Gerência de Memória

Daniel Guimarães – 1910462

Estrutura do projeto

O projeto consiste em um simulador de memória com paginação, com implementação de três algoritmos de *swapping* das páginas.

As estruturas de dados implementadas:

Página: Uma estrutura contendo todas as informações de uma página.

```
/* definicao de uma pagina para uma tabela */
typedef struct pagina {
    pagid_t meuid;          // o id da propria pagina
    tempo_t ultimaref;      // tempo da ultima referencia
    bool presente;          // se esta na memoria
    flagpag_t flagpag;      // FLAG_NULL, FLAG_REF OU FLAG_MOD
    quantidade_t quantidadeUsadas;      // quantidade de vez que foi executada
} Pagina;
```

Memória: Uma estrutura que simula a memória. Possui informações de quantas páginas tem, qual a quantidade máxima, e possui uma *lista encadeada* das páginas (somente alocada e usada nos algoritmos *NRU* e *LFU*.

```
/* definicao da memoria fisica do computador. Explicado mais abaixo */
typedef struct memoria {
    long paginasMax;
    long paginasAtual;
    Lista* paginas;
} Memoria;
```

Para o algoritmo *FIFO*, como o nome sugere, foi implementado uma estrutura de *fila*, que pode ser manipulada por meio de funções como *filaPush*, *filaPop*, *filaCria*.

```
/* definicao de fila, para o algoritmo FIFO*/
typedef struct fila {
    Pagina **lista;
    long ini;
    long n;
    long max;
} Fila;
```

Para os algoritmos *NRU* e *LFU*, foi implementada uma *lista encadeada*, que foi alocada na estrutura da memória. Novamente, para manipular foram implementadas funções como *listaCria, listaPush, listaPop*.

```c
/* definicao de elemento e lista, para uma lista encadeada */
typedef struct elemento {
    struct elemento *prev;         // elemento anterior na lista encadeada
    struct elemento *next;         // elemento sucessor na lista encadeada
    pagid_t x;                      // conteudo
} Elemento;

typedef struct lista {
    Elemento *inicio;
    int quantidade;
} Lista;
```

Por último, foi implementado uma estrutura contendo os resultados, para facilitar o entendimento.

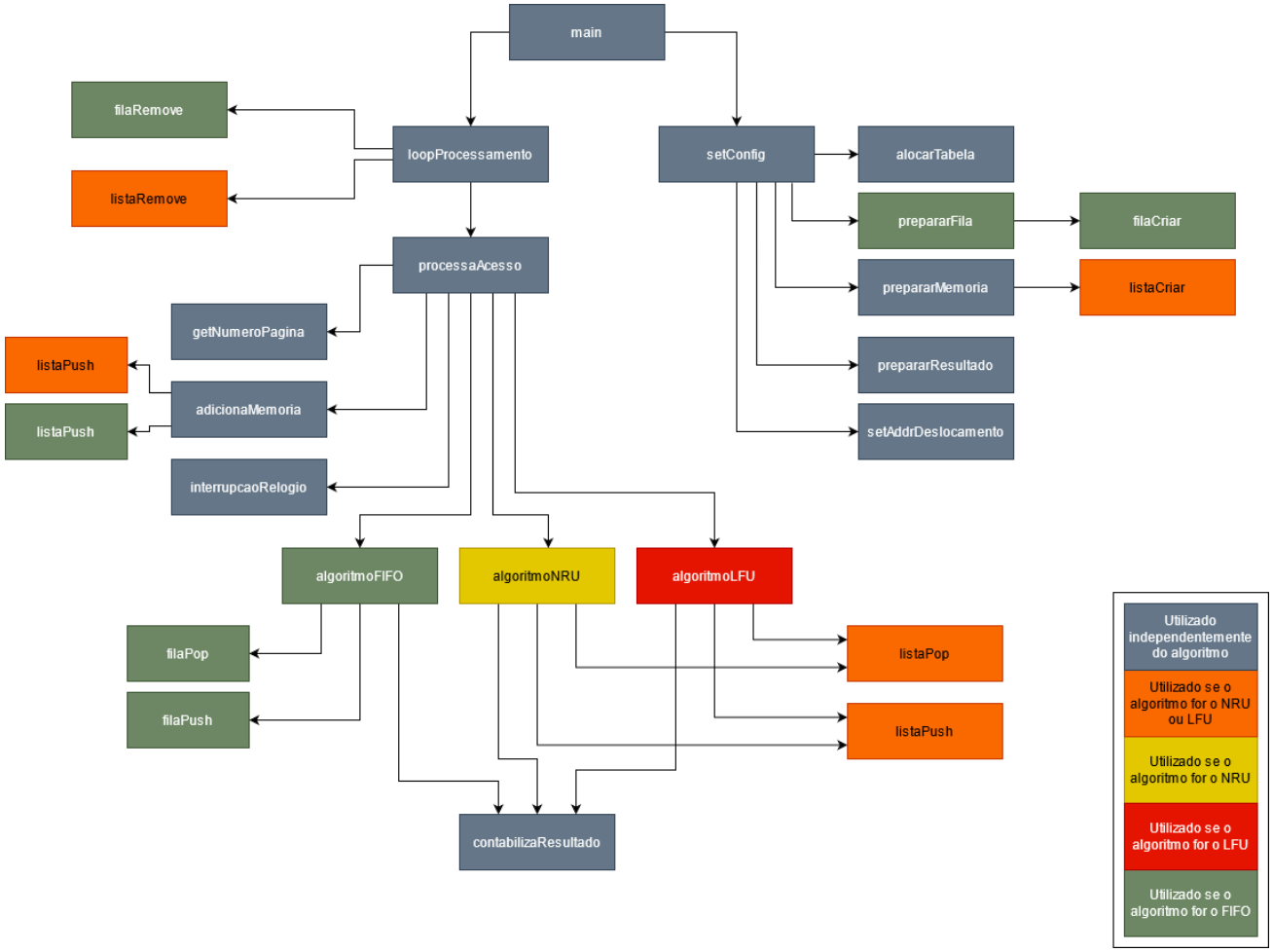
```c
/* definicao do resultado contendo os dados do simulador */
typedef struct resultado {
    long pageFaults;
    long pageWrites;
} Resultado;
```

Para poder manipular cada algoritmo e cada estrutura de dados, foram utilizadas quase trinta funções, cinco estruturas de dados, e algumas variáveis globais.

O código também possui vários *defines* para melhor organização do programa.

Na página a seguir contém um grafo para melhor entender a estrutura do simulador. Cada quadrado representa uma função, e a sua cor diz se ela é usada ou não, de acordo com o algoritmo. O código possui várias funções, e possui em torno de 600 linhas, pois foi criado em vista de facilitar o entendimento e evitar um código complicado. Cada função possui uma *docstring* explicando seu funcionamento.

O código completo foi entregue em conjunto com esse relatório.



A seguir está um exemplo da execução do simulador:

```
                                        >gcc -o simulador main.c

                                        >simulador LFU compressor.log 16 2
Configurando o simulador
Arquivo de entrada: compressor.log
Tamanho da memoria: 2 MB
Tamanho das paginas: 16 KB
Algoritmo de substituicao: LFU
Executando o simulador...
Finalizado!
Time: 1000000
Numero de page-faults: 741136
Numero de page-writes: 82483
```

Resultados

A seguir estão os resultados de simulações para cada *log* e algoritmo, porém sem mudar o tamanho da memória.

Página de 16KB Memória de 4MB	**FIFO** **2ª chance**			**NRU** *Not Recently Used*			**LFU** *Less Frequently Used*		
	fault	*write*	*perc*	*fault*	*write*	*perc*	*fault*	*write*	*perc*
compilador.log	998768	39996	4.00%	709639	23402	3.29%	709675	23402	3.29%
matriz.log	961536	40698	4.23%	677574	39669	5.85%	658397	38855	5.90%
compressor.log	0	0	0%	0	0	0%	0	0	0%
simulador.log	998485	92028	9.21%	891289	135311	15.1%	709675	23402	3.29%

page-writes e page-faults utilizando 4MB de memória e 16KB por página

Algumas observações para o resultado acima:

- Apesar de serem algoritmos diferentes, o percentual de *page-writes* para *para-faults* não foi muito diferente para cada arquivo, exceto o *simulador.log*.
- O *FIFO* teve a maior quantidade de *page-faults* em todos os arquivos dentre os algoritmos.
- Para todos os algoritmos, o *compressor.log* não houve *page-faults*. Isso pode ser devido ao tamanho da memória ou tamanho da página.

Refazendo os mesmo testes para diferentes tamanhos de memória ou paginas.

Página de 32KB Memória de 1MB	**FIFO** **2ª chance**			**NRU** *Not Recently Used*			**LFU** *Less Frequently Used*		
	fault	*write*	*perc*	*fault*	*write*	*perc*	*fault*	*write*	*perc*
compilador.log	999929	62047	6.20%	988282	160192	16.2%	929687	23402	2.51%
matriz.log	999622	58736	5.81%	834651	57005	6.82%	829266	57047	6.87%
compressor.log	999631	55876	5.58%	913147	108790	11.9%	913155	108799	11.91%
simulador.log	999923	114707	11.47%	988282	160192	16.2%	982478	160182	16.3%

Página de 8KB Memória de 16MB	**FIFO** **2ª chance**			**NRU** *Not Recently Used*			**LFU** *Less Frequently Used*		
	fault	*write*	*perc*	*fault*	*write*	*perc*	*fault*	*write*	*perc*
compilador.log	205022	6094	2.97%	36277	8065	22.2%	8223	3075	37.3%
matriz.log	176313	6298	3.57%	111278	7373	6.62%	108572	6367	5.86%
compressor.log	0	0	0%	0	0	0%	0	0	0%
simulador.log	516528	47469	9.19%	476844	76618	16.0%	131071	21247	16.2%

Observações gerais das simulações:

- O *compressor* precisa de pouca memória, pois com um grande número de páginas na memória física, não houve nenhum *page-fault*. Isso significa que o programa é pequeno e consulta pouca memória, ou que foi bem comprimido para consultar sempre as mesmas páginas.

- No caso da terceira tabela, em que há a maior memória e menor página possível, o *FIFO* foi o pior algoritmo, em quantidade de *page-faults* e *page-writes*.

- Apesar de que na ultima tabela, para os algoritmos de *NRU* e *LFU*, a porcentagem de *page-write* sobre *page-fault*, isso é devido a quantidade de *page-faults* serem bem menores, por isso chegou próximo ao número de *page-writes*.