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

Descrição CORRIGIDA do Trabalho T1 – 13 de maio de 2019

1. O Trabalho T1 envolve a simulação de um sistema com alocação de armazenamento dinâmico na memória com partições de tamanho variável e algoritmo *first-fit*, com escalonamento FCFS no escalonamento de longo prazo e Round-Robin no escalonamento de CPU e swapping padrão, com garantia de exclusão mútua.
2. O Trabalho T1 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) O tamanho da memória principal (em bytes) é referido como *tmp*. Assuma que todo esse tamanho está disponível aos processos que precisem ser executados e que os processos, as filas e demais estruturas de dados do núcleo do sistema operacional estão em outra memória.
   (f) Há n processos do usuário, identificados com id, $0 \leq id \leq n-1$.
   (g) Cada processo id possui tamanho tp_{id} bytes, tempo de chegada na *Fila de entrada* definido como tc_{id} segundos, e tamanho do CPU burst definido como tb_{id} segundos.
   (h) O *Criador de processos* que criar cada processo id no instante tc_{id} segundos e então irá colocar id em uma *fila de entrada*.
   (i) As variáveis mencionadas até este ponto serão definidas pelo usuário do sistema, a partir de parâmetros de entrada via linha de comando.
   (j) O *Escalonador FCFS de longo prazo* irá utilizar a política FIFO (*First-In-Firt-Out*) para escolher e retirar um processo p_j da *fila de entrada*, para colocá-lo em uma *fila de prontos*, **em duas situações: a) quando houver espaço na memória para colocar p_j, e b) quando um processo terminar seu CPU burst tb_{id}. Em ambos os casos, o *Escalonador FCFS de longo prazo* solicita ao *Swapper* que coloque p_j na memória. Assim que o *Swapper* avisar que p_j está na memória, o *Escalonador FCFS de longo prazo* coloca p_j na *fila de prontos*.**
   (k) **O *Swapper* verifica se há espaço na memória para trazer o p_j solicitado pelo *Escalonador FCFS de longo prazo*:**
      - **Se p_j cabe na memória (usar *first-fit*), então o *Swapper* coloca p_j na memória e avisa o *Escalonador FCFS de longo prazo* que p_j está na memória;**
      - **Caso contrário, o *Swapper* vai removendo os processos da parte inicial da memória, colocando os processos removidos no disco, até que haja espaço suficiente para colocar p_j na memória, coloca p_j na memória, e avisa o *Escalonador FCFS de longo prazo* que p_j está na memória.**
   (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) 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.
   (n) O *Despachante*, ao receber a indicação de p_k, irá verificar se p_k está ou não na memória:
      - Se p_k está na memória, então o *Despachante* reinicia o *Timer* e libera a CPU a p_k;
      - Caso contrário, p_k está no disco e o *Despachante* solicita que o *Swapper* traga p_k à memória e espera até que o *Swapper* avise que p_k está na memória, e quando avisado, o *Despachante* reinicia o *Timer* e libera a CPU a p_k.
   (o) O *Swapper* **também** verifica se há espaço na memória para trazer p_k solicitado pelo *Despachante*:
      - Se p_k cabe na memória (usar *first-fit*), então o *Swapper* coloca p_k na memória e avisa o *Despachante* que p_k está na memória;
      - Caso contrário, o *Swapper* vai removendo os processos da parte inicial da memória, colocando os processos removidos no disco, até que haja espaço suficiente para colocar p_k na memória, coloca p_k na memória, e avisa o *Despachante* que p_k está na memória.

(p) Os componentes *Criador de processos*, *Escalonador FCFS de longo prazo*, *Timer*, *Escalonador Round-Robin de CPU*, *Despachante* e *Swapper* **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 entrada*, à *fila de prontos*, e à alocação de memória.

(q) 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:

i. Início da observação
ii. *Criador de processos* criou o processo *id* e o colocou na *fila de entrada*
iii. *Escalonador FCFS de longo prazo* escolheu **e retirou** o processo *id* da ***fila de entrada***
iv. *Escalonador FCFS de longo prazo* **solicitou que o *Swapper* traga** *id* **à memória**
v. *Escalonador FCFS de longo prazo* **colocou** *id* **na *fila de prontos***
vi. *Timer* informa ao *Escalonador Round-Robin de CPU* que o processo *id* atualmente em execução precisa ser retirado da CPU
vii. *Escalonador Round-Robin de CPU* escolheu o processo *id*, retirou-o da *fila de prontos* e o encaminhou ao *Despachante*
viii. *Despachante* percebe que o processo *id* está na memória
ix. *Despachante* reiniciou o *Timer* com *tq* e liberou a CPU ao processo *id*
x. *Despachante* percebe que o processo *id* está no disco e solicita que o *Swapper* traga *id* à memória
xi. *Swapper* percebe que há espaço no processo *id* na memória
xii. *Swapper* traz o processo *id* do disco e o coloca na memória
xiii. *Swapper* percebe que não há espaço ao processo *id* na memória
xiv. *Swapper* retirou o processo *id* para liberar espaço na memória, e o enviou ao disco
xv. ***Swapper* avisa o *Escalonador FCFS de longo prazo* que o processo** *id* **está na memória**
xvi. *Swapper* avisa o *Despachante* que o processo *id* está na memória
xvii. *Despachante* é avisado pelo *Swapper* que o processo *id* está na memória
xviii. Processo *id* terminou sua execução
xix. Término da observação

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

3. O Trabalho T1 é composto da implementação em Java, C ou C++ sobre a solução para o Problema.
4. Nota:T1 é a nota do Trabalho T1. A seguir, na descrição de cada item que compõe Nota:T1, $\{0, a\}$ significa o valor 0 ou o valor a, e $[0 - a]$ significa algum valor real de 0 até a.
5. Nota:T1 = T1:1 × T1:2 × T1:3 × T1:4 × (T1:5 + T1:6 + T1:7), onde:

T1:1) $\{0, 1\}$: 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.

T1:2) $\{0, 1\}$: O código fonte zipado (**.zip**) da solução deverá ser entregue diretamente via "Entrega do Trabalho T1" 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 **27 de maio 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.

T1:3) $\{0, 1\}$: 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.

T1:4) $\{0, 1\}$: Atendimento ao item 2p.

T1:5) $[0 - 2]$: Implementação que recebe números inteiros positivos, separados por espaço como parâmetros de entrada via linha de comando, no formato:

tmp n tq (lista de dados de cada processo *id tp tc tb*).

Um exemplo de valores de parâmetros de entrada é:

1000 3 2 1 500 5 15 0 700 1 5 2 600 2 10.

T1:6) $[0 - 3]$: Implementação que execute e gere a saída correta conforme o item 2q.

T1:7) $[0 - 5]$: 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:T1 = 0.**