PUCRS - Escola Politécnica
Disciplina: Sistemas Operacionais - 2022/2 - Trabalho Prático - Escalonamento
Prof. Fernando Luís Dotti

Nesta fase do trabalho introduzimos o escalonamento de processos.

1. Antecedentes

Assume-se que você dispõe do trabalho 1, com paginação, operante. Poderemos carregar vários processos em memória e com um novo comando, seu Sistema Operacional escalonará todos os processos.

2. Escalonamento

Neste passo adotaremos uma política de escalonamento de processos. Assim, faz-se necessário poder parar e retomar a execução de processos, em qualquer ponto (após o final da execução da instrução corrente na CPU). Para isso, teremos que **salvar** o contexto da CPU, no momento que o processo é retirado da mesma. Este estado da CPU será **restaurado** quando o processo retorna à CPU para continuar sua execução. O contexto da CPU é salvo no PCB do processo, em um campo criado para isso.

O escalonamento em um sistema pode ser provocado por diferentes eventos que bloqueiam o processo atual, fazendo-se necessário escalonar outro. **Neste trabalho,** exercitaremos apenas a **perda de processador por tempo de uso.** Ou seja, assim que um processo "entra" na CPU, ele executa por um tempo *Delta*. Para facilitar a implementação, este *Delta* pode ser em ciclos de CPU. Ou seja, *Delta* diz quantas instruções um processo executa na CPU. Então ele "sai" da CPU.

Após este número *Delta* de instruções, uma interrupção é gerada. Exatamente isso. "Liga-se" a interrupção do relógio, que no nosso trabalho serve somente para escalonamento. Ao final da execução da instrução, a CPU verifica se tem interrupção, e desvia para a rotina específica (como qualquer interrupção). A rotina de tratamento desta interrupção deve fazer o *salvamento* de contexto do processo interrompido, colocá-lo na fila de processos *prontos* a executar, escolher o próximo processo desta fila que deve ir para a CPU, *restaurar* o contexto deste na CPU, permitir então a CPU a executar o processo por *Delta* unidades - quando este ciclo explanado se repete.

Desta maneira, os processos ciclicamente ganham o direto de executar na CPU. Quando um processo acaba, este é desalocado do sistema e obviamente não será mais executado.

2.1. Observações

Em um sistema real existe um relógio independente no HW. Este relógio interrompe periodicamente a CPU. A CPU desvia para a rotina de tratamento do relógio. Entre outras funções, esta rotina avalia se o processo que está rodando deve ser trocado. No nosso sistema sugerimos implementar o relógio como um contador de instruções executadas na CPU. Mas em um projeto multithreaded você pode ter uma thread relógio que gera a interrupção.

2.2. Fim de Processo

A instrução STOP passa é uma chamada de sistema. A rotina de tratamento respectiva deve deslocar o processo que executou STOP. Isso significa liberar a memória e desalocar o PCB. Além disso, o escalonamento de um novo processo para ocupara a CPU deve ser efetuado.

3**. Testes**

Crie um novo comando no seu sistema, execAll por exemplo. Este comando dispara a execução de todos processos carregados de forma escalonada.

Carregue vários processos em memória. Estes processos estão todos parados, na fila ready.
Então dispare execAll. A partir disso, o ciclo exposto acima deverá acontecer até que todos processos acabem.
Deve ser possível acompanhar o progresso dos diferentes processos ao longo do tempo e as atividades de escalonamento.

Isto significa que os processos, assim que submetidos ao sistema, já poderão ser escolhidos para execução na CPU conforme critério do escalonador.