1. Termo de compromisso

Os membros do grupo afirmam que todo o código desenvolvido para este trabalho é de autoria própria. Exceto pelo material listado no item 3 deste relatório, os membros do grupo afirmam não ter copiado material da Internet nem obtiveram código de terceiros.

2. Membros do grupo e alocação de esforço

Gabriel Lemos <gabriellemos1901@gmail.com>

Emanuelly Carvalho <emanuellyvcarv@gmail.com> - Implementação e testes de `pager.c`

Rafael Castro <rafael.castro.ab@gmail.com>

3. Referências bibliográficas

Não foram utilizadas referências bibliográfiacas outras que as aulas online gravadas do professor, disponibilizadas no Moodle.

4. Estruturas de dados

1. Estruturas de Dados Utilizadas

1.1. `ProcessInfo`

- **Descrição:**
  - Estrutura que armazena informações sobre um processo.
- **Campos:**
  - `pid`: Identificador do processo.
  - `pages`: Ponteiro para a lista encadeada de páginas associadas a este processo.
  - `num_pages`: Número de páginas alocadas para este processo.
  - `next`: Ponteiro para o próximo `ProcessInfo` na lista de processos.
- **Justificativa:**
  - Esta estrutura é usada para manter o controle de todos os processos que estão sendo gerenciados pelo paginador. Ela permite a rápida localização das páginas associadas a um processo e o número total de páginas.

1.2. `PageInfo`

- **Descrição:**
  - Estrutura que armazena informações sobre uma página específica.
- **Campos:**
  - `vaddr`: Endereço virtual da página.
  - `frame`: Índice do quadro de memória onde a página está atualmente carregada. Se o valor for `-1`, significa que a página não está na memória.
  - `disk_block`: Índice do bloco de disco associado à página. Se o valor for `-1`, a página não tem um bloco de disco associado.
  - `next`: Ponteiro para a próxima `PageInfo` na lista de páginas do processo.
- **Justificativa:**
  - Esta estrutura é usada para gerenciar informações sobre cada página. O campo `frame` permite verificar se a página está na memória e o campo `disk_block` permite gerenciar o armazenamento e recuperação de páginas do disco. A lista encadeada permite que o sistema adicione ou remova páginas de maneira dinâmica e eficiente.

1.3. `frame_table` e `block_table`

- **Descrição:**
  - `frame_table`: Array que mantém o controle do uso dos quadros de memória. Cada posição indica se um quadro específico está ocupado (`1`) ou livre (`0`).
  - `block_table`: Array que mantém o controle do uso dos blocos de disco. Cada posição indica se um bloco específico está ocupado (`1`) ou livre (`0`).
- **Justificativa:**
  - Essas tabelas permitem ao paginador gerenciar eficientemente a alocação e liberação de quadros de memória e blocos de disco. Usar arrays para esse propósito é eficiente em termos de tempo, pois permite acesso direto e verificações rápidas de disponibilidade.

2. Mecanismo de Controle de Acesso e Modificação às Páginas

2.1. Controle de Acesso

- **Uso de Mutex:**
  - O `pthread_mutex_t pager_mutex` é utilizado para garantir que o acesso às estruturas de dados compartilhadas (como `process_list`, `frame_table` e `block_table`) seja feito de maneira segura em um ambiente de execução concorrente. O mutex evita condições de corrida, garantindo que apenas um thread possa modificar as estruturas ao mesmo tempo.

- **Verificação de Página e Processos:**
    - A função `find_process(pid)` é usada para localizar um processo com um identificador específico. Se o processo não for encontrado, a função `pager_fault` retorna imediatamente.
    - Dentro de `pager_fault`, a busca pela página correspondente ao endereço fornecido é realizada. Se a página não for encontrada, a função encerra com um erro crítico.

2.2. Modificação de Páginas

- **Substituição de Páginas:**
    - Quando uma página precisa ser carregada na memória e não há quadros livres, o algoritmo de substituição de páginas do tipo 'clock' é utilizado. Este algoritmo percorre a tabela de quadros de memória, selecionando um quadro que pode ser substituído com base em uma política de 'segunda chance'. A página que é removida é escrita no disco se necessário, e o quadro é liberado para a nova página.
- **Leitura e Escrita de Disco:**
    - Se uma página que está sendo carregada já possui um bloco de disco associado, o conteúdo é lido do disco e carregado no quadro de memória. Se a página foi previamente substituída, o bloco de disco associado é liberado para uso por outras páginas.
- **Gerenciamento de Permissões:**
    - As permissões de acesso às páginas são gerenciadas usando as funções `mmu_chprot` e `mmu_resident`. Essas funções ajustam as permissões de leitura e escrita conforme necessário. Por exemplo, ao carregar uma página, ela é inicialmente marcada como residente com permissões de leitura. Se a página já estiver na memória, as permissões são ajustadas para leitura e escrita conforme necessário.