Universidade Federal do Rio de Janeiro
DCC - IM

T2 - Sistemas Operacionais

Gerenciamento de Memória

Leonardo Neves da Silva
Luan Cerqueira Martins

Rio de Janeiro
2015.1

SUMÁRIO

1. **Grupo ……... página 4**
2. **Definição do trabalho …….. página 4**
3. **Códigos ……... página 5**
    a. **makefile ……...…….. página 5**
    b. **memoria.h ……..…... página 6**
    c. **memoria.c ….……..…… página 10**
4. **Resultados …….. página 24**
5. **Considerações finais …… página 30**

Grupo

- **Leonardo Neves da Silva**
- **Luan Cerqueira Martins**

https://github.com/LawnOni/gerencia-de-memoria-so-ufrj

Definição do trabalho

https://github.com/LawnOni/gerencia-de-memoria-so-ufrj/blob/master/SO-Trabalho2.pdf

Código - makefile

```
all:
	@echo
_____________________________________________________________________Cria
	clear all
	@echo
_____________________________________________________________________compila
	make gcc
	ls
	@echo
_____________________________________________________________________Executa
	./memoria.out




clean:
	clear all
	rm a.out memoria memoria.out


gcc:
	@gcc memoria.c -o memoria.out -lpthread -w


guake_sublime:
	@gcc memoria.c -o memoria.out -lpthread
	guake
	guake -e ./memoria.out
```

Codigo - memoria.h

```c
#include      <stdio.h>
#include      <stdlib.h>
#include      <pthread.h>
#include      <sched.h>
#include      <stdbool.h>
#include      <time.h>
//print colorido -exemplo
//printf(ANSI_COLOR_RED "This text is RED!"
ANSI_COLOR_RESET "\n");
#define ANSI_COLOR_RED     "\x1b[31m"
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define ANSI_COLOR_YELLOW  "\x1b[33m"
#define ANSI_COLOR_BLUE    "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN    "\x1b[36m"
#define ANSI_COLOR_RESET   "\x1b[0m"
#define ANSI_BOLD_ON    "\x1b[1m"
#define ANSI_INVERSE_ON "\x1b[7m"
#define ANSI_BOLD_OFF   "\x1b[22m"
#define ANSI_FG_BLACK   "\x1b[30m"
#define ANSI_FG_RED     "\x1b[31m"
#define ANSI_FG_GREEN   "\x1b[32m"
#define ANSI_FG_YELLOW  "\x1b[33m"
#define ANSI_FG_BLUE    "\x1b[34m"
#define ANSI_FG_MAGENTA "\x1b[35m"
#define ANSI_FG_CYAN    "\x1b[36m"
#define ANSI_FG_WHITE   "\x1b[37m"
#define ANSI_BG_RED     "\x1b[41m"
#define ANSI_BG_GREEN   "\x1b[42m"
#define ANSI_BG_YELLOW  "\x1b[43m"
#define ANSI_BG_BLUE    "\x1b[44m"
#define ANSI_BG_MAGENTA "\x1b[45m"
#define ANSI_BG_CYAN    "\x1b[46m"
#define ANSI_BG_WHITE   "\x1b[47m"



#define FRAME_LIMIT 24 //64*MULTIPLUS //main_memory_size
//64
#define MAIN_MEMORY_SIZE FRAME_LIMIT //64
#define VIRTUAL_MEMORY_SIZE 2*FRAME_LIMIT //NAOSEIQTO
```

```
#define THREAD_LIMIT 15//20*MULTIPLUS //20
#define PAGE_LIMIT 10//50*MULTIPLUS //50
#define WORKSET_LIMIT 5//PAGE_LIMIT //PARA TESTES //4


// #define FRAME_LIMIT 64
// #define MAIN_MEMORY_SIZE FRAME_LIMIT
// #define VIRTUAL_MEMORY_SIZE FRAME_LIMIT
// #define THREAD_LIMIT 120
// #define PAGE_LIMIT 50
// #define WORKSET_LIMIT 5




struct Page{
        int process_id;
   int number;
   int value;
};


typedef union{
        int ids[PAGE_LIMIT];
        int frames[PAGE_LIMIT];
} WorkingSet;


struct Process{
   int id;
   struct Page page_list[PAGE_LIMIT];
   WorkingSet works;
};


int number_of_process = 0;
struct Process process_list[THREAD_LIMIT];


int running_process[THREAD_LIMIT] = { [0 ... THREAD_LIMIT-1
] = -1 };
int running_process_index =0;
int stopped_process[THREAD_LIMIT] = { [0 ... THREAD_LIMIT-1
] = -1 };
int stopped_process_index=0;
```

```
//LRU
int recent_frame[FRAME_LIMIT] = { [0 ... FRAME_LIMIT-1 ] =
-1 };
int number_of_free_frames = FRAME_LIMIT;
int number_of_non_free_frames = 0;


struct Page main_memory[FRAME_LIMIT];
struct Page virtual_memory[VIRTUAL_MEMORY_SIZE];


pthread_t thread[THREAD_LIMIT];
pthread_mutex_t memory_lock;
pthread_mutex_t process_list_lock;


int page_queue[FRAME_LIMIT];


// Gerenciador de memória
void print_memories();
void reset_main_memory();
void reset_virtual_memory();
int free_frames();
void memory_overflow();


//process functions
void request_page(int process_id, int page_number);
int create_process();
void* execute_process(int id);
void initialize_page_list_of_process(int size, int
process_id);
void running_processes();
void stop_process(int process_id);
void print_workingset(int process_id);
bool using_all_working_set(int process_id);
int insert_pag_empty_frames(int process_id, int
page_number);
int insert_pag_full_memory(int process_id, int
page_number);
int insert_pag_full_workingset(int process_id, int
page_number);
```

```
bool workingset_is_full(int process_id);


//Queue functions
void add_page_to_queue(int newPage);
void refresh_queue(int page);
void shift_queue(int offSet);
void print_queue();
void print_queue_details();
int get_queue_offset(int page);
void print_LRUF();
int refresh_LRUF(int old_frame_in_memory);
```

Codigo - memoria.c

```
#define MULTIPLUS 1 //multiplicador para alterar
facilmente/proporcionalmente o tamanho das threads e memorias
#define SLEEP_TIME 500000//500000/2 //3000000


#include "memoria.h"
//ao terminar lembrar de colocar argumentos: como os tamanhos e o sleep
time
int main( int argc, char *argv[ ] ){
        reset_main_memory();
        reset_virtual_memory();


        //Inicializa mutex
        if (pthread_mutex_init(&memory_lock, NULL) != 0) {
        printf("\n mutex init failed\n");
        return 1;
    }
    if (pthread_mutex_init(&process_list_lock, NULL) != 0) {
        printf("\n mutex init failed\n");
        return 1;
    }


    //inicia threads
        int i;
        for(i = 0; i < THREAD_LIMIT; i++){
                pthread_create(&thread[i], NULL, execute_process,
create_process());
                usleep(SLEEP_TIME);
        }


        for(i = 0; i < THREAD_LIMIT; i++){
                pthread_join(thread[i], NULL);
        }


        pthread_mutex_destroy(&memory_lock);
```

```
        //printa situação final
        system("clear");
        print_memories();




        return 0;
}


void* execute_process(int id){
        int i;
        for(i = 0; i < PAGE_LIMIT; i++){
                pthread_mutex_lock(&memory_lock);
                usleep(SLEEP_TIME);
                system("clear");
                printf("--->Entrando com PID: %d e Pagina: %d\n", id, i);
                request_page(id, i);
                print_memories();
                pthread_mutex_unlock(&memory_lock);


                usleep(0); //Troca de contexto
        }


        pthread_mutex_lock(&memory_lock);
        system("clear");
        printf(ANSI_COLOR_RED"\n\t---->Parando processo com PID:
%d\n"ANSI_COLOR_RESET, id);
        stop_process(id);
        print_memories();
        pthread_mutex_unlock(&memory_lock);
}


int create_process(){
        int i;
        struct Process _process;
        pthread_mutex_lock(&process_list_lock);
        _process.id = number_of_process;
        process_list[_process.id] = _process;
        number_of_process++;
        running_process[running_process_index] = _process.id;
```

```
        running_process_index++;
        for(i=0;i<PAGE_LIMIT;i++)
process_list[_process.id].works.frames[i] = -1;
        pthread_mutex_unlock(&process_list_lock);
        initialize_page_list_of_process(PAGE_LIMIT, number_of_process -
1);


        return _process.id;
}


void print_memories(){
        running_processes();


        int i;
        number_of_non_free_frames = 0;
        number_of_free_frames = 0;


        printf("\tMEMORIA PRINCIPAL\t\t\t\t\t\tMEMORIA SECUNDARIA\n");



printf("_____________________________________\t\t_____________________
_____________________________________________________\n");


        for (i = 0; i < FRAME_LIMIT; i++){
                //main 0-15
                if(main_memory[i].process_id > -1){
                        printf("Frame: %2d -> Processo: %2d -> Page:
%2d.\t\t", i, main_memory[i].process_id, main_memory[i].number);
                        number_of_non_free_frames++;
                }
                else{
                        printf("Frame: %2d Vazio\t\t\t\t\t", i);
                        number_of_free_frames++;
                }

                //0-15
                if(virtual_memory[i].process_id > -1)  printf("Frame: %2d
-> Processo: %2d -> Page: %2d.\t\t", i, virtual_memory[i].process_id,
```

```
virtual_memory[i].number);
                else  printf("Frame: %2d Vazio\t\t\t\t\t", i);


                //16-31
                if (FRAME_LIMIT+i>=VIRTUAL_MEMORY_SIZE)
printf("--\t\t\t\t\t\t");
                else if(virtual_memory[FRAME_LIMIT+i].process_id > -1)
printf("Frame: %2d -> Processo: %2d -> Page: %2d.\t\t", FRAME_LIMIT+i,
virtual_memory[FRAME_LIMIT+i].process_id,
virtual_memory[FRAME_LIMIT+i].number);
                else printf("Frame: %2d Vazio\t\t\t\t\t", FRAME_LIMIT+i);
                printf("\n");
        }


        if ( (number_of_non_free_frames + number_of_free_frames) !=
FRAME_LIMIT) { printf("Erro em  qtdade frames"); exit(0);}
        print_LRUF();


        //// **verificar
        //print_queue_details();




        printf("_____Numero da
Pagina_____");
        for (i = 0; i < PAGE_LIMIT; i++)     printf("_%2i",i);
        printf("_\n");
        for(i=0;i<THREAD_LIMIT;i++) print_workingset(i);
}


void initialize_page_list_of_process(int size, int process_id){
        int i;
        for(i = 0; i < size; i++)
        {
                process_list[process_id].page_list[i].process_id =
process_id;
                process_list[process_id].page_list[i].number = i;
        }
}
```

```
void stop_process(int process_id){
	int i;
	stopped_process[stopped_process_index] = process_id;
	stopped_process_index++;
	for (i=0; i<THREAD_LIMIT;i++) if (running_process[i] ==
process_id) running_process[i] = -1;


	for (i = 0; i < FRAME_LIMIT; i++){
		if(main_memory[i].process_id == process_id){
			main_memory[i].process_id = -1;
		}
	}


	for (i = 0; i < PAGE_LIMIT; i++)
process_list[process_id].works.frames[i]=-1;
}


void reset_main_memory(){
	int i;
	for (i = 0; i < FRAME_LIMIT; i++){
		main_memory[i].process_id = -1;
	}
}


void reset_virtual_memory(){
	int i;
	for (i = 0; i < VIRTUAL_MEMORY_SIZE; i++){
		virtual_memory[i].process_id = -1;
	}
}


void memory_overflow(){


	printf("*MEMORIA ESTOURADA - DEVEMOS TRATAR?\n"); //IDEIA, PARAR E
GRAVA-LA TODA EM UM ARQUIVO APOS ISSO RESETA-LA
}
```

```
void add_page_to_queue(int newPage){
        shift_queue(0);
        page_queue[FRAME_LIMIT - 1] = newPage;
}


void refresh_queue(int page){
        int offSet = get_queue_offset(page);
        shift_queue(offSet);
        page_queue[FRAME_LIMIT - 1] = page;
}


void shift_queue(int offSet){
        int i;
        for (i = offSet; i < FRAME_LIMIT - 1; i++) page_queue[i] =
page_queue[i+1];
}


int get_queue_offset(int page){
        int i;
        for (i = 0; i < FRAME_LIMIT; i++)
        {
            if(page_queue[i] == page)
                return i;
        }
}


void running_processes(){
        int i;
        printf(ANSI_COLOR_YELLOW "Estamos executando os processos:
\n"ANSI_COLOR_RESET);
        for (i=0; i<THREAD_LIMIT;i++)
                if (running_process[i] > -1)
                        printf(ANSI_COLOR_YELLOW "%i
"ANSI_COLOR_RESET,running_process[i]);
        printf("\n");
        printf(ANSI_COLOR_GREEN "Os seguintes processos terminaram: \n"
ANSI_COLOR_RESET);
        for (i=0; i<THREAD_LIMIT;i++)
                if (stopped_process[i] >-1)
                        printf(ANSI_COLOR_GREEN "%i "
ANSI_COLOR_RESET,stopped_process[i]);
```

```
        printf("\n");
}


void print_queue(){
        int i;
        printf(ANSI_COLOR_CYAN"\nFila:  {process,page}\n");
        for(i=0;i<FRAME_LIMIT;i++)   printf("[%d,%d] \v",
page_queue[i]/PAGE_LIMIT, page_queue[i]%PAGE_LIMIT);
        printf("\n"ANSI_COLOR_RESET);
}


void print_queue_details(){
        int i;


                for (i = 0; i < FRAME_LIMIT; i++){
                        if(i == 0)
                                printf("Fila:\tSai ----> %d: Processo: %d ->
Page: %d.\n", i, page_queue[i]/PAGE_LIMIT, page_queue[i]%PAGE_LIMIT);
                        else if(i == FRAME_LIMIT -1)
                                printf("    |\tEntra --> %d: Processo: %d ->
Page: %d.\n", i, page_queue[i]/PAGE_LIMIT, page_queue[i]%PAGE_LIMIT);
                        else
                                printf("    |\t--------> %d: Processo: %d ->
Page: %d.\n", i, page_queue[i]/PAGE_LIMIT, page_queue[i]%PAGE_LIMIT);
                }
}


void print_LRUF(){
        int i;
        printf(ANSI_BOLD_ON"LRUF - Last Recent Used Frames: [new ..
old]"ANSI_COLOR_RESET);
        printf("\nRecente ->" );
        printf("  "ANSI_INVERSE_ON "%i" ANSI_COLOR_RESET,
recent_frame[0]);
        for(i=1;i<FRAME_LIMIT;i++) printf("  %i", recent_frame[i]);
        printf(" -> Proximo a ser removido \n");
}


void print_workingset(int process_id){
```

```
        int i;
        printf("Paginas do processo %2i esta alocado nos seguintes frames:
",process_id);
        for (i = 0; i < PAGE_LIMIT; i++) if
(process_list[process_id].works.frames[i]==-1) printf("
%2i",process_list[process_id].works.frames[i]);
                                                                    else
printf(ANSI_INVERSE_ON"
%2i"ANSI_COLOR_RESET,process_list[process_id].works.frames[i]);
        printf("\n");
}


bool using_all_working_set(int process_id){
        int i;
        for (i = 0; i < PAGE_LIMIT; i++) if
(process_list[process_id].works.frames[i]==-1){
                return false;
        }
        return true;
}


void request_page(int process_id, int page_number){
        int frame=FRAME_LIMIT-1;//por padrao, em caso de erro, remover o
last frame da lista para a virtual
        int freeframes = free_frames();


        if( workingset_is_full(process_id) ){
                printf("... O working set do processo %i esta
cheio\n",process_id);
                frame=insert_pag_full_workingset(process_id, page_number);
        }
        //verifica se ha frames vazios
        else if ( freeframes> 0){
                printf("... Ainda existem frames vazios\n");
                frame=insert_pag_empty_frames(process_id, page_number);
        }


        else{
                printf("... A memoria esta cheia\n");
                frame=insert_pag_full_memory(process_id, page_number);
        }
```

```
	//adiciona o frame da nova pagina ao workingsetlimit
2_metodos
	//process_list[ main_memory[frame].process_id ].works.frames[
main_memory[frame].number ]=frame;
	process_list[process_id].works.frames[page_number]=frame;


	//insere na memoria principal
	main_memory[frame] =
process_list[process_id].page_list[page_number];
	add_page_to_queue(PAGE_LIMIT * process_id +
main_memory[frame].number);
}


int insert_pag_empty_frames(int process_id, int page_number){
	int i;
	int frame=FRAME_LIMIT-1;//por padrao, em caso de erro, remover o
last frame da lista para a virtual
	int randompage = rand()%FRAME_LIMIT; //sorteia uma pagina,
	srand(time(NULL));


	for (i = randompage; i < FRAME_LIMIT; i++){
		if(main_memory[i].process_id == -1){
			frame=refresh_LRUF(i); //atribui a variavel frame o
valor do frame vazio
			return frame;
		}
	}


	for (i = 0; i < randompage; i++){
		if(main_memory[i].process_id == -1){
			frame=refresh_LRUF(i); //atribui a variavel frame o
valor do frame vazio
			return frame;
		}
	}
}


int insert_pag_full_memory(int process_id, int page_number){
	int i;
```

```
	int frame=FRAME_LIMIT-1;//por padrao, em caso de erro, remover o
last frame da lista para a virtual
	int last = FRAME_LIMIT-1; //ultimo da fila, ira para o inicio da
fila, será primeiro


	//atualiza processos na virtual
	if (virtual_memory[0].process_id != -1) {
		if(virtual_memory[VIRTUAL_MEMORY_SIZE-1].process_id != -1)
memory_overflow();
		for (i = VIRTUAL_MEMORY_SIZE-1; i > 0; i--)
virtual_memory[i] = virtual_memory[i-1];

	}

	//movimenta o LRUF
	if(recent_frame[last] != -1) {
		frame=refresh_LRUF(recent_frame[last] );


		//COPIA PARA A MEMORIA VIRTUAL O FRAME Q SAIRA
		virtual_memory[0] = main_memory[ recent_frame[0] ];
	}


	//remove a pagina do workingset
	//Frame:		frame
	//Processo:	main_memory[frame].process_id
	//Page:		main_memory[frame].number;

process_list[main_memory[frame].process_id].works.frames[main_memory[fram
e].number]=-1;


	return frame;
}


int insert_pag_full_workingset(int process_id, int page_number){
	int i;
	int remover = FRAME_LIMIT-1;//POR PADRAO COLOCA NA ULTIMA POSICAO
	int recent = -1;
	int index = -1;
	int randompage = rand()%PAGE_LIMIT; //sorteia uma pagina,
	srand(time(NULL));
```

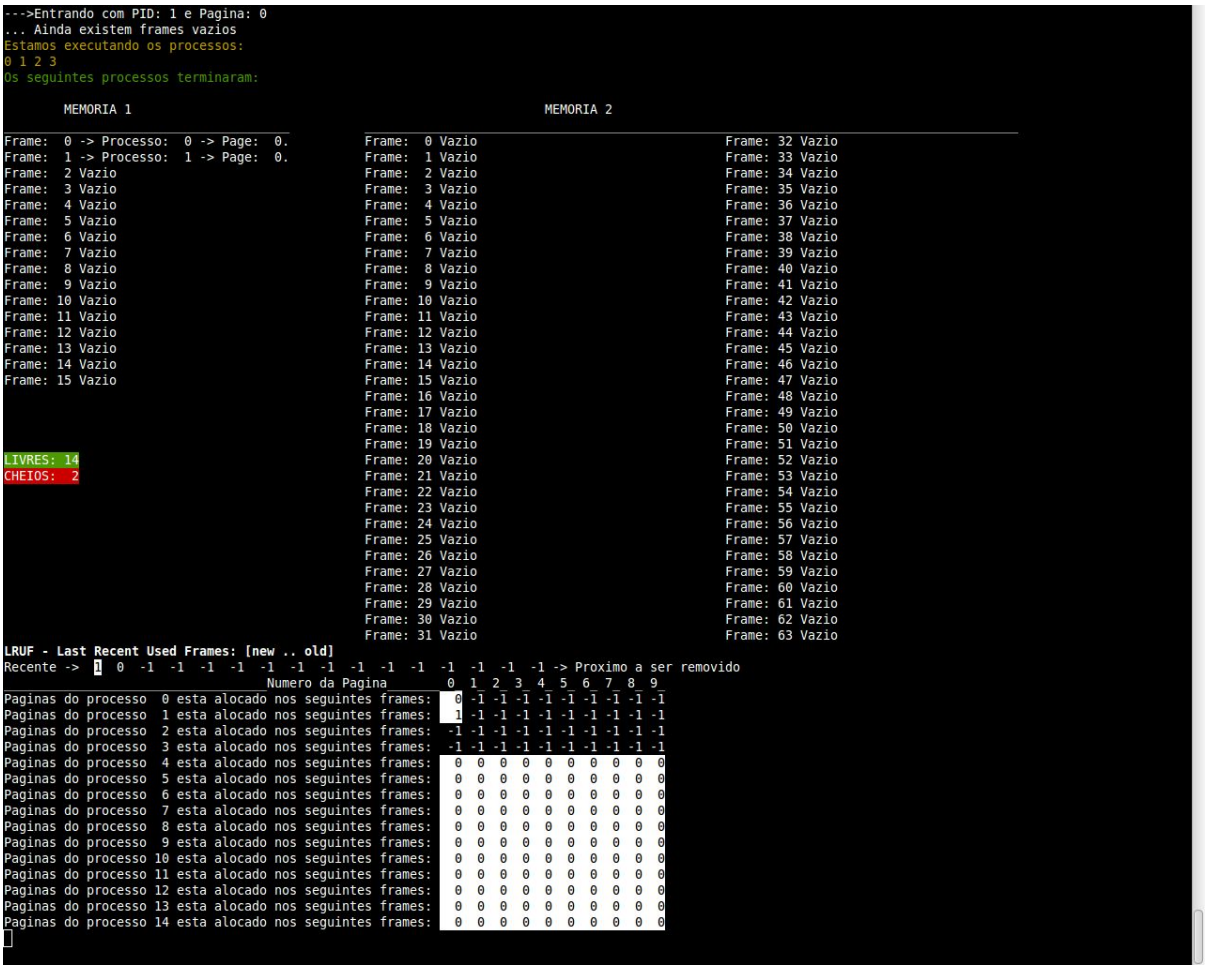
```
        //sorteia dentre o workingset qual pagina/frame saira da memoria
        for (i = 0; i < PAGE_LIMIT; i++) if
(process_list[process_id].works.frames[i] != -1) {
                remover = process_list[process_id].works.frames[i];
                //se encontrarmos uma pagina alocada antes da sorteada
continuamos procurando, e encontrarmos depois usamos aquele frame para
substituicao
                if (i > randompage) break;
        }
        printf("Removendo processo do Frame %i\n", remover);



        //atualiza processos na virtual
        if (virtual_memory[0].process_id != -1) {
                if(virtual_memory[VIRTUAL_MEMORY_SIZE-1].process_id != -1)
memory_overflow();
                for (i = VIRTUAL_MEMORY_SIZE-1; i > 0; i--)
virtual_memory[i] = virtual_memory[i-1];
        }


        for (i = 0; i<FRAME_LIMIT; i++) {
                if ( recent_frame[i] == remover ){
                        recent = recent_frame[i];
                        index=i;
                        break ;
                }
        }


        if (recent == -1 || index ==-1)
        {
                printf("Erro**\n%i %i", recent, index);
                exit(-1);
        }



        refresh_LRUF(remover);
```

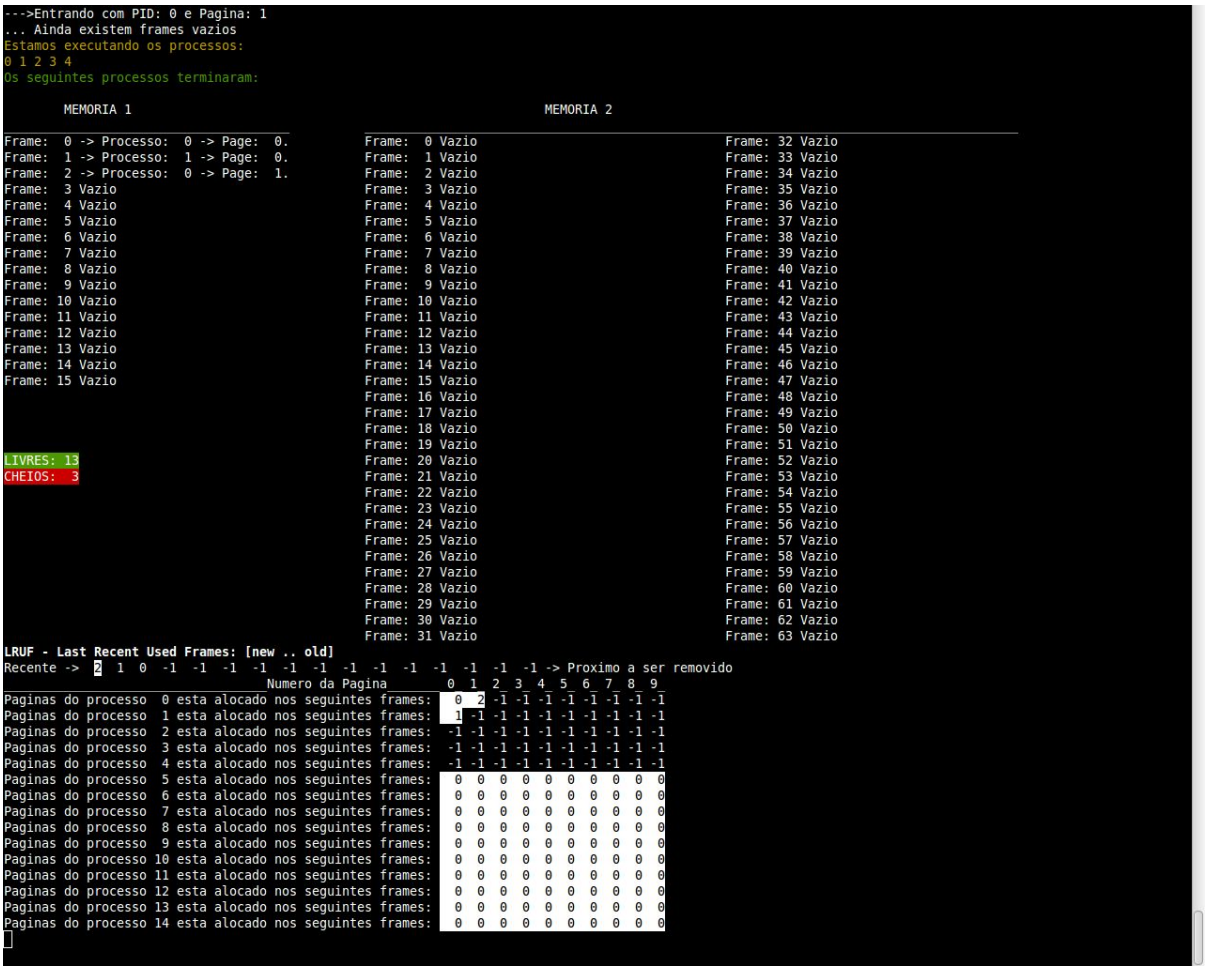
```
        //COPIA PARA A MEMORIA VIRTUAL O FRAME Q SAIRA
        virtual_memory[0] = main_memory[ remover ];



        //atualiza o valor do frame a ser retirado da memoria principal
        //frame=remover;


        //remove a pagina do workingset
        //Frame:              frame
        //Processo:    main_memory[frame].process_id
        //Page:               main_memory[frame].number;
        process_list[ main_memory[remover].process_id
].works.frames[main_memory[remover].number]=-1;


        return remover;
}


bool workingset_is_full(int process_id){
        int i,workingset =0;


        for (i = 0; i < PAGE_LIMIT; i++) if
(process_list[process_id].works.frames[i] != -1) workingset++;


        if (workingset == WORKSET_LIMIT) return true;
        else if (workingset < WORKSET_LIMIT) return false;
        else if (workingset > WORKSET_LIMIT) {
                printf(ANSI_BG_RED ":::: WORKSET_LIMIT Estourado pelo
processo %i ::::\n", process_id);

                // printf("Remover FRAME %i \n",
insert_pag_full_workingset(process_id, NULL));
                // print_memories();
                exit(-1);
        }
}
```

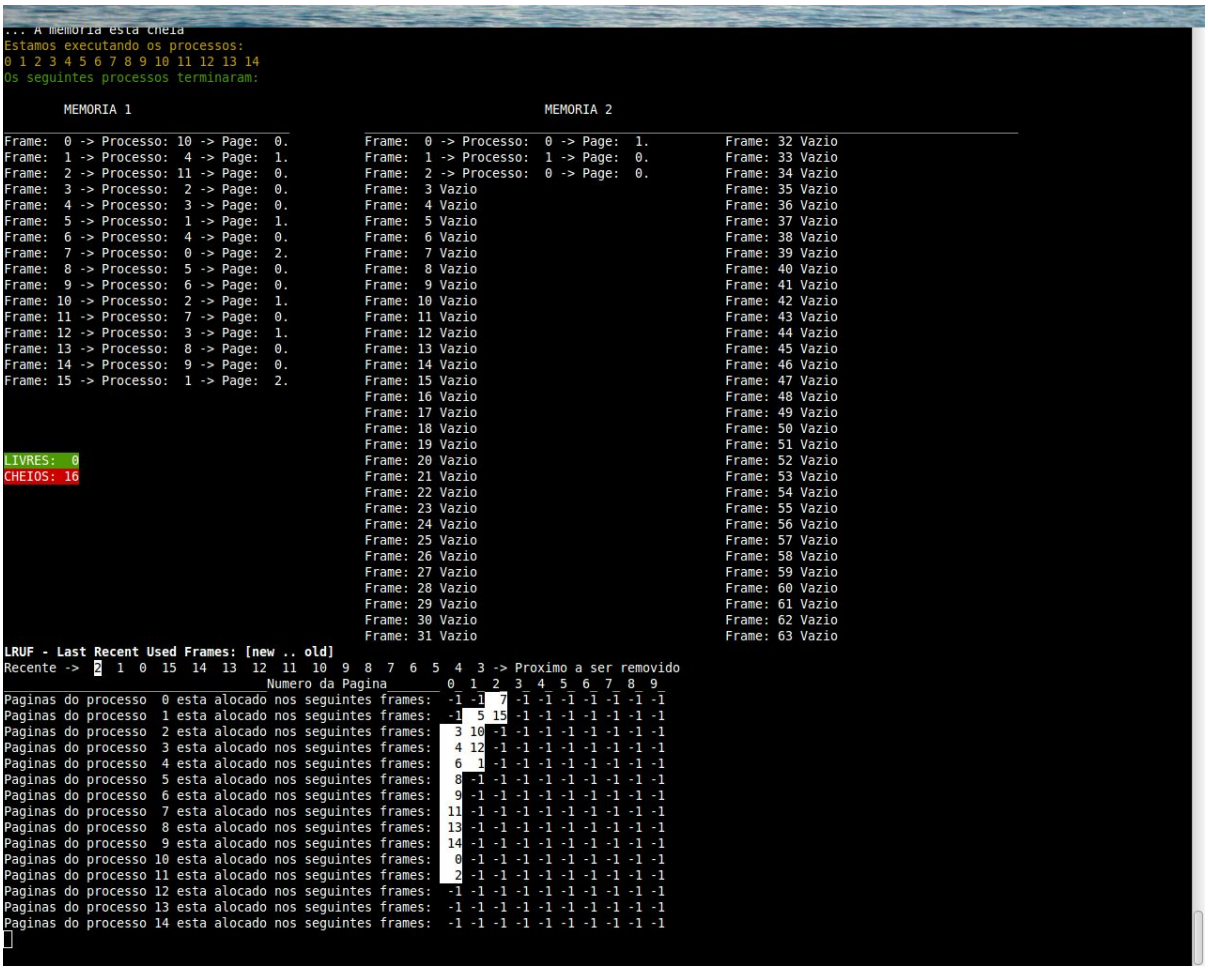
```
int free_frames(){
	int i;
	number_of_non_free_frames = 0;
	number_of_free_frames = 0;


	for (i = 0; i < FRAME_LIMIT; i++){
		if(main_memory[i].process_id > -1)
number_of_non_free_frames++;
		else number_of_free_frames++;
	}


	if ( (number_of_non_free_frames + number_of_free_frames) !=
FRAME_LIMIT) {
		printf("f=%i nf=%i\nErro2 em  qtdade
frames\n",number_of_non_free_frames , number_of_free_frames);
		exit(0);
	}
	return number_of_free_frames;
}


int refresh_LRUF(int old_frame_in_memory){
	int i,j;
	int ToDo=0;
	int recent=-1;


	for (i = 0; i < FRAME_LIMIT; i++){
		if (recent_frame[i] == old_frame_in_memory){
			//significa que o frame ja esta referenciado no LRUF
			ToDo++;
			break;
		}
	}


	//insere na LRU pela primeira vez
	if (ToDo == 0){
		for (i = FRAME_LIMIT; i > 0; i--) {
			recent_frame[i] = recent_frame[i-1];
		}
		recent_frame[0] = old_frame_in_memory;
	}
```

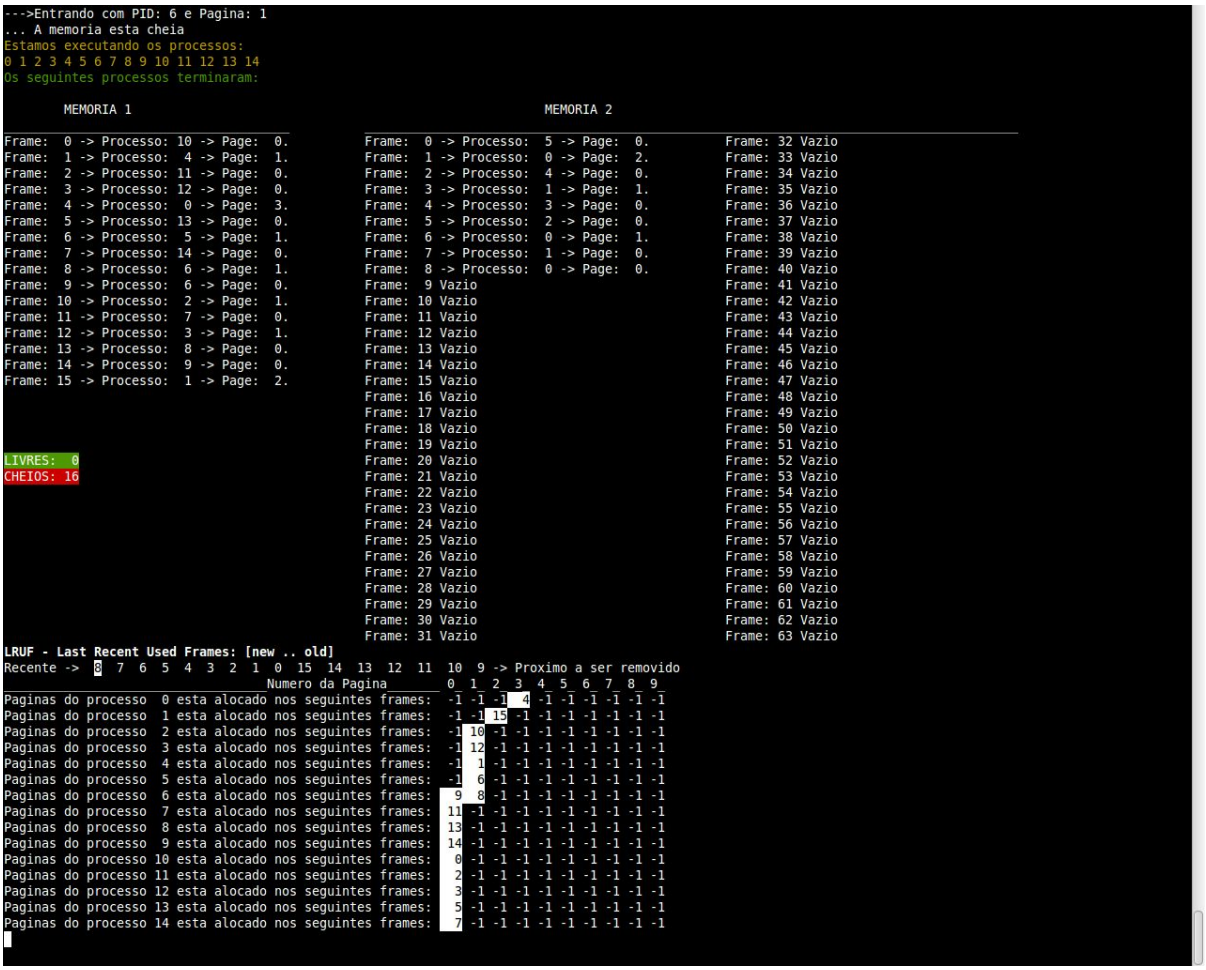
```
        //reinsere no LRU
        else if (ToDo == 1){
                for (i = 0; i<FRAME_LIMIT; i++) {
                        if(recent_frame[i]== old_frame_in_memory ){
//atualiza o LRU, e remove copias da fila antes de inserir
                                for (j = i; j > 0; j--) {
                                        recent_frame[j]=recent_frame[j-1];
                                }
                                recent_frame[0]=old_frame_in_memory; //o
ultimo vira o primeiro
                        }
                }
        }


        //erros
        else if ( ToDo >1){
                printf("Erro ***\n r %i t %i",recent,ToDo);
                exit(-1);
        }


        return recent_frame[0];
}
```

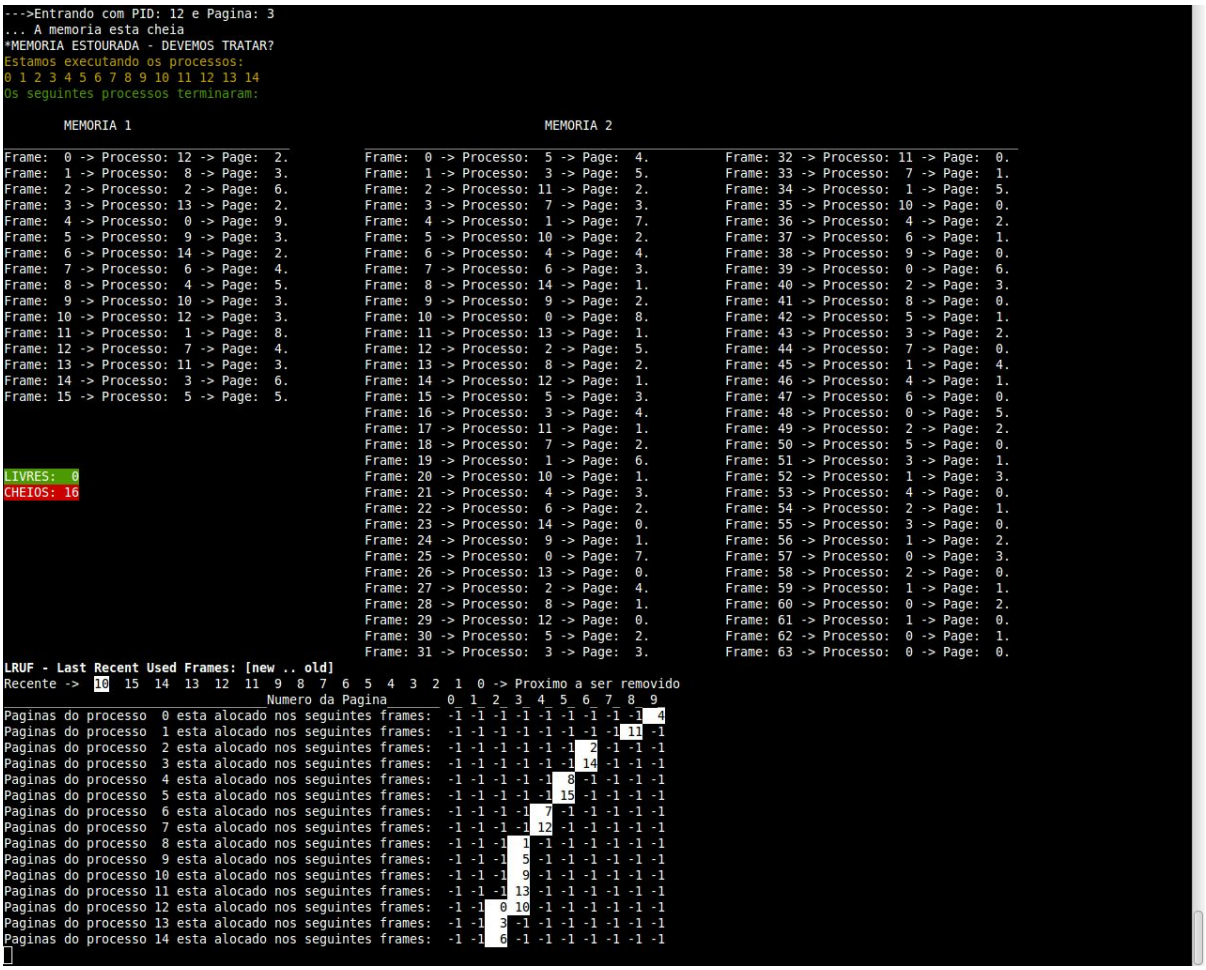
Resultados

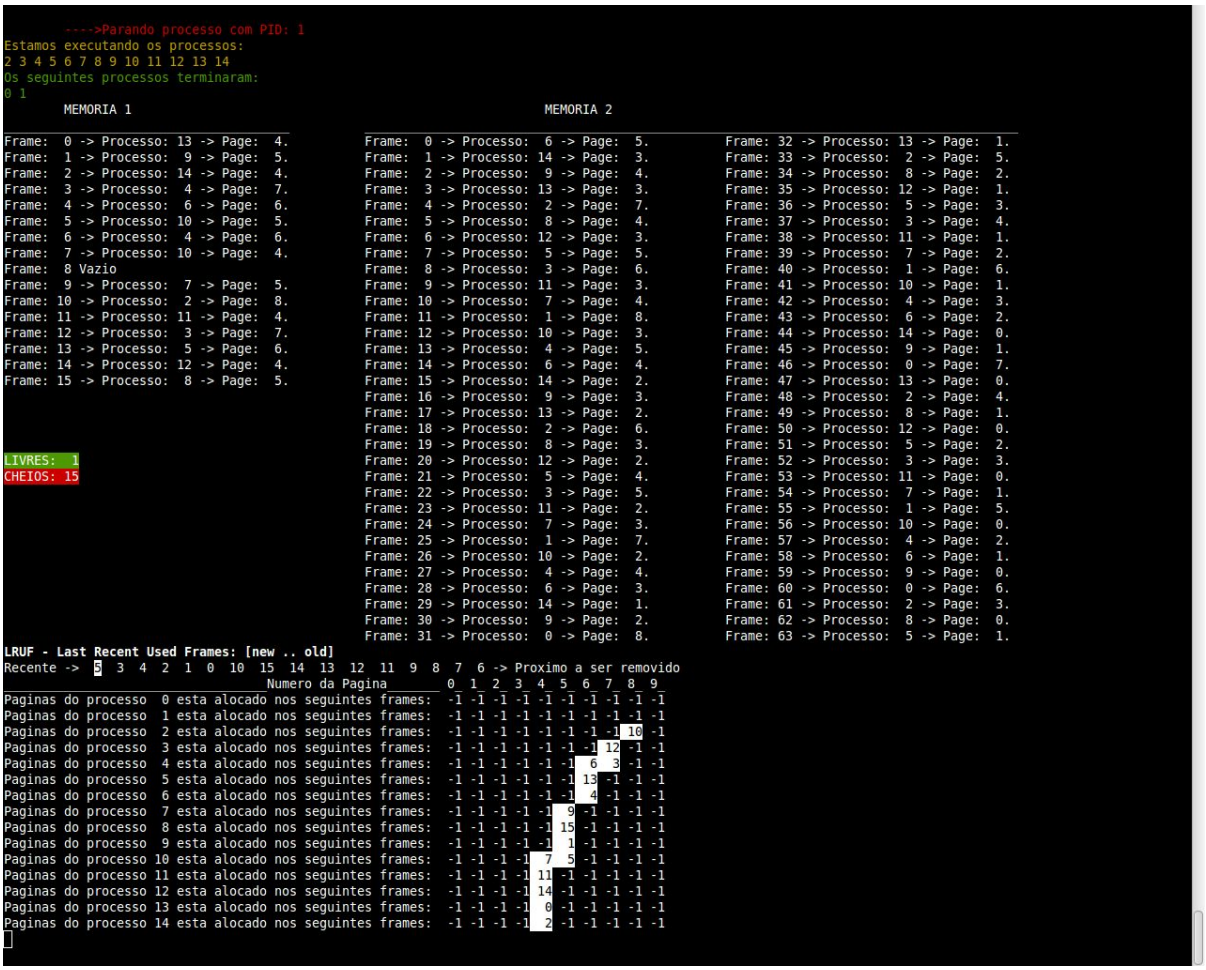

--->Entrando com PID: 1 e Pagina: 0
... Ainda existem frames vazios
Estamos executando os processos:
0 1 2 3
Os seguintes processos terminaram:
MEMORIA 1
MEMORIA 2
Frame: 0 -> Processo: 0 -> Page: 0.
Frame: 1 -> Processo: 1 -> Page: 0.
LIVRES: 14
CHEIOS: 2
LRUF - Last Recent Used Frames: [new .. old]
Recente -> 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -> Proximo a ser removido
Numero da Pagina 0 1 2 3 4 5 6 7 8 9
Paginas do processo 0 esta alocado nos seguintes frames: 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 1 esta alocado nos seguintes frames: 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 2 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 3 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 4 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 5 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 6 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 7 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 8 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 9 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 10 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 11 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 12 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 13 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 14 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0


--->Entrando com PID: 0 e Pagina: 1
... Ainda existem frames vazios
Estamos executando os processos:
0 1 2 3 4
Os seguintes processos terminaram:
MEMORIA 1
MEMORIA 2
Frame: 0 -> Processo: 0 -> Page: 0.
Frame: 1 -> Processo: 1 -> Page: 0.
Frame: 2 -> Processo: 0 -> Page: 1.
LIVRES: 13
CHEIOS: 3
LRUF - Last Recent Used Frames: [new .. old]
Recente -> 2 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -> Proximo a ser removido
Numero da Pagina
Paginas do processo 0 esta alocado nos seguintes frames: 0 2 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 1 esta alocado nos seguintes frames: 1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 2 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 3 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 4 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 5 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 6 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 7 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 8 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 9 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 10 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 11 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 12 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 13 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0
Paginas do processo 14 esta alocado nos seguintes frames: 0 0 0 0 0 0 0 0 0 0


... A memoria esta cheia
Estamos executando os processos:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Os seguintes processos terminaram:
MEMORIA 1
Frame: 0 -> Processo: 10 -> Page: 0.
Frame: 1 -> Processo: 4 -> Page: 1.
Frame: 2 -> Processo: 11 -> Page: 0.
Frame: 3 -> Processo: 2 -> Page: 0.
Frame: 4 -> Processo: 3 -> Page: 0.
Frame: 5 -> Processo: 1 -> Page: 1.
Frame: 6 -> Processo: 4 -> Page: 0.
Frame: 7 -> Processo: 0 -> Page: 2.
Frame: 8 -> Processo: 5 -> Page: 0.
Frame: 9 -> Processo: 6 -> Page: 0.
Frame: 10 -> Processo: 2 -> Page: 1.
Frame: 11 -> Processo: 7 -> Page: 0.
Frame: 12 -> Processo: 3 -> Page: 1.
Frame: 13 -> Processo: 8 -> Page: 0.
Frame: 14 -> Processo: 9 -> Page: 0.
Frame: 15 -> Processo: 1 -> Page: 2.
LIVRES: 0
CHEIOS: 16
MEMORIA 2
Frame: 0 -> Processo: 0 -> Page: 1.
Frame: 1 -> Processo: 1 -> Page: 0.
Frame: 2 -> Processo: 0 -> Page: 0.
Frame: 3 Vazio
Frame: 4 Vazio
Frame: 5 Vazio
Frame: 6 Vazio
Frame: 7 Vazio
Frame: 8 Vazio
Frame: 9 Vazio
Frame: 10 Vazio
Frame: 11 Vazio
Frame: 12 Vazio
Frame: 13 Vazio
Frame: 14 Vazio
Frame: 15 Vazio
Frame: 16 Vazio
Frame: 17 Vazio
Frame: 18 Vazio
Frame: 19 Vazio
Frame: 20 Vazio
Frame: 21 Vazio
Frame: 22 Vazio
Frame: 23 Vazio
Frame: 24 Vazio
Frame: 25 Vazio
Frame: 26 Vazio
Frame: 27 Vazio
Frame: 28 Vazio
Frame: 29 Vazio
Frame: 30 Vazio
Frame: 31 Vazio
Frame: 32 Vazio
Frame: 33 Vazio
Frame: 34 Vazio
Frame: 35 Vazio
Frame: 36 Vazio
Frame: 37 Vazio
Frame: 38 Vazio
Frame: 39 Vazio
Frame: 40 Vazio
Frame: 41 Vazio
Frame: 42 Vazio
Frame: 43 Vazio
Frame: 44 Vazio
Frame: 45 Vazio
Frame: 46 Vazio
Frame: 47 Vazio
Frame: 48 Vazio
Frame: 49 Vazio
Frame: 50 Vazio
Frame: 51 Vazio
Frame: 52 Vazio
Frame: 53 Vazio
Frame: 54 Vazio
Frame: 55 Vazio
Frame: 56 Vazio
Frame: 57 Vazio
Frame: 58 Vazio
Frame: 59 Vazio
Frame: 60 Vazio
Frame: 61 Vazio
Frame: 62 Vazio
Frame: 63 Vazio
LRUF - Last Recent Used Frames: [new .. old]
Recente -> 2 1 0 15 14 13 12 11 10 9 8 7 6 5 4 3 -> Proximo a ser removido
Numero da Pagina 0 1 2 3 4 5 6 7 8 9
Paginas do processo 0 esta alocado nos seguintes frames: -1 -1 7 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 1 esta alocado nos seguintes frames: -1 5 15 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 2 esta alocado nos seguintes frames: 3 10 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 3 esta alocado nos seguintes frames: 4 12 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 4 esta alocado nos seguintes frames: 6 1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 5 esta alocado nos seguintes frames: 8 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 6 esta alocado nos seguintes frames: 9 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 7 esta alocado nos seguintes frames: 11 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 8 esta alocado nos seguintes frames: 13 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 9 esta alocado nos seguintes frames: 14 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 10 esta alocado nos seguintes frames: 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 11 esta alocado nos seguintes frames: 2 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 12 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 13 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 14 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1


--->Entrando com PID: 6 e Pagina: 1
... A memoria esta cheia
Estamos executando os processos:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Os seguintes processos terminaram:
MEMORIA 1
MEMORIA 2
Frame: 0 -> Processo: 10 -> Page: 0.
Frame: 1 -> Processo: 4 -> Page: 1.
Frame: 2 -> Processo: 11 -> Page: 0.
Frame: 3 -> Processo: 12 -> Page: 0.
Frame: 4 -> Processo: 0 -> Page: 3.
Frame: 5 -> Processo: 13 -> Page: 0.
Frame: 6 -> Processo: 5 -> Page: 1.
Frame: 7 -> Processo: 14 -> Page: 0.
Frame: 8 -> Processo: 6 -> Page: 1.
Frame: 9 -> Processo: 6 -> Page: 0.
Frame: 10 -> Processo: 2 -> Page: 1.
Frame: 11 -> Processo: 7 -> Page: 0.
Frame: 12 -> Processo: 3 -> Page: 1.
Frame: 13 -> Processo: 8 -> Page: 0.
Frame: 14 -> Processo: 9 -> Page: 0.
Frame: 15 -> Processo: 1 -> Page: 2.
LIVRES: 0
CHEIOS: 16
Frame: 0 -> Processo: 5 -> Page: 0.
Frame: 1 -> Processo: 0 -> Page: 2.
Frame: 2 -> Processo: 4 -> Page: 0.
Frame: 3 -> Processo: 1 -> Page: 1.
Frame: 4 -> Processo: 3 -> Page: 0.
Frame: 5 -> Processo: 2 -> Page: 0.
Frame: 6 -> Processo: 0 -> Page: 1.
Frame: 7 -> Processo: 1 -> Page: 0.
Frame: 8 -> Processo: 0 -> Page: 0.
Frame: 9 Vazio
Frame: 10 Vazio
Frame: 11 Vazio
Frame: 12 Vazio
Frame: 13 Vazio
Frame: 14 Vazio
Frame: 15 Vazio
Frame: 16 Vazio
Frame: 17 Vazio
Frame: 18 Vazio
Frame: 19 Vazio
Frame: 20 Vazio
Frame: 21 Vazio
Frame: 22 Vazio
Frame: 23 Vazio
Frame: 24 Vazio
Frame: 25 Vazio
Frame: 26 Vazio
Frame: 27 Vazio
Frame: 28 Vazio
Frame: 29 Vazio
Frame: 30 Vazio
Frame: 31 Vazio
Frame: 32 Vazio
Frame: 33 Vazio
Frame: 34 Vazio
Frame: 35 Vazio
Frame: 36 Vazio
Frame: 37 Vazio
Frame: 38 Vazio
Frame: 39 Vazio
Frame: 40 Vazio
Frame: 41 Vazio
Frame: 42 Vazio
Frame: 43 Vazio
Frame: 44 Vazio
Frame: 45 Vazio
Frame: 46 Vazio
Frame: 47 Vazio
Frame: 48 Vazio
Frame: 49 Vazio
Frame: 50 Vazio
Frame: 51 Vazio
Frame: 52 Vazio
Frame: 53 Vazio
Frame: 54 Vazio
Frame: 55 Vazio
Frame: 56 Vazio
Frame: 57 Vazio
Frame: 58 Vazio
Frame: 59 Vazio
Frame: 60 Vazio
Frame: 61 Vazio
Frame: 62 Vazio
Frame: 63 Vazio
LRUF - Last Recent Used Frames: [new .. old]
Recente -> 8 7 6 5 4 3 2 1 0 15 14 13 12 11 10 9 -> Proximo a ser removido
Numero da Pagina
0 1 2 3 4 5 6 7 8 9
Paginas do processo 0 esta alocado nos seguintes frames: -1 -1 -1 4 -1 -1 -1 -1 -1 -1
Paginas do processo 1 esta alocado nos seguintes frames: -1 -1 15 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 2 esta alocado nos seguintes frames: -1 10 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 3 esta alocado nos seguintes frames: -1 12 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 4 esta alocado nos seguintes frames: -1 1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 5 esta alocado nos seguintes frames: -1 6 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 6 esta alocado nos seguintes frames: 9 8 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 7 esta alocado nos seguintes frames: 11 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 8 esta alocado nos seguintes frames: 13 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 9 esta alocado nos seguintes frames: 14 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 10 esta alocado nos seguintes frames: 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 11 esta alocado nos seguintes frames: 2 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 12 esta alocado nos seguintes frames: 3 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 13 esta alocado nos seguintes frames: 5 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 14 esta alocado nos seguintes frames: 7 -1 -1 -1 -1 -1 -1 -1 -1 -1


--->Entrando com PID: 12 e Pagina: 3
... A memoria esta cheia
*MEMORIA ESTOURADA - DEVEMOS TRATAR?
Estamos executando os processos:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Os seguintes processos terminaram:
MEMORIA 1
MEMORIA 2
Frame: 0 -> Processo: 12 -> Page: 2.
Frame: 1 -> Processo: 8 -> Page: 3.
Frame: 2 -> Processo: 2 -> Page: 6.
Frame: 3 -> Processo: 13 -> Page: 2.
Frame: 4 -> Processo: 0 -> Page: 9.
Frame: 5 -> Processo: 9 -> Page: 3.
Frame: 6 -> Processo: 14 -> Page: 2.
Frame: 7 -> Processo: 6 -> Page: 4.
Frame: 8 -> Processo: 4 -> Page: 5.
Frame: 9 -> Processo: 10 -> Page: 3.
Frame: 10 -> Processo: 12 -> Page: 3.
Frame: 11 -> Processo: 1 -> Page: 8.
Frame: 12 -> Processo: 7 -> Page: 4.
Frame: 13 -> Processo: 11 -> Page: 3.
Frame: 14 -> Processo: 3 -> Page: 6.
Frame: 15 -> Processo: 5 -> Page: 5.
LIVRES: 0
CHEIOS: 16
Frame: 0 -> Processo: 5 -> Page: 4.
Frame: 1 -> Processo: 3 -> Page: 5.
Frame: 2 -> Processo: 11 -> Page: 2.
Frame: 3 -> Processo: 7 -> Page: 3.
Frame: 4 -> Processo: 1 -> Page: 7.
Frame: 5 -> Processo: 10 -> Page: 2.
Frame: 6 -> Processo: 4 -> Page: 4.
Frame: 7 -> Processo: 6 -> Page: 3.
Frame: 8 -> Processo: 14 -> Page: 1.
Frame: 9 -> Processo: 9 -> Page: 2.
Frame: 10 -> Processo: 0 -> Page: 8.
Frame: 11 -> Processo: 13 -> Page: 1.
Frame: 12 -> Processo: 2 -> Page: 5.
Frame: 13 -> Processo: 8 -> Page: 2.
Frame: 14 -> Processo: 12 -> Page: 1.
Frame: 15 -> Processo: 5 -> Page: 3.
Frame: 16 -> Processo: 3 -> Page: 4.
Frame: 17 -> Processo: 11 -> Page: 1.
Frame: 18 -> Processo: 7 -> Page: 2.
Frame: 19 -> Processo: 1 -> Page: 6.
Frame: 20 -> Processo: 10 -> Page: 1.
Frame: 21 -> Processo: 4 -> Page: 3.
Frame: 22 -> Processo: 6 -> Page: 2.
Frame: 23 -> Processo: 14 -> Page: 0.
Frame: 24 -> Processo: 9 -> Page: 1.
Frame: 25 -> Processo: 0 -> Page: 7.
Frame: 26 -> Processo: 13 -> Page: 0.
Frame: 27 -> Processo: 2 -> Page: 4.
Frame: 28 -> Processo: 8 -> Page: 1.
Frame: 29 -> Processo: 12 -> Page: 0.
Frame: 30 -> Processo: 5 -> Page: 2.
Frame: 31 -> Processo: 3 -> Page: 3.
Frame: 32 -> Processo: 11 -> Page: 0.
Frame: 33 -> Processo: 7 -> Page: 1.
Frame: 34 -> Processo: 1 -> Page: 5.
Frame: 35 -> Processo: 10 -> Page: 0.
Frame: 36 -> Processo: 4 -> Page: 2.
Frame: 37 -> Processo: 6 -> Page: 1.
Frame: 38 -> Processo: 9 -> Page: 0.
Frame: 39 -> Processo: 0 -> Page: 6.
Frame: 40 -> Processo: 2 -> Page: 3.
Frame: 41 -> Processo: 8 -> Page: 0.
Frame: 42 -> Processo: 5 -> Page: 1.
Frame: 43 -> Processo: 3 -> Page: 2.
Frame: 44 -> Processo: 7 -> Page: 0.
Frame: 45 -> Processo: 1 -> Page: 4.
Frame: 46 -> Processo: 4 -> Page: 1.
Frame: 47 -> Processo: 6 -> Page: 0.
Frame: 48 -> Processo: 0 -> Page: 5.
Frame: 49 -> Processo: 2 -> Page: 2.
Frame: 50 -> Processo: 5 -> Page: 0.
Frame: 51 -> Processo: 3 -> Page: 1.
Frame: 52 -> Processo: 1 -> Page: 3.
Frame: 53 -> Processo: 4 -> Page: 0.
Frame: 54 -> Processo: 2 -> Page: 1.
Frame: 55 -> Processo: 3 -> Page: 0.
Frame: 56 -> Processo: 1 -> Page: 2.
Frame: 57 -> Processo: 0 -> Page: 3.
Frame: 58 -> Processo: 2 -> Page: 0.
Frame: 59 -> Processo: 1 -> Page: 1.
Frame: 60 -> Processo: 0 -> Page: 2.
Frame: 61 -> Processo: 1 -> Page: 0.
Frame: 62 -> Processo: 0 -> Page: 1.
Frame: 63 -> Processo: 0 -> Page: 0.
LRUF - Last Recent Used Frames: [new .. old]
Recente -> 10 15 14 13 12 11 9 8 7 6 5 4 3 2 1 0 -> Proximo a ser removido
Numero da Pagina 0 1 2 3 4 5 6 7 8 9
Paginas do processo 0 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 4
Paginas do processo 1 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 11 -1
Paginas do processo 2 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 2 -1 -1 -1
Paginas do processo 3 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 14 -1 -1 -1
Paginas do processo 4 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 8 -1 -1 -1 -1
Paginas do processo 5 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 15 -1 -1 -1 -1
Paginas do processo 6 esta alocado nos seguintes frames: -1 -1 -1 -1 7 -1 -1 -1 -1 -1
Paginas do processo 7 esta alocado nos seguintes frames: -1 -1 -1 -1 12 -1 -1 -1 -1 -1
Paginas do processo 8 esta alocado nos seguintes frames: -1 -1 -1 1 -1 -1 -1 -1 -1 -1
Paginas do processo 9 esta alocado nos seguintes frames: -1 -1 -1 5 -1 -1 -1 -1 -1 -1
Paginas do processo 10 esta alocado nos seguintes frames: -1 -1 -1 9 -1 -1 -1 -1 -1 -1
Paginas do processo 11 esta alocado nos seguintes frames: -1 -1 -1 13 -1 -1 -1 -1 -1 -1
Paginas do processo 12 esta alocado nos seguintes frames: -1 -1 0 10 -1 -1 -1 -1 -1 -1
Paginas do processo 13 esta alocado nos seguintes frames: -1 -1 3 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 14 esta alocado nos seguintes frames: -1 -1 6 -1 -1 -1 -1 -1 -1 -1


---->Parando processo com PID: 1
Estamos executando os processos:
2 3 4 5 6 7 8 9 10 11 12 13 14
Os seguintes processos terminaram:
0 1
MEMORIA 1
Frame: 0 -> Processo: 13 -> Page: 4.
Frame: 1 -> Processo: 9 -> Page: 5.
Frame: 2 -> Processo: 14 -> Page: 4.
Frame: 3 -> Processo: 4 -> Page: 7.
Frame: 4 -> Processo: 6 -> Page: 6.
Frame: 5 -> Processo: 10 -> Page: 5.
Frame: 6 -> Processo: 4 -> Page: 6.
Frame: 7 -> Processo: 10 -> Page: 4.
Frame: 8 Vazio
Frame: 9 -> Processo: 7 -> Page: 5.
Frame: 10 -> Processo: 2 -> Page: 8.
Frame: 11 -> Processo: 11 -> Page: 4.
Frame: 12 -> Processo: 3 -> Page: 7.
Frame: 13 -> Processo: 5 -> Page: 6.
Frame: 14 -> Processo: 12 -> Page: 4.
Frame: 15 -> Processo: 8 -> Page: 5.
LIVRES: 1
CHEIOS: 15
MEMORIA 2
Frame: 0 -> Processo: 6 -> Page: 5.
Frame: 1 -> Processo: 14 -> Page: 3.
Frame: 2 -> Processo: 9 -> Page: 4.
Frame: 3 -> Processo: 13 -> Page: 3.
Frame: 4 -> Processo: 2 -> Page: 7.
Frame: 5 -> Processo: 8 -> Page: 4.
Frame: 6 -> Processo: 12 -> Page: 3.
Frame: 7 -> Processo: 5 -> Page: 5.
Frame: 8 -> Processo: 3 -> Page: 6.
Frame: 9 -> Processo: 11 -> Page: 3.
Frame: 10 -> Processo: 7 -> Page: 4.
Frame: 11 -> Processo: 1 -> Page: 8.
Frame: 12 -> Processo: 10 -> Page: 3.
Frame: 13 -> Processo: 4 -> Page: 5.
Frame: 14 -> Processo: 6 -> Page: 4.
Frame: 15 -> Processo: 14 -> Page: 2.
Frame: 16 -> Processo: 9 -> Page: 3.
Frame: 17 -> Processo: 13 -> Page: 2.
Frame: 18 -> Processo: 2 -> Page: 6.
Frame: 19 -> Processo: 8 -> Page: 3.
Frame: 20 -> Processo: 12 -> Page: 2.
Frame: 21 -> Processo: 5 -> Page: 4.
Frame: 22 -> Processo: 3 -> Page: 5.
Frame: 23 -> Processo: 11 -> Page: 2.
Frame: 24 -> Processo: 7 -> Page: 3.
Frame: 25 -> Processo: 1 -> Page: 7.
Frame: 26 -> Processo: 10 -> Page: 2.
Frame: 27 -> Processo: 4 -> Page: 4.
Frame: 28 -> Processo: 6 -> Page: 3.
Frame: 29 -> Processo: 14 -> Page: 1.
Frame: 30 -> Processo: 9 -> Page: 2.
Frame: 31 -> Processo: 0 -> Page: 8.
Frame: 32 -> Processo: 13 -> Page: 1.
Frame: 33 -> Processo: 2 -> Page: 5.
Frame: 34 -> Processo: 8 -> Page: 2.
Frame: 35 -> Processo: 12 -> Page: 1.
Frame: 36 -> Processo: 5 -> Page: 3.
Frame: 37 -> Processo: 3 -> Page: 4.
Frame: 38 -> Processo: 11 -> Page: 1.
Frame: 39 -> Processo: 7 -> Page: 2.
Frame: 40 -> Processo: 1 -> Page: 6.
Frame: 41 -> Processo: 10 -> Page: 1.
Frame: 42 -> Processo: 4 -> Page: 3.
Frame: 43 -> Processo: 6 -> Page: 2.
Frame: 44 -> Processo: 14 -> Page: 0.
Frame: 45 -> Processo: 9 -> Page: 1.
Frame: 46 -> Processo: 0 -> Page: 7.
Frame: 47 -> Processo: 13 -> Page: 0.
Frame: 48 -> Processo: 2 -> Page: 4.
Frame: 49 -> Processo: 8 -> Page: 1.
Frame: 50 -> Processo: 12 -> Page: 0.
Frame: 51 -> Processo: 5 -> Page: 2.
Frame: 52 -> Processo: 3 -> Page: 3.
Frame: 53 -> Processo: 11 -> Page: 0.
Frame: 54 -> Processo: 7 -> Page: 1.
Frame: 55 -> Processo: 1 -> Page: 5.
Frame: 56 -> Processo: 10 -> Page: 0.
Frame: 57 -> Processo: 4 -> Page: 2.
Frame: 58 -> Processo: 6 -> Page: 1.
Frame: 59 -> Processo: 9 -> Page: 0.
Frame: 60 -> Processo: 0 -> Page: 6.
Frame: 61 -> Processo: 2 -> Page: 3.
Frame: 62 -> Processo: 8 -> Page: 0.
Frame: 63 -> Processo: 5 -> Page: 1.
LRUF - Last Recent Used Frames: [new .. old]
Recente -> 5 3 4 2 1 0 10 15 14 13 12 11 9 8 7 6 -> Proximo a ser removido
Numero da Pagina 0 1 2 3 4 5 6 7 8 9
Paginas do processo 0 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 1 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
Paginas do processo 2 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 -1 10 -1
Paginas do processo 3 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 -1 12 -1 -1
Paginas do processo 4 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 6 3 -1 -1
Paginas do processo 5 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 13 -1 -1 -1
Paginas do processo 6 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 -1 4 -1 -1 -1
Paginas do processo 7 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 9 -1 -1 -1 -1
Paginas do processo 8 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 15 -1 -1 -1 -1
Paginas do processo 9 esta alocado nos seguintes frames: -1 -1 -1 -1 -1 1 -1 -1 -1 -1
Paginas do processo 10 esta alocado nos seguintes frames: -1 -1 -1 -1 7 5 -1 -1 -1 -1
Paginas do processo 11 esta alocado nos seguintes frames: -1 -1 -1 -1 11 -1 -1 -1 -1 -1
Paginas do processo 12 esta alocado nos seguintes frames: -1 -1 -1 -1 14 -1 -1 -1 -1 -1
Paginas do processo 13 esta alocado nos seguintes frames: -1 -1 -1 -1 0 -1 -1 -1 -1 -1
Paginas do processo 14 esta alocado nos seguintes frames: -1 -1 -1 -1 2 -1 -1 -1 -1 -1

Conclusão

Tratamos como memoria principal o local aonde os processos alocam suas paginas e memoria virtual o local para onde as paginas vão após o “swap out”, isso descreve apenas a nomenclatura das variáveis usadas, podendo ser entendidas como memoria 1 e memoria 2 ou mesmo como memoria virtual e disco por exemplo.

Desde o inicio criamos variáveis que definissem o tamanho da memoria, o tempo de intervalo entre alocação de paginas, o tamanho do workingset de cada processo, etc. Todas essas variáveis que encabeçam o nosso arquivo memoria.h são importantes para testes e alterações, assim podemos tratar todos os tipos e tamanhos de estruturas.

Nos prints mostrados acima usamos tamanhos menores para que tudo ficasse constantemente organizado na tela.

SLEEP_TIME	Tempo entre a inserção de novas páginas na memória P.
FRAME_LIMIT	Tamanho da memória principal (Qtdade de frames)
MAIN_MEMORY_SIZE	Tamanho da memória principal (Qtdade de frames)
VIRTUAL_MEMORY_SIZE	Tamanho da memória secundária
THREAD_LIMIT	Quantidade de processos a serem criados
PAGE_LIMIT	Quantidade de páginas que cada processo possui
WORKSET_LIMIT	Quantidade de paginas que cada processo pode alocar na memoria simultâneamente

Ao iniciar criamos dois mutex que são úteis para manter o sincronismo e a validação dos dados do programa. Pois como cada processo usa uma thread independente, poderia haver acesso simultâneo a áreas importantes do programa.

Desta forma, isolamos áreas para que sejam acessadas por apenas um processo de cada vez, áreas como o acesso as memorias, a fila de processos e a fila de frames usados recentemente (LRUF - Least Recent Used Frame).

Cada processo é criado por uma thread e roda independente de seus irmãos.
E esperado (join) pelo fluxo principal que só acaba após que todos os seus filhos tenha retornado.
De tempos em tempos o processo solicita um frame da memória para que possa alocar suas novas páginas.

A parte mais complexa do programa é o retorno após a requisição de páginas, esse retorno leva em consideração alguns requisitos.
Primeiro, o gerenciados só pode liberar um frame para o processo se ele estiver usando menos espaço do que o limitador de working set permite, ou seja, se o processo estiver alocando o limite de frames definido pela variável **WORKSET_LIMIT** ele escolher dentre os frames utilizados um para que a nova página entre na memória. Um detalhe interessante é

que todos os processos gravam consigo a lista de frames utilizados por seus frames.
Em seguida caso o processo ainda esteja abaixo do seu limite, devemos verificar se ainda há memória livre.
Em caso positivo basta procurar um frames vazio e alocar a página.
Porém se a memória estiver cheia o gerenciador precisa saber qual frame tem prioridade para ficar e qual tem que sair, para isso implementamos a fila chamada LRUF.

LRUF ou recent_frame[ ] é uma fila cíclica onde a primeira posição indica o frame recém utilizado e a ultima posição indica o frame mais indicado para sair caso seja necessário.

Assim a cada alteração na memória principal devemos atualizar esta fila, tomando cuidado para que nenhum frame apareça mais de uma vez, ou seja a cada requisição de página concluída a fila LRUF deve ser atualizada com um “shift-right” e na primeira posição colocado o frame obtido pela requisição.

Após todas as suas páginas terem passado pela memória principal, o processo se encerra.
Ao se encerrar ele limpa da memória principal as suas páginas e libera espaço para os novos processos.

Notamos que se deixarmos um tempo de espera entre requisições muito pequeno os processos inserem muitas páginas de uma única vez na memória o que aumenta o trashing, pois atingem mais facilmente o seu working set, ao mesmo tempo que se deixarmos um tempo muito pequeno e um número de páginas pequeno o processo inicia e se encerra rapidamente , deixa a memória sempre livre e não havendo conflitos com outros processos.
Porém ao esticar bastante o tempo, a concorrência aumenta logo aumentamos a probabilidades do processo encontrar a memória sempre cheia.