

## EP2 - MAC0422 - 2015

### main, mem\_mgr e page\_mgr

Renato Lui Geh e Ricardo Fonseca

# main

Escrever coisas

# Gerenciador de Memória

Para facilitar o trabalho da implementação do nosso Gerenciador de Memória, criamos dois arquivos: `utils.c` e `utils.h`. Nesses arquivos foram criados as structs para representar `mem_node`, `size_node`, funções para auxiliar algoritmos e escrever bytes no arquivo de saída.

Listas duplamente ligadas com cabeça foram criadas para armazenar e manipular os blocos de memória mais facilmente. Para o algoritmo Quick Fit usamos uma lista de listas para separar os diferentes tamanhos de espaços livres na memória.

## mem\_node e size\_node

Para manipular a memória virtual, utilizamos duas structs diferentes, uma representando um bloco na memória virtual e outro que é apenas usado para o algoritmo Quick Fit

**mem\_node:** Possui 6 campos: (`char t`) para o tipo do bloco, (`int i`) para a posição início da memória do bloco, (`int s`) para tamanho do bloco, (`mem_node *n`) e (`mem_node *p`) como ponteiros para os próximos nós da lista.

**size\_node:** Lista de listas de tamanhos de blocos para o algoritmo Quick Fit. Possui 4 campos. (`int s`) para tamanho dos blocos da lista que o nó aponta, (`mem_node *f`) como ponteiro para uma lista ligada sem cabeça com os blocos livres de tamanho `s`, (`size_node *n`) e (`size_node *p`) como ponteiros para os próximos nós da lista.

# Algoritmos

Para ser mais fácil de se escolher qual gerenciador usar, criamos a variável `manager`, que é um ponteiro para função. Os argumentos de linha de comando são lidos e a função é atribuída a `manager` em seguida.

O gerenciador pelo método Quick Fit utiliza uma lista de listas, que é criado apenas no caso dele ser escolhido ao rodar o programa.

# Listas Ligadas

Em todos os gerenciadores implementados foram usadas a lista com a cabeça `v_mem_h`. Para a memória física total utiliza-se `t_mem_h`

`v_mem_h`: guarda todos os blocos de memória livres ou ocupados na memória virtual. O algoritmo Quick Fit utiliza de forma um pouco diferente essa lista.

`t_mem_h`: guarda todos os blocos de memória ocupando a memória física total.

# First Fit (FF)

Este gerenciador usa diretamente a lista da memória virtual. Quando o  $t\_secs$  chega no instante  $t_0$  de um processo, se há algum espaço livre que ele caiba, o processo é atribuído àquela parte da memória virtual. O bloco de memória é mudado para P.

O processo permanece na memória até terminar.

## Next Fit (NF)

Este gerenciador também usa diretamente a lista da memória virtual. Quando o `t_secs` chega no instante `t0` de um processo, se há algum espaço livre que caiba o processo, o processo é atribuído àquela parte da memória virtual. Porém usamos um apontador de nó especial `v_last` que marca a última posição vista na lista, dessa forma para futuros usos da função, ela começa do último bloco criado.



# Quick Fit (QF)

Ao escolher o algoritmo Quick Fit, o dicionário é criado dinamicamente por nosso algoritmo, que faz o seguinte:

- Pega o valor disponível (inicialmente toda a memória virtual disponível).
- Divide em 4 esse espaço, e aproxima até o maior múltiplo de 2 menor que esse valor.
- Se o resultado for maior que o limite inferior (definido por nós como 16, de acordo com o tamanho das páginas), separa  $3/4$  do tamanho total recebido pela função como espaço livre nas listas (para cada  $1/3$  é criado um bloco de espaço livre desse tamanho próximo de  $1/4$  do tamanho total). Em seguida chama a função recursivamente para o restante do espaço.
- Caso o resultado seja igual ou menor que o limite inferior, separa o maior número de intervalos com tamanho limite inferior como espaço livre na lista do quick fit.

## QF - Continuação

Quando o `t_secs` chega no instante `t0` de um processo, ele procura o menor bloco livre que seja maior ou igual que o tamanho necessário do processo (percorre a lista de `size_node`), e depois marca como ocupado esse bloco, e põe no começo da lista `v_mem_h` esse processo (essa lista ligada não possui ordenação para processos, apenas tem todos os blocos que estão sendo ocupados por processos).

# ff\_nf\_free

Tanto para o gerenciador de memória FF quanto NF utilizamos a mesma função para quando um processo termina. De forma simples ela transforma o bloco como livre, e com a ajuda das listas duplamente ligadas, verifica se o bloco estava entre algum outro bloco livre, eficientemente unindo-os em um só bloco pronto para ocupar outro processo.

# qf\_free

Para o gerenciador de memória QF utilizamos a função `qf_free` para quando um processo termina. Ela transforma o bloco do processo como livre, e utilizando nossa lista de listas, disponibiliza como livre o bloco na sua respectiva lista (a que possui outros blocos com seu mesmo tamanho).

# Paginação

Escrever coisas

# Not Recently Used Page (NRU)

Coisas da vida

# First-In, First-Out (FIFO)

Coisas do universo

## Second-Chance Page (SC)

Coisas ddo amor



# Least Recently Used Page (LRU)

Where is the Love

# Testes

Veremos alguns testes feitos com os gerenciadores de memória e algoritmos de substituição de página.

# Observações

Nosso algoritmo Quick Fit divide a memória virtual até tamanhos maiores ou iguais que o limite inferior, o que gera um problema de desperdício de memória, pois se a memória não for múltiplo dele (16 no caso), é possível que até 15 bytes sejam desperdiçados e nunca utilizados, mas comparado com a memória virtual disponível não é de grande impacto esse valor.