Lista de Exercícios - Sistemas Operacionais
Aula 4: *Processos*

Professores: Valmir C. Barbosa e Felipe M. G. França
Assistente: Alexandre H. L. Porto

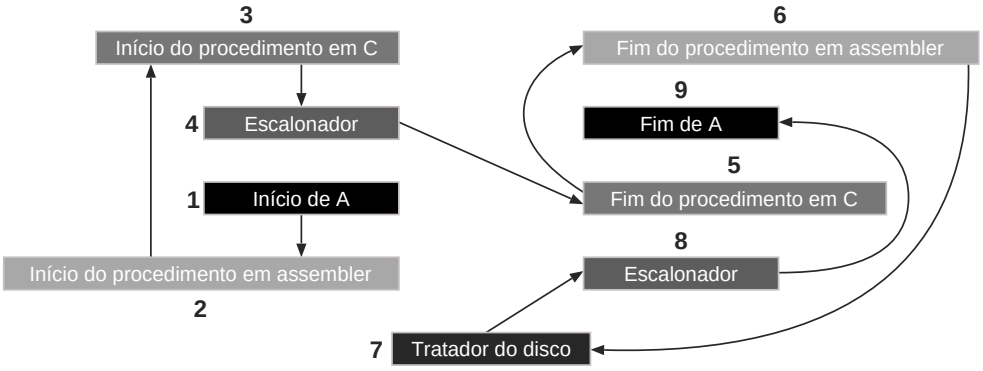
1. Um processo que realize uma computação estritamente sequencial pode ser visto como um único fluxo de controle. No entanto, no caso mais geral, é possível que um processo possa ter mais de um fluxo de controle, correspondendo a ações executadas concorrentemente. Cada um desses fluxos de controle é chamado de **thread** (fio). Assim como os processos, as threads são escalonadas para execução pelo sistema operacional. Porém, o escalonamento das threads é mais leve que o dos processos, já que seu contexto é bem menor. A pilha deve ou não fazer parte desse contexto a ser salvo no caso das threads? Justifique a sua resposta.

   Resp.: Como foi visto na disciplina de Organização de Computadores, o objetivo da pilha é o de armazenar várias informações referentes ao fluxo de execução das instruções no processador. Dentre estas informações, temos as variáveis locais do procedimento atualmente em execução no processador, e os endereços de retorno das chamadas que foram executadas e que ainda não foram terminadas. Como um processo com múltiplas threads possui múltiplos fluxos de controle, e como cada um deles pode armazenar variáveis locais na pilha e chamar procedimentos, deveremos ter uma pilha para cada thread do processo, que deverá ser salva no contexto desta thread. Note que enquanto o escalonador estiver escolhendo threads do mesmo processo, somente o contexto da thread deve ser salvo, pois todas as threads de um processo compartilham o contexto deste processo. O contexto do processo somente deverá ser salvo quando o escalonador escolher uma thread de

um outro processo, pois esta thread não compartilhará o contexto do processo da thread que estava em execução. Logo, o escalonamento de threads do mesmo processo é mais leve do que o de threads pertencentes a processos diferentes, pois o contexto a ser salvo é menor. Note também que se a pilha não fosse salva no contexto, deveríamos executar as threads do processo seqüencialmente, o que não seria lógico, pois o objetivo de termos múltiplas threads é exatamente o de permitir que elas executem concorrentemente. Perceba que no caso de termos mais de um processador ou de o único processador possuir múltiplos contadores de programa, as threads poderão de fato executar paralelamente.

2. Suponha que tenha ocorrido uma interrupção do disco enquanto o processo A estava em execução no sistema. Na figura a seguir damos as ações que ocorrem quando uma interrupção é tratada pelo sistema operacional. Dê a ordem em que as ações são executadas no tempo.

Resp.: Na figura a seguir damos a ordem correta em que as ações são executadas. Note que o processo tratador do disco, executado devido à primeira chamada ao escalonador, somente será executado após o tratamento da interrupção ser finalizado. Já a segunda chamada ao escalonador fará com que A imediatamente inicie a sua execução no ponto em que parou anteriormente (note que, neste caso, A precisa ser o processo mais prioritário).

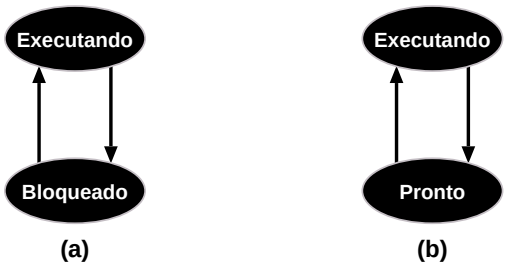


3. Suponha que o computador possua somente um processador, e que os processos A, B, C e D sejam os únicos em execução no computador. Suponha ainda que cada processo precise executar no processador por 2 unidades de tempo. Se o escalonador alternar o processador entre estes processos a cada unidade de tempo, de tal modo que um processo somente volte a executar após todos os outros processos terem executado, quantas unidades de tempo decorrerão entre o início e o término da execução de cada processo? Quais serão os novos tempos decorridos se o computador possuir agora 2 processadores? Esses tempos serão os mesmos se aumentarmos o número de processadores para 4?

   Resp.: -Quando o computador tiver somente um processador, note que um dos processos, após executar por uma unidade de tempo, somente executará novamente no processador depois de três unidades de tempo. Isto ocorrerá porque, neste intervalo de tempo, os três outros processos executarão também por uma unidade de tempo no processador (o que sempre ocorrerá, pois todos os processos precisam executar por duas unidades de tempo no processador). Ao executar novamente no processador por mais uma unidade de tempo, o processo terminará a sua execução, pois já terá executado por duas unidades de tempo. Logo, o tempo decorrido entre o início e o término da execução de cada processo (A, B, C e D) será de cinco unidades de tempo, pois todos os processos precisam executar por duas unidades de tempo no processador.
   -Agora, se o computador tiver dois processadores, dois processos poderão executar em paralelo. Logo, após executar por uma unidade de tempo em um dos processadores, um processo somente poderá executar novamente em um dos processadores após uma unidade de tempo. Isto ocorrerá porque os dois processadores precisarão executar, por uma unidade de tempo, os processos que não foram executados na unidade

de tempo anterior (o que também sempre ocorrerá, pois cada processo ainda precisa executar por duas unidades de tempo). Logo, o tempo decorrido entre o início e o término da execução de cada um dos processos A, B, C e D será de três unidades de tempo, novamente porque todos os processos precisam executar por duas unidades de tempo.
-Neste último caso um processo, após executar por uma unidade de tempo em um dos processadores, poderá executar novamente por uma unidade de tempo (no mesmo processador ou em um outro processador). Isto ocorrerá porque, como o número de processadores do computador é igual ao número de processos, sempre existirão processadores disponíveis para executar os três outros processos. Com isso, o tempo decorrido entre o início e o término da execução de cada um dos processos A, B, C e D será o mesmo tempo que cada um deles precisa executar no processador, isto é, duas unidades de tempo.

4. Um aluno de sistemas operacionais mostrou a seguinte versão estendida do diagrama de transição dos estados de um processo, com dois novos estados: o estado **Novo**, em que o processo é colocado quando é criado, e o estado **Terminado**, em que o processo é colocado quando termina a sua execução. Este diagrama está correto? Justifique a sua resposta.



Resp.: Não, existem diversos erros no diagrama dado na figura. Quando um processo é criado, isto é, está no estado **Novo**, ele deve passar ao estado pronto para esperar pela sua vez de executar no processador. Se o processo criado for crítico e precisar executar imediatamente no processador, o escalonador pode ser chamado logo após a criação deste processo para colocá-lo em execução no processador (mudando o estado do processo para **Executando**). Logo, a transição 0 (assim como a sua descrição) está incorreta, e ela deve ser do estado **Novo** para o estado **Pronto**. A transição do estado **Terminado** para o estado **Pronto** também está incorreta, porque quando o processo termina ele não muda de estado. Além disso, como o processo deve estar no estado

Pronto para terminar a sua execução, deveria existir uma transição do estado **Executando** para o estado **Terminado** (a descrição da transição 5, porém, continua correta). Finalmente, com base no diagrama que vimos na Aula 4, vemos que as descrições das transições 2, 3 e 4 foram trocadas, isto é, a descrição dada na 2 é a da 3, a dada na 3 é a da 4, e a dada na 4 é a da 2. Na figura dada a seguir temos o diagrama correto, com todas as correções descritas anteriormente.