Universidade Federal de Roraima
Departamento de Ciência da Computação
Sistemas Operacionais

# Atividade - Aula 05/08

**Atenção: Vale ressaltar que esta atividade será usada como critério para a contabilização de sua frequência de aula.**

**Prazo de Entrega: 09/08/2021**

Aluno: GUILHERME LUCAS PEREIRA BERNARDO

**1. Conceitue e diferencie threads em modo kernel e usuário.**

**R:**
Threads em modo usuário são implementadas por chamadas a uma biblioteca de rotinas que são ligadas e carregadas em tempo de execução (run-time) no mesmo espaço de endereçamento do processo e executadas em modo usuário.

Threads em modo kernel (TMK) são implementados diretamente pelo núcleo do sistema operacional, através de chamadas a rotinas do sistema que oferecem todas as funções de gerenciamento e sincronização

Sendo assim os dois modos diferem no quesito de implementação onde o modo usuário é chamado por uma biblioteca de rotinas ligado ao usuário enquanto o modo kernel executa a thread diretamente no CORE do sistema oferecendo mais opções.

**2. Usando o site www.kernel.org , analise o código fonte da última versão para identificar e apresentar um exemplo do uso threads em modo kernel.**
**R:**
As threads do arquivo debug_core.c, que monitoram e capturam informações relevantes do Kernel, CPU, dados, memoria e etc.
**3. Utilizando o código disponível (arquivo `class_src_0308.zip`) no site da disciplina no tópico de aula Processos: Threads e Modelos Multithreading, modifique o código em C ou Rust que usa múltiplas thread para utilizar a rotina `pthread_join()` que espera pelo término de uma thread. Apresenta os resultado da execução com e sem o uso da rotina `pthread_join()`**
**R:** Sem pthread_join:

```
PS D:\Documentos\GitHub\DCC403-SistemasOperacionais-20211-ERE\Exercicios\exercicio8-codes\class_src\C> gcc -pthread multi_thread.c -o m
ulti_thread
PS D:\Documentos\GitHub\DCC403-SistemasOperacionais-20211-ERE\Exercicios\exercicio8-codes\class_src\C> .\multi_thread.exe
No main: criando thread 0
No main: criando thread 1
No main: criando thread 2
No main: criando thread 3
No main: criando thread 4
Thread #1!
Thread #2!
Thread #0!
Thread #3!
Thread #4!
```

Com pthread_join:

```
PS D:\Documentos\GitHub\DCC403-SistemasOperacionais-20211-ERE\Exercicios\exercicio8-codes\class_src\C> gcc -pthread multi_thread.c -o m
ulti_thread
PS D:\Documentos\GitHub\DCC403-SistemasOperacionais-20211-ERE\Exercicios\exercicio8-codes\class_src\C> .\multi_thread.exe
No main: criando thread 0
Thread #0!
No main: criando thread 1
Thread #1!
No main: criando thread 2
Thread #2!
No main: criando thread 3
Thread #3!
No main: criando thread 4
Thread #4!
```

**4. Escreva um programa na linguagem de programação C utilizando threads para identificar todos os números pares de uma lista com N números.**

**R:**

```c
#include <stdio.h>
#include <pthread.h>
#define N 5

void *isEven()
{
    int a[N], i, even = 0, odd = 0;
    printf("Digite %d numeros inteiros\n", N);
    for (i = 0; i < N; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("\n\nOs numeros pares dos %d numeros digitados
sao...\n", N);
    for (i = 0; i < N; i++)
    {
        if (a[i] % 2 == 0)
        {
            printf("%d\t", a[i]);
            even++;
        }
    }
    pthread_exit(NULL);
}

int main()
{
    pthread_t mythread;
    pthread_create(&mythread, NULL, isEven, NULL);
    pthread_join(mythread, NULL);
    pthread_exit(NULL);
    return 0;
}
```