “j” e certifique-se de que funciona corretamente, caso não funcione corrija.
- ✓ Faça sucessivamente “m = 2” e “m = 3” e descreva o que ocorre em cada caso.

c. Prog2 – Subprocessos Cooperativos

Construa um programa que:

a) Gere uma matriz aleatória quadrada de dimensões “k” (fornecido como parâmetro em tempo de execução);
b) Encontre o menor e o maior valor da matriz e suas respectivas coordenadas (x,y);
c) Calcule os “k” produtos internos conforme a fórmula

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

d) Encontre o maior, o menor, o valor médio e o desvio padrão dos produtos internos gerados.

(*) 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 “k”.

Versão 1 do programa: Construa toda a solução em um único fluxo de execução.

Versão 2 do programa: Faça uso de sub-processos cooperativos que retornem ao processo pai a parte que lhes coube calcular e rode-o em uma máquina com apenas um processador (ou desabilite os demais processadores).

Versão 3 do programa: Habilite os demais processadores, de forma a ter 2 ou mais processadores.

Teste e compare o desempenho das versões para diferentes valores de “k” até que sinta ter compreendido a tendência do comportamento apresentado (varie-os de k = 10, 20, 30, 100, 1000, 5000 e 10000 ou mais, por exemplo).

Analise os resultados obtidos, apresente-os em uma tabela e descreva suas conclusões. Procure acompanhar o uso da(s) CPU(s).

d. Prog3 – Threads

Construa um programa **multithread** que resolva o mesmo problema do item anterior.

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

Analise os resultados obtidos, apresente-os em uma tabela e descreva suas conclusões.

e. Prog4 – Threads (pode ser desenvolvido em C ou Java)

Construa um programa multithread que resolva o problema clássico conhecido como produtor / consumidor.

Simule um ambiente assíncrono com 2 threads produtoras e 3 consumidoras. Deverão ser produzidos 5000 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.

Mostre, de alguma forma, o estado da fila e controle para que a fila não seja lida quando vazia nem que o produtor insira um novo item com a fila cheia.

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?
}
```

Prog.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)
            {

                ***** mostre quem executa este trecho.

                d1 = d1 + (i + 1);

                d2 = d2 + d1 * 3;

                ***** mostre o valor e discuta o escopo de d1 e d2.

            } else
            {

                ***** verifique quem executa este trecho.

                j = i + 1;

                d1= d1 + 10;

                d2 = 1.5 * d1;

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

```
                ***** diga 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 processos

                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, 04 de outubro de 2013