**Relatório Impresso – Implementação SO**

Pedro Yan Ornelas de Oliveira – 14/0158995

Pietro Bertarini Motta – 14/0159118

Vitor Satoru Machi Matsumine – 14/0033262

1. **Descrição das ferramentas/linguagens utilizadas**

Todo (Pietro)

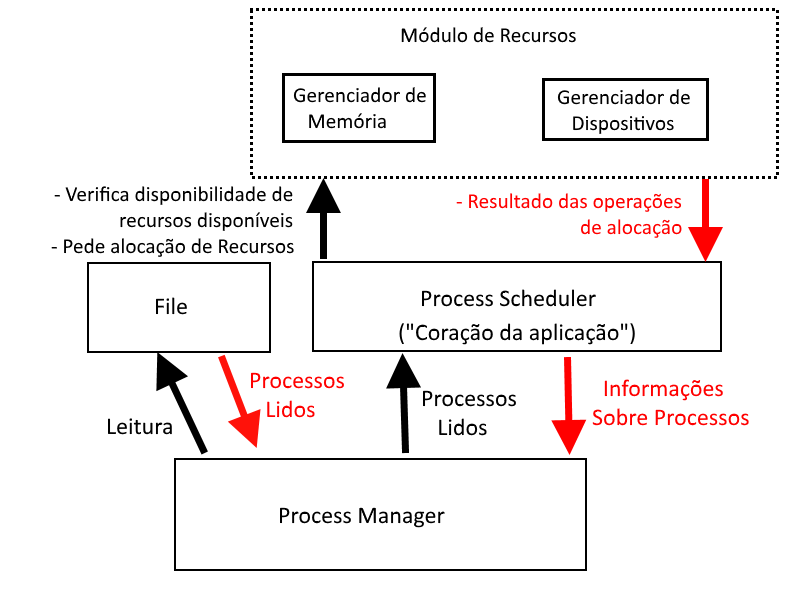
1. **Descrição Teórica e Prática da solução dada**

Para resolver o problema pedido, foi desenvolvido um sistema dividido em dois grandes módulos:

* 1. **Process Manager:** módulo responsável por todas as operações de processo;
  2. **File Manager:** módulo responsável por todas as operações de arquivos.

A solução desenvolvida para o Process Manager segue a seguinte arquitetura:

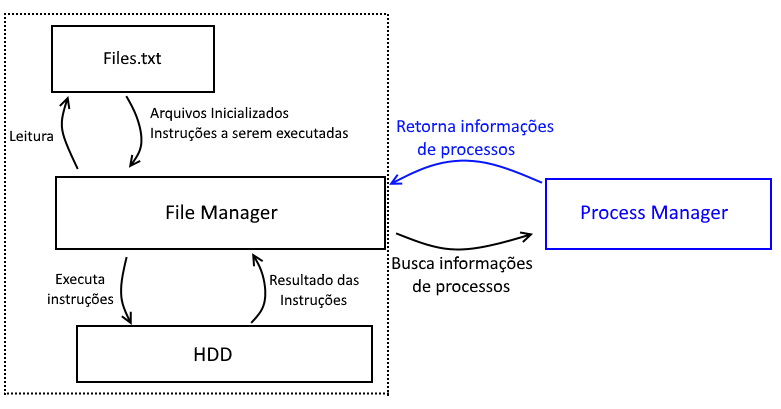
**Arquitetura do Process Manager**



* **Process Manager:**
  + **Função:** recebe como entrada um arquivo contendo informações dos processos e realiza uma simulação da execução dos mesmos até que todos os processos especificados no arquivo tenham sido executados por completo. Fornece também uma interface de comunicação com módulos externos que precisam buscar informações sobre os processos.
  + **Equivale aos seguintes módulos da especificação:**
    - **Módulo de Processos** – contém as estruturas de dados utilizadas para abstrair o funcionamento de um processo.
    - **Módulo de Leitura (módulo extra) –** realiza a leitura do arquivo de processos, validando-o e convertendo as linhas para as estruturas de dados definidas no módulo de processos.
* **Process Scheduler:**
  + **Função:** funciona como o “Coração” do programa. É responsável por montar a abstração das estruturas de fila e escalonar quais processos devem ser executados em determinado momento, levando em conta as prioridades de cada processo e os recursos disponíveis para uso. [Pietro: Explicar algoritmo utilizado para evitar starvation]
  + **Equivale ao módulo de filas da especificação.**
* **Gerenciador de Memória**
  + **Função:** responsável por verificar a disponibilidade de memória, tanto para processos de tempo real quanto para os demais processos, alocar blocos contíguos de memória utilizando o algoritmo First-Fit e liberar a memória quando um processo encerrar.
  + **Equivale ao módulo de memória da especificação.**
* **Gerenciador de Dispositivos**
  + **Função:** responsável por gerenciar todos os dispositivos de E/S, realizando alocação e liberação desses recursos.
  + **Equivale ao módulo de recurso da especificação.**
  + **Obs.:** no algoritmo construído, tanto a memória quanto os dispositivos foram considerados como **recursos do sistema**, mas isso apenas no âmbito da nomenclatura. O funcionamento de cada módulo **permanece em conformidade com a especificação.**

A solução desenvolvida para o File Manager, por sua vez, segue a seguinte arquitetura:

**Arquitetura do File Manager**

****

* **File Manager**
  + **Função:** recebe como entrada um arquivo contendo o estado inicial do disco e as instruções à serem executadas pelo sistema de arquivos. O disco é inicializado com tais dados, para que logo em seguida realize a execução das instruções.

O módulo também é responsável pela verificação de erros na sintaxe (definidas pela especificação), verificação de erros lógicos nos arquivos de inicialização (como a inicialização de dois arquivos que ocupam os mesmos setores) e a realização de operações no disco (por meio da busca de dados dos processos e subsequente verificação de permissões para tais operações por meio de uma interface de comunicação com o Gerenciador de Processos).

* + **Equivale aos seguintes módulos da especificação:**
    - **Módulo de Arquivos –** responsável por operações de Create e Delete.
* **HDD**
  + **Função:** abstrair as funcionalidades de um disco rígido conforme a especificação.

1. **Principais Dificuldades e suas soluções**
2. **Comunicação entre módulos trabalhando em paralelo**

Conseguir dividir o problema em módulos bem definidos com responsabilidades claras, conceito conhecido como *Separation of Concerns)* é um dos principais pilares que visam uma boa qualidade de código, desenvolvimento ágil e desenvolvimento paralelo. Como muitos dos módulos dependiam de outros, a solução foi estabelecer uma interface de comunicação com os pontos de contato das funções que seriam usadas para ligar os módulos. Logo, não houve a necessidade da implementação concreta de um módulo para que o outro funcionasse, bastava-se criar uma implementação provisória que seria substituída pelo módulo de fato assim que o seu desenvolvimento fosse concluído.

1. **Problemas de refatoração de código**

Durante o desenvolvimento do programa, muitas vezes percebia-se que o código poderia ser feito de uma forma mais genérica para melhorar a reutilização de código. Porém, realizar mudanças em um código já funcional gerava incertezas, uma vez que seria necessário revalidar toda a implementação já feita para conferir que o código refatorado atende a todos os requisitos que o código antigo atendia. Para facilitar esse processo, foram desenvolvidos testes de unidade automatizados, que validam automaticamente os requisitos estabelecidos.

1. **Testes de unidades**

Testes de unidade tem como característica principal a checagem de requisitos estabelecidos em um módulo. No entanto, como os módulos do sistema desenvolvidos estão interligados, foi necessário o desenvolvimento de implementações de teste que realizassem a comunicação com os módulos testados por meio da interface estabelecida. Desta forma, o desenvolvimento de um módulo X não impactava nos resultados do teste de um módulo Y.

1. **Erros de validações de arquivo**

Foi necessário efetuar uma validação do input do usuário, antes das devidas conversões serem realizadas. Assim, o programa era interrompido e exibia detalhadamente qual foi o erro encontrado, para que o usuário realizasse uma nova validação do arquivo antes que o enviasse novamente para o sistema.

1. **Função dos Integrantes**

**[Decisão em grupo]**

1. **Bibliografia**

**[Pietro]**