Universidade de Brasília (UnB)
Departamento de Ciência da Computação

Disciplina: Fundamentos de Sistemas Operacionais
Prof: Alba C. M. A. Melo

Descrição do Trabalho Prático (02/2025)

1. **Título:** Shell para escalonamento preemptivo-interrupt com prioridades estáticas

2. **Pré-requisitos**
   O aluno deverá conhecer a linguagem de programação C e chamadas de sistema Unix.

3. Especificação:
O presente trabalho consiste do projeto de um shell_sched que vai implementar um escalonador preemptivo-interrupt com prioridades dinâmicas no sistema operacional Linux ou MacOS. Inicialmente, será lançado o shell_sched, que criará o processo escalonador e fará o print do prompt na tela. No shell_sched, os comandos possíveis são: create_user_scheduler <number_of_queues>; execute_process <command priority>; list_scheduler; exit_scheduler.

3.1 Processo shell_sched: Será executado via shell.

Descrição:
`> shell_sched`
Após a execução desse comando, deve-se imprimir o prompt:
`">shell_sched: "`

3.2 Processo user_scheduler <number_of_queues>: Será executado via shell_sched.

Descrição:
Será criado um processo shell_sched, que inicializará o número de filas fornecido como parâmetro. Podemos ter 2 ou 3 filas round-robin, onde a prioridade 1 é a mais alta. Após a inicialização, o processo ficará esperando requisições de escalonamento. Cada requisição vai colocar o processo no final da fila de sua prioridade. Caso haja processo com prioridade maior do que o processo em execução, o processo com maior prioridade é executado. O processo em execução é colocado no início de sua fila e, quando obtiver a CPU, executará pelo restante do quantum.

3.3 Processo execute_process <command priority>: Será executado via shell_sched, e se comunicará com o processo user_scheduler, informando os dados do novo processo. Os processos a serem executados devem ser CPU-bound, com duração em torno de 20 segundos.

3.4 Processo list_scheduler: será executado via o shell_sched e apresentará informações sobre os processos nas filas de prioridade e o processo que está em execução.

3.5 Processo exit_scheduler: terminará o escalonador, apresentando o tempo de turnaround de cada processo e os processos que ainda não terminaram sua execução.

Atenção: Os processos devem utilizar os mecanismos de comunicação Unix apresentados em sala de aula: pipes, filas de mensagem (msg), memória compartilhada (shm), semáforos (sem) ou sinais. Não pode ser utilizado o semaphore da biblioteca pthread.

4. Documentação:

O código fonte do escalonador deve ser entregue, tendo como comentário, no início do programa, o nome dos alunos e matrícula, versão do compilador e versão do sistema operacional.

Bom trabalho !!!!!