

**Data de Entrega: A definir – NÃO SERÃO ACEITOS TRABALHOS ENVIADOS APOS ESTA DATA.**

**O que deve ser entregue:**

1. Deve ser enviado pelo SIGAA um arquivo zipado contendo os códigos fontes dos programas implementados. Use o nome especificado em cada questão para nomear o seu programa;

**Como será avaliado o trabalho:**

A avaliação deste trabalho levará em conta a correção e qualidade das questões respondidas. Cada questão descrita abaixo possui uma pontuação definida.

### Trabalho Pratico 3 - Threads

Implemente os seguintes programas relacionados abaixo. Antes de iniciar os exercícios, execute os programas exemplos de uso de threads apresentados no final deste documento.

1. A sequência de Fibonacci é a série de números 0,1,1,2,3,5,8... Formalmente, pode ser expressa como:  
$$\text{Fib}(0) = 0$$
$$\text{Fib}(1) = 1$$
$$\text{Fib}(n) = \text{Fib}(n-1) + \text{Fib}(n-2)$$

Escreva o programa FIBONACCI.C usando a biblioteca de threads Pthreads (Linux) para gerar a serie de Fibonacci. O programa deve funcionar da seguinte maneira: O usuário deve entrar através da linha de comando o numero de termos da serie de Fibonacci a ser gerado. O programa criara' uma thread separada que gerara' os números da serie de Fibonacci, colocando a sequencia em uma estrutura de dados compartilhada com a thread pai (um array deve ser a estrutura de dados mais conveniente). Quando a execução da thread termina a thread pai ira' imprimir a sequencia gerada pela thread filha. Como a thread pai não pode começar imprimir a sequencia de Fibonacci ate' que a thread filha termine, a thread pai tera' que esperar a finalização da thread filha usando técnicas de para aguardar a finalização de threads;

2. Modificar o servidor baseado em sockets, apresentado na Figura abaixo, para que atenda cada solicitação de cliente em uma thread separada.

```
import java.net.*;
import java.io.*;

public class DateServer
{
    public static void main(String[] args) {
        try {
            ServerSocket sock = new ServerSocket(6013);

            // now listen for connections
            while (true) {
                Socket client = sock.accept();

                PrintWriter pout = new
                    PrintWriter(client.getOutputStream(), true);

                // write the Date to the socket
                pout.println(new java.util.Date().toString());

                // close the socket and resume
                // listening for connections
                client.close();
            }
        }
        catch (IOException ioe) {
            System.err.println(ioe);
        }
    }
}
```

3. No exercício 2 do Trabalho 2 (Comunicação entre processos) foi desenvolvido um servidor para realizar cálculos matemáticos. Porém, este servidor é single threaded, o que significa que o servidor não pode responder a mais de uma solicitação por vez. Modifique esta solução para que cada o servidor atenda cada solicitação de cliente em uma thread separada.
4. Modifique a solução do exercício 3 anterior para que cada cliente seja atendido usando uma thread pool. Experimente os três modelos de thread pool apresentados.

## Exemplo de thread usando a Biblioteca Pthreads

```
#include <pthread.h>
#include <stdio.h>

int sum; /* this data is shared by the thread(s) */

void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
    pthread_t tid; /* the thread identifier */
    pthread_attr_t attr; /* set of attributes for the thread */

    if (argc != 2) {
        fprintf(stderr, "usage: a.out <integer value>\n");
        /*exit(1);*/
        return -1;
    }

    if (atoi(argv[1]) < 0) {
        fprintf(stderr, "Argument %d must be non-negative\n", atoi(argv[1]));
        /*exit(1);*/
        return -1;
    }

    /* get the default attributes */
    pthread_attr_init(&attr);

    /* create the thread */
    pthread_create(&tid, &attr, runner, argv[1]);

    /* now wait for the thread to exit */
    pthread_join(tid, NULL);

    printf("sum = %d\n", sum);
}

/**
 * The thread will begin control in this function
 */
void *runner(void *param)
{
    int i, upper = atoi(param);
    sum = 0;

    if (upper > 0) {
        for (i = 1; i <= upper; i++)
            sum += i;
    }

    pthread_exit(0);
}
```

## Exemplo de thread usando a Biblioteca Win32

```
#include <stdio.h>
#include <windows.h>

DWORD Sum; /* data is shared by the thread(s) */

/* the thread runs in this separate function */
DWORD WINAPI Summation(PVOID Param)
{
    DWORD Upper = *(DWORD *)Param;

    for (DWORD i = 0; i <= Upper; i++)
        Sum += i;

    return 0;
}

int main(int argc, char *argv[])
{
    DWORD ThreadId;
    HANDLE ThreadHandle;
    int Param;

    // do some basic error checking
    if (argc != 2) {
        fprintf(stderr, "An integer parameter is required\n");
        return -1;
    }

    Param = atoi(argv[1]);

    if (Param < 0) {
        fprintf(stderr, "an integer >= 0 is required\n");
        return -1;
    }

    // create the thread
    ThreadHandle = CreateThread(NULL, 0, Summation, &Param, 0, &ThreadId);

    if (ThreadHandle != NULL) {
        WaitForSingleObject(ThreadHandle, INFINITE);
        CloseHandle(ThreadHandle);
        printf("sum = %d\n", Sum);
    }
}
```

### Exemplo de thread usando a Biblioteca Java Threads

```
class Sum
{
    private int sum;

    public int get() {
        return sum;
    }

    public void set(int sum) {
        this.sum = sum;
    }
}

class Summation implements Runnable
{
    private int upper;
    private Sum sumValue;

    public Summation(int upper, Sum sumValue) {
        if (upper < 0)
            throw new IllegalArgumentException();

        this.upper = upper;
        this.sumValue = sumValue;
    }

    public void run() {
        int sum = 0;

        for (int i = 0; i <= upper; i++)
            sum += i;

        sumValue.set(sum);
    }
}

public class Driver
{
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage Driver <integer>");
            System.exit(0);
        }

        Sum sumObject = new Sum();
        int upper = Integer.parseInt(args[0]);

        Thread worker = new Thread(new Summation(upper, sumObject));
        worker.start();
        try {
            worker.join();
        } catch (InterruptedException ie) { }
        System.out.println("The sum of " + upper + " is " + sumObject.get());
    }
}
```