**Relatório do µTP5 de Sistemas Operacionais**

Alunos: Emanuel Jesus Xavier e Vinicius Nascimento Targa

Matrículas: 18.4148 e 18.1.4088

**1. Introdução**

Este trabalho prático tem o objetivo de implementar o problema do jantar dos filósofos de três maneiras: Utilizando Mutex, Semáforos e Forks. Esta documentação tem a intenção de definir cada uma desses métodos e compará-los, ressaltando suas peculiaridades.

**2. Desenvolvimento**

**2.1. Thread**

Thread é um pequeno programa que trabalha como subsistema, sendo uma forma de dividir processos em mais de uma tarefa. Essas tarefas podem executar simultaneamente ou revezando tão rápido que praticamente estão rodando em paralelo.

**2.2. Thread Mutex**

Região crítica é um trecho de código que utiliza alguma variável / região de memória compartilhada e não pode ter interferência externa de outras threads, o seu objetivo é transformar o recurso às operações com compartilhado em atômico, isto é, suas instruções não devem ser interrompidas. Quando definimos uma zona crítica o sistema operacional entende que a thread não deve ser interrompida durante a execução daquele trecho de código.

**2.3. Semáforo**

Um semáforo é utilizado para gerenciar processos concorrentemente utilizando uma variável inteira. Essa variável é utilizada para resolver o problema da região crítica para se alcançar sincronização de processos em um ambiente que utiliza diversos processos.

**2.4. Fork**

Forks são usados para criar novos processos (processo filho) a partir de um existente (processo pai). O processo pai é aquele que faz a chamada da função fork() e cria um processo filho, que executa concorrentemente com o pai. Após um processo filho ser criado, ambos os processos executarão as instruções após o comando fork() ao mesmo tempo.

**2.5** **Comparação**

é uma técnica usada em programação concorrente para evitar que dois processos ou *threads* tenham acesso simultaneamente a um recurso compartilhado

Thread Mutex é um trecho ou bloco de instruções denominado de zona crítica, que não será executada em paralelo com outros, todas as operações realizadas neste bloco são atômicas para que dois processos ou threads não tenham acesso simultaneamente ao um recurso compartilhado. Semáforo é um recurso que está em uso, enquanto uma thread ou processo usa aquele recurso ele fica bloqueado para os outros