Projeto Simulador de SO

PCS3746 - Sistemas Operacionais

Neste projeto você deve elaborar um programa capaz de simular o gerenciamento de processos e alocação de memória de um sistema operacional. O programa deve ser escrito na linguagem de programação C\C++.

Detalhes do gerenciamento de processos. Neste simulador um processo deve ser representado por um conjunto de instruções de baixo nível (ex. MOV, ADD, JMP, etc.). Essas instruções servirão para indicar em qual etapa de execução o processo está; portanto, não precisam de fato ser executadas (simuladas). É importante que todo processo tenha uma instrução de término (ex. HLT). A tabela abaixo ilustra três processos e suas respectivas instruções de máquina.

Processo 1	Processo 2	Processo 3
PUSH AX	MOV AX, 0	HLT
MOV AX, 10	ADD AX, 4	
HLT	MOV BX, AX	
	HLT	

Para o simulador, o procedimento de criação de um processo consiste basicamente em alocar memória para o processo, definir o identificador único do processo e indicar a primeira instrução que deve ser executada. A instrução “create –m 4” exemplifica o comando passado ao simulador para criação de um novo processo que requer 4 unidades de memórias. Após sua criação, o processo deve ser colocado na fila de prontos caso não possa ser executado imediatamente.

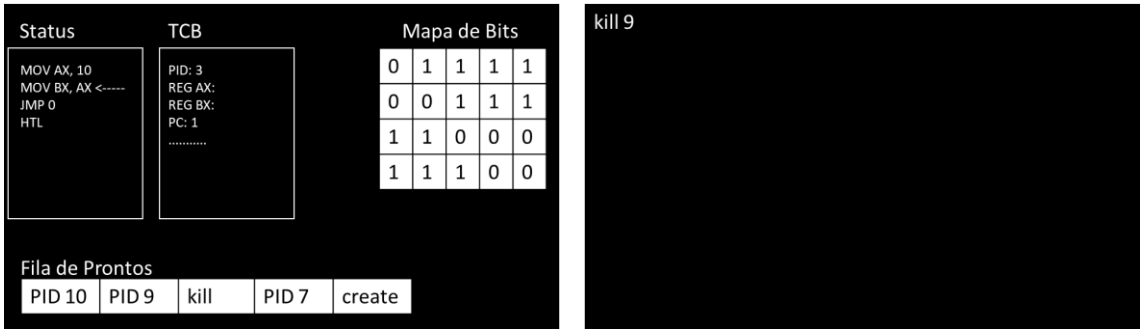
Assim como em um sistema operacional real, seu simulador deve escalonar os processos. Isso significa que um processo pode ter sua execução interrompida a qualquer momento, permitindo que o controle da CPU seja intercalado entre processos de usuários e processos de SO (comandos **create** e **kill**). Para este propósito, você deve simular o trabalho do *dispacher* e a questão da troca de contexto, definindo, por exemplo, quanto tempo um processo pode executar ininterruptamente. O escalonador deve considerar os algoritmos FIFO (não-preemptivo) e Round-Robin (preemptivo); tais configurações podem ser definidas a partir de um arquivo texto ou passados por parâmetro ao disparar o simulador (ex. argv/argc).

Alocação de memória. Neste simulador a memória deve ser representada por unidades de memória. Isso significa que os processos requisitaram unidades de memória independentemente se as unidades representam Bytes, MB, Kb, etc.

Sempre que um novo processo requisitar memória, o sistema operacional deve consultar a estrutura de dados (mapa de bits) que armazena as posições livres (e ocupadas) da memória e

alocar de acordo com o algoritmo *First-Fit*. Analogamente, quando um processo termina sua execução (por exemplo, atinge a instrução HLT ou é terminado pelo SO) suas posições de memória alocadas precisam ser liberadas. Note que a questão de segmentação externa pode ocorrer e precisa ser tratada. Neste simulador a quantidade de memória total disponível é de 20 unidades de memória.

Layout do simulador. É necessário que as opções de gerenciamento (criação e término – create/kill) e a visualização do simulador estejam em terminais (*prompt*) diferentes, como ilustra a figura abaixo.



A visualização do simulador precisa conter as seguintes informações: (i) registradores da CPU (ii) campos na tabela de processo (TCB) (observe que os registradores da CPU precisam também estar na TCB); (iii) mapa de bits e (iv) fila de prontos.

Observações e dicas importantes. Vocês devem se atentar às seguintes questões. (i) Como seu simulador lida com a questão de fragmentação externa? O processo de compactação de memória entraria na fila de prontos? (ii) Como será o controle do quantum (sleep, getchar)? Como ocorre a comunicação entre os dois *prompts*?

As seguintes configurações podem ser predefinidas via arquivo texto ou argumentos passados ao iniciar o simulador: configurações do *dispacher*, posições de memória já alocadas, processos que são criados logo no início do sistema operacional.

Relatório. Todas as decisões de projeto para o funcionamento do simulador devem ser reportadas e entregue juntamente com o(s) código(s) via Moodle. Não esqueça de reportar como você lidou com as questões descritas em "*Observações e dicas importantes*". O formato do relatório deve ser arquivo texto com no máximo 8000 caracteres (com espaço).