MAC0422 – Sistemas Operacionais – 1s2025
EP3 (Individual)
Data de entrega: 10/6/2025 até 13:00:00

Prof. Daniel Macêdo Batista

1 Problema

A tarefa neste EP é implementar um simulador de gerência de memória com diversos algoritmos para alocação de memória. A simulação será feita manipulando arquivos PGM que representarão uma memória fictícia com 65536 unidades de alocação. Cada pixel desse arquivo representará uma unidade de alocação. Unidades que estejam sendo utilizadas por algum processo devem ser representadas por pixels de cor preta e unidades disponíveis devem ser representadas por pixels de cor branca. Todo o código deve ser escrito em C para ser executado no GNU/Linux.

2 Requisitos

O código deve utilizar apenas as funções das bibliotecas que estão presentes em uma instalação padrão do compilador `gcc`. Programas escritos em outra linguagem ou utilizando alguma biblioteca extra para gerenciar a memória simulada ou para manipular os arquivos PGM terão nota ZERO. Toda a manipulação do arquivo PGM deve ser feita diretamente no arquivo. Não é permitido copiar o conteúdo inteiro do arquivo para a memória, fazer todas as mudanças na memória, e depois salvar no arquivo novo. O código deve iniciar fazendo uma cópia do arquivo de entrada para um novo arquivo de saída (Essa cópia pode ser feita de qualquer forma sem as restrições de utilização da memória), fechar o arquivo de entrada e passar a trabalhar apenas no arquivo de saída. A ideia com essa restrição é de fato simular o funcionamento dos algoritmos, que precisam manipular de fato a memória, sem ter um "buffer" para ajudar.

Arquivos PGM têm a vantagem de terem seu conteúdo em texto puro, o que facilita a sua manipulação. A desvantagem óbvia é que os arquivos ficam maiores do que se fossem armazenados em algum formato de imagem que use mecanismos de compressão, mas isso foge do escopo do propósito deste EP. Vamos considerar que o código vai sempre manipular uma memória com 65536 unidades de alocação e que essa memória será representada por um arquivo PGM com 256 linhas por 256 colunas. O arquivo PGM sempre terá o seguinte cabeçalho:

```
P2
256 256
255
```

Em seguida o conteúdo das unidades de alocação será informado em linhas de 63 caracteres com cada unidade de alocação ocupando exatamente 3 caracteres e com um espaço em branco entre as unidades. Ou seja, cada linha do arquivo armazenará o status de 16 unidades de alocação e o arquivo terá um total

de 4096 linhas. A última unidade de alocação de uma linha terminará com uma quebra de linha, sem o espaço em branco após ela. A primeira unidade de alocação de uma linha iniciará sem um espaço em branco antes. Pixels pretos serão representados pelo valor 0 com dois espaços em branco antes e pixels brancos serão representados pelo valor 255. Abaixo estão as cinco primeiras linhas do arquivo `ep3-exemplo01.pgm` compartilhado na área do EP3 no e-Disciplinas:

```
P2
256 256
255
  0   0   0   0   0   0   0   0   0   0   0   0 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
```

Ou seja, nesse trecho estão exibidas as primeiras 32 unidades de alocação da memória que esse arquivo está representando. As 12 primeiras unidades de alocação estão ocupadas (valor 0 representa pixel preto) e as 20 unidades de alocação seguintes estão livres (valor 255 representa pixel branco). Note que o fato de haver um ENTER após a décima ~~segunda~~ sexta unidade de alocação não significa que a imagem terá ~~12~~ 16 linhas! A imagem terá 256 linhas e 256 colunas conforme informado na segunda linha do arquivo ("256 256").

Neste EP o simulador deve implementar os seguintes algoritmos:

1. *First Fit*
2. *Next Fit*
3. *Best Fit*
4. *Worst Fit*

Caso algum algoritmo precise manter em memória algum valor para que ele funcione corretamente (por exemplo, o *Next Fit* precisa guardar qual foi a última posição alocada), você pode usar uma variável para isso no seu código. A restrição de não usar a memória do computador diz respeito apenas à manipulação direta do arquivo PGM.

Para que as alocações sejam realizadas, além de um arquivo PGM com o estado atual da memória, é necessário passar como entrada para o simulador um arquivo de trace que informe o número da linha e a quantidade de unidades de alocação sendo requisitadas naquela linha. Considere que esse arquivo de trace terá várias linhas com o seguinte formato:

```
l m
```

Onde `l` é o número da linha na qual a alocação é requisitada e `m` é a quantidade de unidades de alocação sendo requisitadas. Ambos os valores são números naturais sendo que $1 \leq \texttt{l} \leq 2000$ e $1 \leq \texttt{m} \leq 256$. A numeração das linhas de fato é a numeração correta. Não é para ser utilizado um valor aleatório que não siga a sequência correta dos números das linhas.

É possível ainda que apareçam linhas como esta no arquivo:

```
l COMPACTAR
```

Onde `l` é o número da linha e "`COMPACTAR`" de fato é esta string. Esta linha indica que na linha `l`, toda a memória deve ser compactada.

A invocação do simulador no shell deve receber como argumentos de linha de comando, nesta ordem:

```
n <nome do arquivo PGM de entrada> <nome do arquivo de trace> <nome do arquivo PGM de saída>
```

Onde n é a numeração que corresponde ao algoritmo a ser executado conforme listado anteriormente neste enunciado.

O simulador deve finalizar sua execução assim que todos as alocações requisitadas no arquivo de trace forem processadas. Note que não necessariamente todas as alocações poderão ser realizadas. Aquelas alocações que não puderem ser atendidas precisarão ser registradas pelo simulador. As alocações devem ser feitas de forma sequencial. Não é para ser usado paralelismo.

Além de escrever um arquivo PGM de saída com o status final da memória, o simulador precisa imprimir na saída padrão as linhas inteiras das alocações que não puderam ser realizadas (O l e o m) e uma última linha com a quantidade de alocações que não puderam ser realizadas. Essa última linha sempre precisa ser impressa, mesmo que todas as alocações sejam realizadas. Nesse caso de todas as alocações conseguirem ser realizadas, a única informação impressa na saída padrão deve ser:

0

3 Sobre a entrega

Deve ser entregue um arquivo `.tar.gz` contendo os itens listados abaixo. EPs que não contenham **todos** os itens abaixo **exatamente como pedido em termos de formato e de nomes** terão nota ZERO e não serão corrigidos. **A depender da qualidade do conteúdo entregue**, mesmo que o EP seja entregue, **ele pode ser considerado como não entregue, o que mudará o cálculo da média final**:

- 1 único arquivo `ep3.c` e 1 único arquivo `ep3.h` com a implementação do simulador;
- 1 único arquivo `LEIAME` em formato texto puro explicando como compilar e executar o simulador;
- 1 único arquivo `Makefile` para gerar o executável;
- 1 único arquivo `slides-ep3.pdf` com no máximo 12 slides (contando todos os slides, inclusive capa, roteiro e referências, se houverem) explicando quais foram as variáveis auxiliares que cada algoritmo utilizou e resumindo os resultados obtidos com diversos experimentos que mostrem cenários em que esses algoritmos se comportaram como esperado ou não a depender do arquivo de trace usado com o arquivo `ep3-exemplo01.pgm`. **Esses slides não serão apresentados. Eles devem ser preparados supondo que você teria que apresentá-los.** Não coloque nos slides conteúdo que não foi pedido. Por exemplo, não precisa repetir o enunciado e nem explicar o formato do arquivo PGM;
- quatro arquivos de trace ~~`trace-firsfit`~~`trace-firstfit`, `trace-nextfit`, `trace-bestfit` e `trace-worstfit` criados de modo a mostrar vantagens de cada um dos quatro algoritmos de acordo com o nome de cada arquivo.

Os resultados devem ser exibidos com gráficos que facilitem observar os pontos positivos e negativos de cada algoritmo usando como base cada um dos arquivos criados. Discuta se os resultados saíram conforme o esperado considerando os objetivos de cada algoritmo (você pode usar trechos dos arquivos PGM nos slides para mostrar o efeito de cada algoritmo se você achar necessário). Note que para que bons resultados sejam obtidos, bons arquivos de trace devem ser criados. Compreender os algoritmos de escalonamento é essencial para que sejam bolados bons arquivos de trace capazes de ressaltar as

características de cada algoritmo. Os gráficos e a discussão dos resultados dos experimentos valem 3,0 pontos.

O desempacotamento do arquivo `.tar.gz` deve produzir um diretório contendo os itens. O nome do diretório deve ser `ep3-seu_nome`. Por exemplo: `ep3-mark_dean`. Entregas que sejam tarbombs ou que, quando descompactadas, gerem o diretório com o nome errado perderão ~~3~~1,0 pontos.

A entrega do `.tar.gz` deve ser feita através do e-Disciplinas.

O EP deve ser feito individualmente.

Obs.: não inclua no `.tar.gz` itens que não foram pedidos neste enunciado, como por exemplo, dotdirs como o `.git`, dotfiles como o `.gitignore`, saídas para diversas execuções, arquivos pré-compilados, etc.... A presença de conteúdos não solicitados no `.tar.gz` levarão a um desconto de ~~3~~1,0 na nota final.