MAC0422 – Sistemas Operacionais – 2s2020
PRec (INDIVIDUAL)
Data de entrega: 7/1/2021 até 8:00 da manhã

Prof. Daniel Macêdo Batista

1 Problema

A tarefa nesta PRec é implementar e avaliar quatro algoritmos de alocação de memória. O programa pode ser escrito em qualquer linguagem de programação desde que haja compilador ou interpretador livre e gratuito para GNU/Linux. A interação com o usuário deve ser feita inteiramente no shell, sem interface gráfica, recebendo parâmetros na linha de comando, lendo conteúdo de arquivos e salvando conteúdo em arquivos.

A memória simulada pelo programa estará representada em um arquivo PGM em escala de cinza em que cada pixel do arquivo representa 1 unidade de alocação da memória. Nesse arquivo, 1 pixel com valor 0 (preto) significa 1 unidade de alocação ocupada e 1 pixel com valor 255 (branco) significa 1 unidade de alocação livre. Para entender o formato de um arquivo PGM leia:
`https://www.ime.usp.br/~mac2166/ep3-2020/pgm_ppm.pdf`

Os quatro algoritmos a serem implementados são:

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

2 Interação com o programa

O programa deve ser executado no shell com quatro parâmetros obrigatórios na seguinte ordem:

```
<algoritmo> <arquivo PGM inicial> <arquivo de requisições> <arquivo PGM final>
```

Onde:

- **algoritmo**: Número de 1 a 4 de acordo com a lista dos algoritmos apresentada na Seção anterior;
- **arquivo PGM inicial**: Arquivo PGM de acordo com o explicado na Seção anterior representando o estado atual da memória;

- **arquivo de requisições**: Arquivo texto em que cada linha contém 1 inteiro representando a quantidade de unidades de alocação requisitadas na ordem em que as requisições forem feitas;
- **arquivo PGM final**: Arquivo PGM de acordo com o explicado na Seção anterior representando o estado da memória após o algoritmo selecionado ter atendido (ou tentado atender) as alocações requisitadas no arquivo de requisições.

No e-Disciplinas, no material do Encontro 19, há um .tar.gz com exemplos de arquivos PGM e de arquivos de requisições que podem ser usados para testar seu programa, mas não se restrinja a esses arquivos. Crie outros arquivos.

Abaixo segue um exemplo de invocação do programa para executar o algoritmo Worst Fit. O programa foi compilado no binário `prec`, no diretório atual, o arquivo PGM inicial se chama `inicial.pgm`, o arquivo com as requisições de memória se chama `trace.txt` e o arquivo a ser salvo após as alocações se chama `final.pgm`. Todos os arquivos estão no diretório atual:

```
./prec 4 inicial.pgm trace.txt final.pgm
```

3 Requisitos

A memória a ser simulada deve conter 698896 unidades de alocação. O arquivo PGM equivalente deve ter uma resolução de 836×836 pixels.

Caso o algoritmo informado consiga fazer todas as alocações requisitadas pelo arquivo de requisições, nada deve ser impresso na tela. Apenas o arquivo PGM de saída deve ser criado considerando todas as alocações de memória que foram feitas.

Caso o algoritmo informado não consiga fazer todas as alocações requisitadas pelo arquivo de requisições, deve ser impresso na tela uma linha como um número inteiro representando a quantidade de requisições que foram feitas até chegar naquela que não foi possível. Nesse caso, o arquivo PGM de saída deve ser criado considerando todas as alocações de memória que conseguiram ser feitas. Nenhuma requisição a partir daquela que não foi atendida deve ser realizada.

IMPORTANTE: Não implemente nenhum mecanismo de compactação de memória.

Por exemplo, se o arquivo de requisições possuir apenas 1 linha com o número 700000, independente do conteúdo inicial da memória e do algoritmo selecionado, essa alocação não será possível. Nesse caso a saída do seu programa deve ser apenas uma linha com o número 0 impressa na tela. Além disso, o arquivo PGM de saída será igual ao arquivo PGM de entrada já que nenhuma alocação terá sido feita.

Todos os quatro algoritmos de alocação de memória devem ser implementados do zero. Programas que utilizem bibliotecas ou similares que já implementem qualquer um dos algoritmos terão nota ZERO

4 Experimentos

Após finalizada a implementação do seu programa você deverá realizar experimentos que meçam o tempo médio (média de 30 execuções com intervalo de confiança e nível de confiança de 95%) dos quatro algoritmos nas seguintes situações (serão três arquivos de requisições diferentes):

1. 10000 requisições que somem 12,5% (87362 unidades de alocação) da memória;
2. 10000 requisições que somem 50% (349448 unidades de alocação) da memória;

3. 10000 requisições que somem 75% (524172 unidades de alocação) da memória;

Com a memória em 3 estados iniciais diferentes (serão três arquivos PGM de entrada diferentes):

1. Memória vazia
2. Memória 50% ocupada no início da mesma (unidades de alocação 0 a 349447)
3. Memória 50% ocupada aleatoriamente

5 Sobre a entrega

Deve ser entregue um arquivo .tar.gz contendo os itens listados abaixo. Entregas que não contenham **todos** os itens abaixo **exatamente como pedido** terão nota ZERO e não serão corrigidos.

- código-fonte;
- arquivo LEIAME **em formato texto puro** explicando como compilar e executar o programa;
- Makefile ou similar para facilitar a compilação do código-fonte e a geração do binário do programa;
- apresentação **em .pdf** para ser apresentada em no máximo 15 minutos resumindo os resultados obtidos nos experimentos (use gráficos!). Informações sobre o hardware e o SO do computador usado nos experimentos também devem ser fornecidas. Outras informações que forem julgadas como importantes, principalmente relacionadas com decisões de projeto que não ficaram especificadas no enunciado também podem ser apresentadas nos slides. **Esses slides não serão apresentados. Eles devem ser preparados supondo que você teria que apresentá-los.**

O desempacotamento do arquivo .tar.gz deve produzir um diretório contendo os itens. O nome do diretório deve ser prec-nome. Por exemplo: prec-joao. Entregas que não gerem um diretório ou que gerem o diretório com o nome errado perderão 2,0 pontos.

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

A PRec deve ser feita **individualmente**.

Obs.: não inclua no .tar.gz itens que não foram pedidos neste enunciado. Relatórios, saídas para diversas execuções e entradas usadas para testar o programa não devem ser entregues. No máximo um resumo sobre as entradas usadas pode ser colocado na apresentação, caso isso não ocupe muito espaço.