Universidade Federal de Mato Grosso do Sul
Faculdade de Computação
Sistemas Operacionais - T01 - Prof. Irineu Sotoma

Descrição do Trabalho T2 – 15 de maio de 2019

1. O Trabalho T2 envolve a simulação de um sistema com paginação sob demanda com alocação global de quadros, algoritmo de substituição de páginas segunda chance, e escalonamento Round-Robin de CPU, com garantia de exclusão mútua.

2. O Trabalho T2 visa resolver o seguinte Problema, assumindo que todas as variáveis e parâmetros indicados são inteiros:

   (a) Assuma que seu sistema terá os recursos de hardware: memória, disco e CPU, ligados por um barramento.

   (b) Assuma que a CPU possua apenas um único núcleo.

   (c) Deve-se desenvolver um sistema que assuma que sempre haverá espaço no disco.

   (d) Assuma que o instante inicial de execução é o instante 0 segundo.

   (e) Há $nframes$ quadros na memória principal, identificados com $idframe$, $0 \leq idframe \leq nframes - 1$. Assuma que os $nframes$ quadros podem ser utilizados pelos processos que precisem ser executados e que as filas e demais estruturas de dados do núcleo do sistema operacional estão em outra memória.

   (f) Seu sistema deverá controlar uma *lista de quadros livres*.

   (g) No estado inicial do sistema, a semente para geração de números pseudoaleatórios é *seed*, todos os $nframes$ quadros estão livres, e todas as entradas das tabelas de páginas dos processos possuem bit de válido/inválido em 0 e bit de referência em 0.

   (h) Há n processos do usuário, identificados com id, $0 \leq id \leq n - 1$.

   (i) O *Criador de processos* cria cada processo id no instante tc_{id} segundos e então "coloca" id na *fila de prontos*.

   (j) Cada processo id precisa de np_{id} páginas, identificadas por $idpage_{id}$, $0 \leq idpage_{id} \leq np_{id} - 1$, e tem tamanho do CPU burst definido como tb_{id} segundos.

   (k) Cada processo id irá acessar as suas páginas de forma aleatória, e por simplicidade nunca irá acessar uma página fora do intervalo de 0 até $np_{id} - 1$.

   (l) O *Escalonador Round-Robin de CPU* irá utilizar o algoritmo Round-Robin de escalonamento de CPU, com time quantum tq (parâmetro de entrada), para escolher e retirar um processo p_k da *fila de prontos* e enviá-lo a um *Despachante*.

   (m) As variáveis mencionadas até este ponto serão definidas pelo usuário do sistema, a partir de parâmetros de entrada via linha de comando.

   (n) Cada processo id possui uma tabela de páginas $pagetable_{id}$.

   (o) O seu sistema não usa TLB.

   (p) Cada entrada de $pagetable_{id}$ possui um $idframe$ do quadro onde a página está na memória, um bit de válido/inválido, e um bit de referência. Neste Trabalho T2 esses bits podem ser implementados como tipo `boolean` ou tipo `byte`.

   (q) O temporizador *Timer* irá avisar o escalonador *Escalonador Round-Robin de CPU* de que o CPU burst do processo p_i, que está na CPU, terminou ou que ele já tenha executado por tq unidades de tempo.

   (r) O *Despachante*, ao receber a indicação de p_k, irá gerar aleatoriamente, a partir de uma semente *seed*, uma referência a página, sorteando uma página $idpage_{p_k}$:

   - Se a página estiver com bit válido, então não haverá falha de página e o bit de referência será definido em 1, e o *Despachante* reinicia o *Timer* e libera a CPU a p_k;
   - Se a página estiver com bit inválido, então haverá falha de página e o *Despachante* solicita que o *Pager* traga $idpage_{p_k}$ à memória e espera até que o *Pager* avise que $idpage_{p_k}$ está na memória, e quando avisado, o *Despachante* reinicia o *Timer* e libera a CPU a p_k.

(s) Na ocorrência de falha de página:

i. Se houver quadro livre, então o *Pager* aloca-o para conter a página $idpage_{p_k}$ solicitada, lê do disco a página solicitada, atualiza a tabela de páginas e a *lista de quadros livres*, e avisa o *Despachante*.

ii. Caso contrário, o *Pager* utiliza o algoritmo de substituição de páginas segunda chance considerando alocação global de quadros para substituir a página de um quadro, lê do disco a página solicitada, e altera devidamente tanto a tabela de páginas referente à página vítima $idpage_{id_{victim}}$ do processo vítima id_{victim} quanto a tabela de páginas referente à página que foi contemplada, e avisa o *Despachante*.

(t) A variável $npagefaults$ contabiliza o número de ocorrências de falhas de página.

(u) Quando um processo id já executou por tb_{id} segundos, ele é retirado da *fila de prontos*, e todos os quadros alocados ao processo id voltam à *lista de quadros livres* e $pagetable_{id}$ é destruída.

(v) Os componentes *Criador de processos*, *Timer*, *Escalonador Round-Robin de CPU*, *Despachante* e *Pager* **devem ser implementados como threads** que se comunicam por memória compartilhada com garantia de exclusão mútua quando for o caso, principalmente quanto ao acesso à *fila de prontos*, e à *lista de quadros livres*.

(w) Durante a simulação do sistema, utilize a hora do sistema como estampa de tempo para mostrar modificação ou ocorrência dos seguintes eventos, em que id é a identificação do processo cuja ação está sendo mostrada, $idpage$ é a identificação da página que está sendo substituída ou trazida do disco, e $idframe$ é o quadro em que está havendo a colocação de uma página ou retirada de uma página vítima:

i. Início da observação

ii. *Criador de processos* criou o processo id e o colocou na *fila de prontos*

iii. *Timer* informa ao *Escalonador Round-Robin de CPU* que o processo id atualmente em execução precisa ser retirado da CPU

iv. *Escalonador Round-Robin de CPU* escolheu o processo id, retirou-o da *fila de prontos* e o encaminhou ao *Despachante*

v. *Despachante* percebe que a página $idpage_{id}$ do processo id está na memória

vi. *Despachante* reiniciou o *Timer* com tq e liberou a CPU ao processo id

vii. *Despachante* percebe que a página $idpage_{id}$ do processo id não está na memória e solicita que o *Pager* traga $idpage_{id}$ à memória

viii. *Pager* percebe que há quadro livre $idframe$

ix. *Pager* lê do disco a página $idpage_{id}$ solicitada e o coloca no quadro livre $idframe$

x. *Pager* percebe que não há quadro livre e substitui a página $idpage_{id_{victim}}$ que está no quadro $idframe$

xi. *Pager* avisa o *Despachante* que o processo id está na memória

xii. *Despachante* é avisado pelo *Pager* que a página $idpage_{id}$ do processo id está no quadro $idframe$

xiii. O número de falhas de página foi $npagefaults$

xiv. Processo id terminou sua execução

xv. Logo após a efetivação dos eventos indicados nos itens 2(w)ix, 2(w)x e 2(w)xiv, deve-se mostrar a *lista de quadros livres* e as tabelas de páginas de todos os processos que ainda não terminaram

xvi. Término da observação

(x) A execução de seu sistema termina quando todos os processos do usuário tiverem terminado.

3. O Trabalho T2 é composto da implementação em Java, C ou C++ sobre a solução para o Problema.

4. Nota:T2 é a nota do Trabalho T2. A seguir, na descrição de cada item que compõe Nota:T2, $\{0, a\}$ significa o valor 0 ou o valor a, e $[0 - a]$ significa algum valor real de 0 até a.

5. Nota:T2 = T2:1 × T2:2 × T2:3 × T2:4 × (T2:5 + T2:6 + T2:7), onde:

T2:1) $\{0, 1\}$: Cada grupo, de **1, 2 ou 3** acadêmicos, deverá desenvolver as implementações em **Java**, **C** ou **C++** em **Linux** sem a geração de erros de compilação e sem geração de exceções durante a execução.

T2:2) $\{0, 1\}$: O código fonte zipado (**.zip**) da solução deverá ser entregue diretamente via "Entrega do Trabalho T2" de "Sistemas Operacionais - T01" em `http://ava.ufms.br`. Um fórum de discussão deste trabalho já se encontra aberto. Você pode entregar o trabalho quantas vezes quiser até às **23 horas** do dia **12 de junho de 2019**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos. Para prevenir imprevistos como falhas de energia, sistema ou internet, recomendamos que a entrega do trabalho seja feita pelo menos um dia antes do prazo.

T2:3) $\{0, 1\}$: O cabeçalho do seu programa Java, C ou C++ deve informar detalhadamente qual é a estrutura de diretórios do seu Trabalho, como compilar e executar seu código via linha de comando do Linux, e quais são os nomes completos dos membros do grupo.

T2:4) $\{0, 1\}$: Atendimento ao item 2v.

T2:5) $[0 - 2]$: Implementação que recebe números inteiros positivos, separados por espaço como parâmetros de entrada via linha de comando, no formato:

$seed \quad nframes \quad n \quad tq$ (lista de dados de cada processo $id\ np\ tc\ tb$).

Um exemplo de valores de parâmetros de entrada é:

345 5 3 2 1 5 5 15 0 7 1 5 2 6 2 10.

T2:6) $[0 - 3]$: Implementação que execute e gere a saída correta conforme o item 2w.

T2:7) $[0 - 5]$: Implementação que trate corretamente os eventos a partir do início da observação.

6. **Caso o professor detecte plágio entre trabalhos, no todo ou em parte, os trabalhos envolvidos terão Nota:T2 = 0.**