MAC0422-EP3
Sistemas de Arquivos

Sistemas Operacionais

Fabio Muller (Nº USP: 8536127) - Daniel Jorge (Nº USP: 8531845)

Agenda

- Linguagem
- Organização dos Arquivos
- ep3.py
- estruturas.py
- sistema.py
- util.py
- Resultados

Linguagem

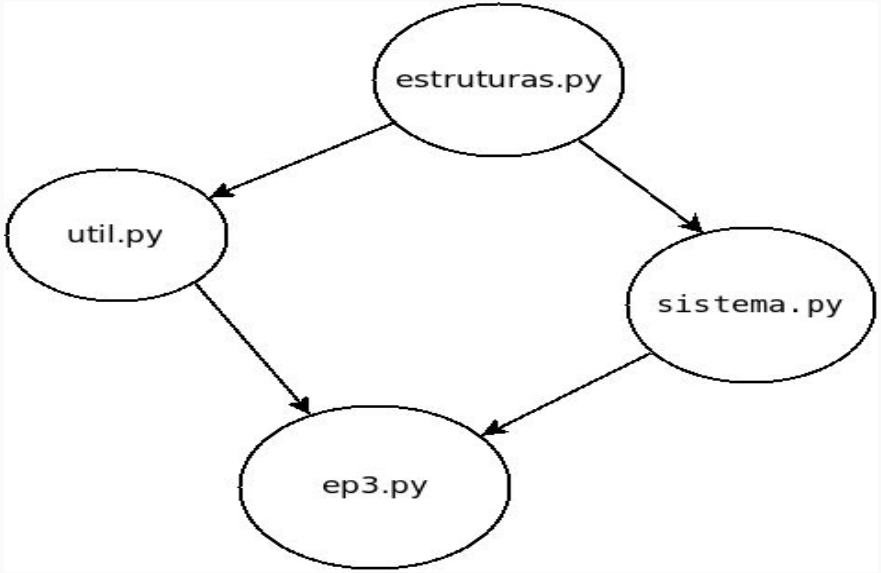
Python:

Ótima para prototipagem

Fácil de tratar entrada e saída e texto

Orientada a objetos

Organização



ep3.py

Módulo principal com While (True)

Lê entrada do usuário a cada iteração

Usa outros módulos para executar os comandos

Usa módulos da biblioteca padrão para saber o tempo (**time**), ver se arquivo existe (**os.path**), cálculos (**math**)

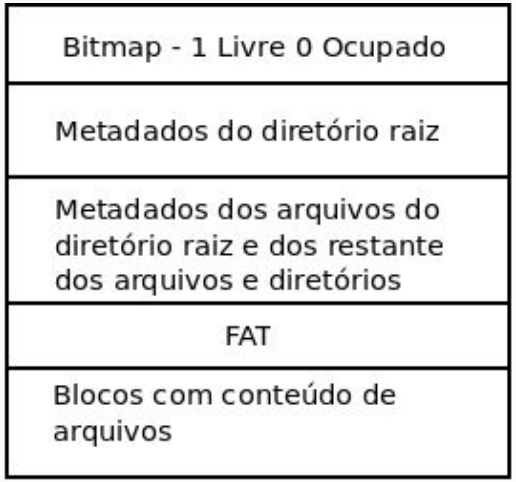
estruturas.py

Módulo com as estruturas usadas (Arquivo e Diretorio):

- Arquivo: Classe que representa um arquivo regular
  - Atributos: nome, tamanho, instantes criado, modificado, acessado, bloco início
  - Métodos: conteúdo, metadados
- Diretório: Classe que representa um diretório
  - Atributos: nome, caminho, instantes criado,acessado, lista de arquivos/subdiretórios
  - Métodos: espaço desperdiçado, número de arquivos/subdiretórios, find, metadados

sistema.py

Composição do sistema de arquivos:



sistema.py

Módulo com operações para o sistema de arquivo

- SistemaArquivos: Classe que representa um sistema de arquivos
  - Atributos: nome, bitmap, diretório raiz, FAT, blocos com conteúdo
  - Métodos: espaço ocupado, espaço ocupado por metadados

Outras funções do módulo:

- Ler um sistema de arquivo já existente para montar ele
- Gravar no disco um sistema de arquivos novo ou um existente modificado

sistema.py

- Obs.: o bitmap é compactado para ser guardado no arquivo
- Função que descompacta o bitmap do arquivo e devolve lista
- Função que recebe lista e devolve bitmap compactado
- Função que recebe lista que representa a FAT atual e guarda no formato para ser guardado no sistema de arquivo

util.py

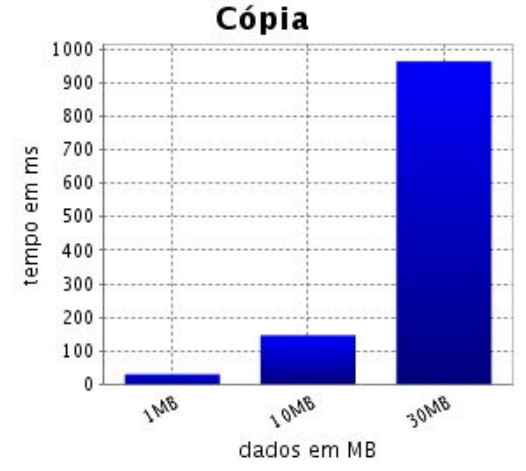
Módulo com funções utilizadas para executar outros comandos

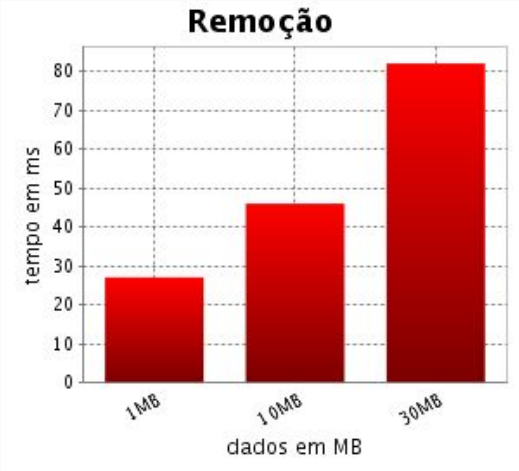
- rm: remove os ponteiros da FAT e marca no bitmap como livres
- rmdir: se vazio remove diretório, senão remove arquivos e subdiretórios dentro antes
- Verifica se a adição de um novo arquivo extrapola o limite de 100 MB
- Coloca o arquivo no sistemas de arquivo sabendo que o limite será respeitado

Resultados

Obs.: Testes rodados no Ubuntu15 com Intel Core i-5 e 8 GB de RAM

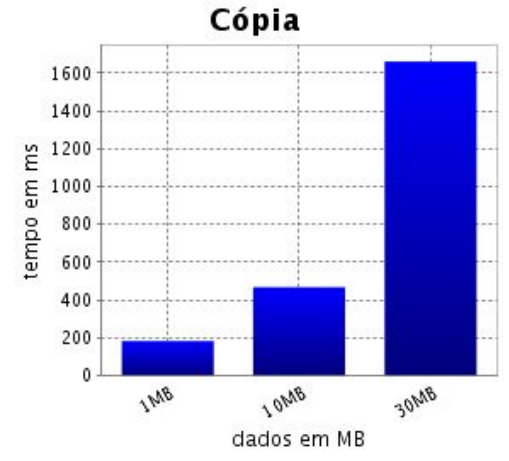
Para sistema de arquivos vazio:





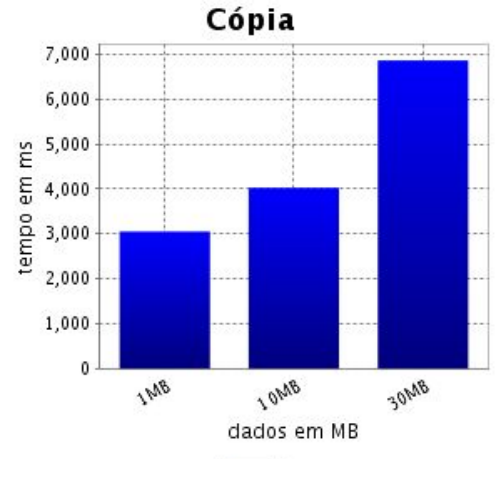
Resultados

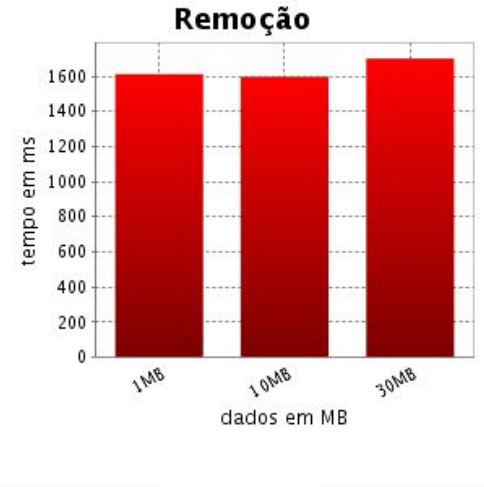
Para sistema de arquivos com 10 MB ocupados:



Resultados

Para sistema de arquivos com 50 MB ocupados:





Resultados

Intervalos de confiança de 95% para 30 execuções (as médias estão nos gráficos; valores em segundos):

Com o sistema vazio:

Cópia: 1 MB - [0.028, 0.033], 10 MB - [0.135, 0.164], 30 MB - [0.943, 0.981]

Remoção: 1 MB - [0.023, 0.031], 10 MB - [0.045, 0.047], 30 MB - [0.077, 0.085]

Remoção diretório: vazios - [0.029, 0.035], 200 arquivos/subdir - [1.851, 1.042]

Resultados

Intervalos de confiança de 95% para 30 execuções (as médias estão nos gráficos; valores em segundos):

Com sistema com 10 MB ocupados:

Cópia: 1 MB - [0.172, 0.193], 10 MB - [0.432, 0.495], 30 MB - [1.623, 1.672]

Remoção: 1 MB - [0.097, 0.112], 10 MB - [0.123, 0.128], 30 MB - [0.157, 0.162]

Remoção diretório: vazios - [0.092, 0.112], 200 arquivos/subdir - [2.112, 2.231]

Resultados

Intervalos de confiança de 95% para 30 execuções (as médias estão nos gráficos; valores em segundos):

Com sistema com 50 MB ocupados:

Cópia: 1 MB - [3.041, 3.062], 10 MB - [4.007, 4.045], 30 MB - [6.712, 6.812]

Remoção: 1 MB - [1.592, 1.632], 10MB - [1.529, 1.612], 30 MB - [1.698, 1.711]

Remoção diretório: vazios - [1.512, 1.621], 200 arquivos/subdir - [5.642, 6.123]