

Trabalho Prático 2

Sistema Operacionais

Camila Guimarães - 2256

Marcos Mota - 2295

Samuel Jhonata - 2282

Vinícius Kodama - 2259



ROTEIRO

1. Introdução
2. Desenvolvimento
3. Decisões de Código
4. Código
5. Execução
6. Conclusão

1

Introdução

Um pouco sobre o que é o trabalho



INTRODUÇÃO

Este trabalho consiste na simulação de um gerenciador de processos, onde algumas funções devem ser exploradas como

- Criar e executar processos
- Substituir processo atual com uma cópia dele
- Mudança de estado do processo
- Escalonar processos
- Troca de contexto

2

Desenvolvimento

Com o que produzimos?



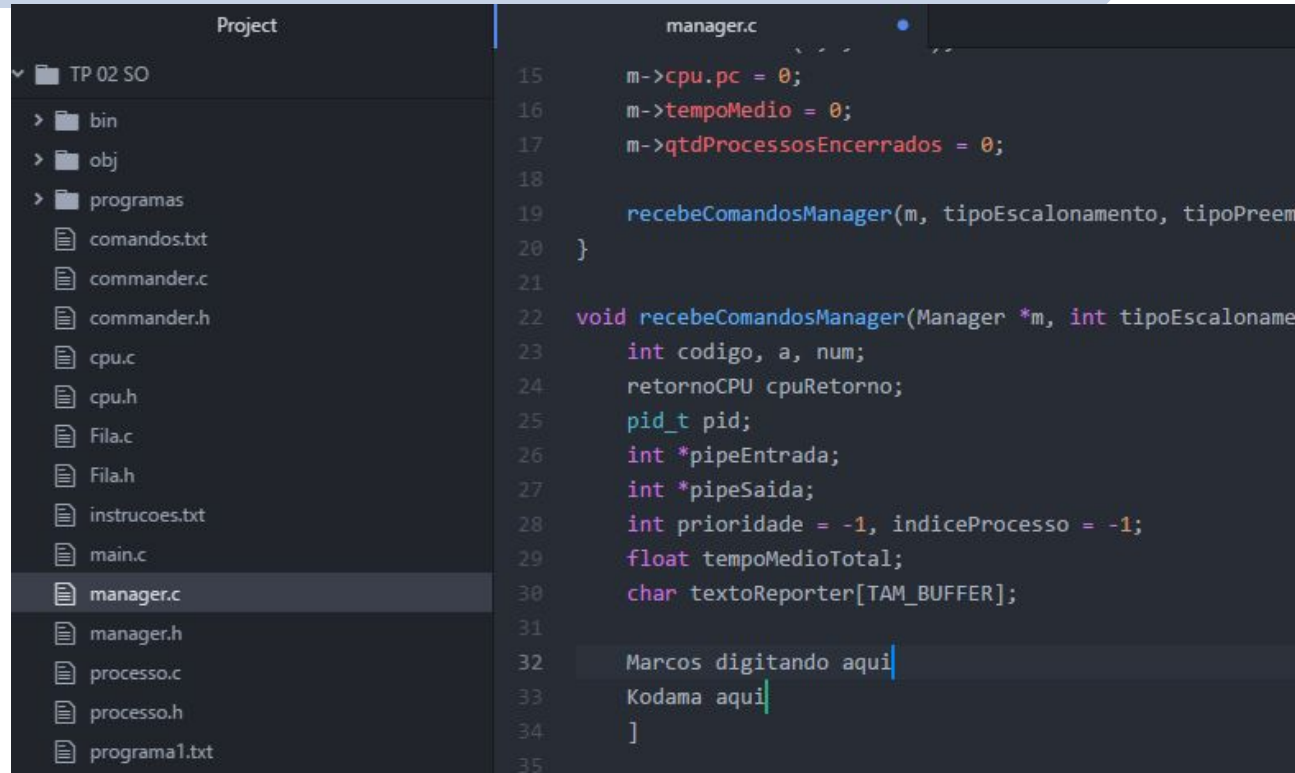
DESENVOLVIMENTO

- Desenvolvimento no ambiente Ubuntu
- Linguagem C
- IDE: CodeBlocks
- Sistema de edição compartilhada em tempo real: ATOM e Teletype

ubuntu[®]

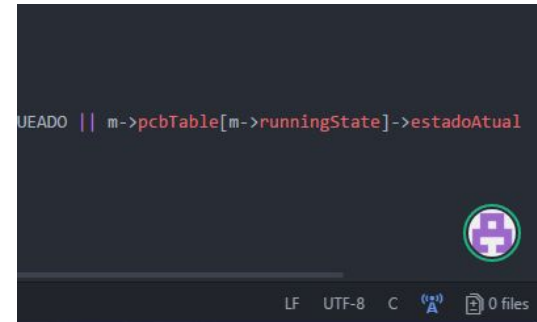


DESENVOLVIMENTO



The screenshot shows a code editor with a project structure on the left and the implementation of a manager module in the center. The project structure includes a folder 'TP 02 SO' with subfolders 'bin', 'obj', and 'programas', and several files including 'comandos.txt', 'commander.c', 'commander.h', 'cpu.c', 'cpu.h', 'Fila.c', 'Fila.h', 'instrucoes.txt', 'main.c', 'manager.c', 'manager.h', 'processo.c', 'processo.h', and 'programa1.txt'. The 'manager.c' file is selected, showing the following code:

```
15     m->cpu.pc = 0;
16     m->tempoMedio = 0;
17     m->qtdProcessosEncerrados = 0;
18
19     recebeComandosManager(m, tipoEscalonamento, tipoPreem
20 }
21
22 void recebeComandosManager(Manager *m, int tipoEscaloname
23     int codigo, a, num;
24     retornoCPU cpuRetorno;
25     pid_t pid;
26     int *pipeEntrada;
27     int *pipeSaida;
28     int prioridade = -1, indiceProcesso = -1;
29     float tempoMedioTotal;
30     char textoReporter[TAM_BUFFER];
31
32     Marcos digitando aqui
33     Kodama aqui
34 ]
35
```



The screenshot shows a terminal window with a dark background. The command being executed is `m->pcbTable[m->runningState]->estadoAtual`. The output of the command is `UEADO`. The terminal window has a status bar at the bottom showing `LF UTF-8 C` and `0 files`.

3

Decisões de Código

Escolhas no planejamento do projeto



DECISÕES DE CÓDIGO

- Fork para Commander e Manager, não para Reporter
Não foi possível para Reporter por lixo de memória
- Comunicação entre Commander e Manager através de pipes
- Ou seja, executam em paralelo



DECISÕES DE CÓDIGO

- Classe de prioridade de 0 a 3 (sendo 3 a mais alta)
- Prioridade aumenta ao ser escalonado antes do fim do seu tempo de execução
- Prioridade diminui ao ser bloqueado antes do fim do seu tempo de execução



DECISÕES DE CÓDIGO

- Estrutura de filas para simulação da memória, lista de prontos/bloqueados
- Processos salvos num vetor de processos
- Single Core



DECISÕES DE CÓDIGO

- Preemptivo e não preemptivo
- Commander executando por terminal ou leitura de arquivo

Ao executar por arquivo é possível escolher o tempo entre comandos através do `wait()`



ESCALONAMENTO

- Escalonamento de prioridade
- Escalonamento de relógio
- Ambos preemptivo ou não preemptivo

4

Código

Código produzido no trabalho



CÓDIGO

Composto de 7 arquivos

- Commander
- Manager
- CPU
- Fila
- Main
- Processo
- Reporter



CÓDIGO - Main

```
int main()
{
    Commander commander;

    int tipoEscalonamento = -1;
    int tipoPreemptivo = -1;
    int tipoEntrada = -1;

    printf("----- M E N U -----\n");
    printf("Escolha do Escalonamento:\n");
    printf("1- Lista de Prioridade\n");
    printf("2- Relógio\n");

    while (tipoEscalonamento < 1 || tipoEscalonamento > 2) {
        scanf("%d", &tipoEscalonamento);
    }

    printf("Tipo de Escalonamento:\n");
    printf("1- Preemptivo\n");
    printf("2- Não Preemptivo\n");

    while (tipoPreemptivo < 1 || tipoPreemptivo > 2) {
        scanf("%d", &tipoPreemptivo);
    }

    printf("Tipo Entrada de Comandos:\n");
    printf("1- Prompt de Comandos\n");
    printf("2- Arquivo\n");

    while (tipoEntrada < 1 || tipoEntrada > 2) {
        scanf("%d", &tipoEntrada);
    }

    inicializaCommander(&commander, tipoEscalonamento, tipoPreemptivo, tipoEntrada);

    return 0;
```




CÓDIGO - Commander

```
1  #ifndef COMMANDER_H_INCLUDED
2  #define COMMANDER_H_INCLUDED
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include "manager.h"
7
8  typedef struct Commander{
9      int pipeEntrada[2];
10     int pipeSaida[2];
11     Manager manager;
12 }Commander;
13
14 int carregaComandos(String nomeArq, char *retorno);
15 void inicializaCommander(Commander *c, int tipoEscalonamento, int tipoPreemptivo, int tipoEntrada);
16 void recebeComandos(Commander *c, int tipoEntrada);
17 int executaComando(Commander *c, char comando);
18
19 #endif // COMMANDER_H_INCLUDED
20
```



CÓDIGO - Manager

```
25 typedef struct Manager{
26     int *pipeEntrada;
27     int *pipeSaida;
28     int pipeRepEntrada[2];
29     int pipeRepSaida[2];
30     int time;
31     Cpu cpu;
32     Processo* pcbTable[QTD_PROCESSOS_MAX]; //vetor de processos
33     TLista readyState;
34     TLista blockedState;
35     int runningState;
36     int contadorID;
37     int ultimoExecutado;
38     int tempoMedio;
39     int qtdProcessosEncerrados;
40 } Manager;
41
42 void inicializaManager(Manager *m, int tipoEscalonamento, int tipoPreemptivo);
43 void recebeComandosManager(Manager *m, int tipoEscalonamento, int tipoPreemptivo);
44 void inicializaEstruturas(Manager *m);
45 int criaProcesso(Manager *m, int id, int prioridade, String nomeArq);
46 void criaProcessoFilho(Manager *m, Processo *pai);
47 int insereListaPronto(Manager *m, int indice, int prioridade);
48 int insereListaBloqueado(Manager *m, int indice, int prioridade);
49 void trocaContexto(Manager *m, int fatiaTempo, int indexProcessoColocar);
50 int escalonar(Manager *m, int tipoEscalonamento);
```



CÓDIGO - Processo

```
14 typedef char* String;
15 typedef int Estado;
16
17 typedef struct Processo{
18     int id; //id do processo
19     int idPai;
20     int pc; //contador de programa
21     int variavel; //variavel a ser manipulada
22     int prioridade;
23     Estado estadoAtual; //estado atual do processo
24     int tempoInicio;
25     int tempoCPU;
26     char texto[TAM_TEXTO][20]; //codigo do programa
27     int qtdInst; //conta total de comandos
28
29 }Processo;
30
31 int carregaInstrucoes(Processo *p, String nomeArq);
32
33 #endif // PROCESSO_H_INCLUDED
34
```



CÓDIGO - Fila

```
10 typedef int TipoChave;
11
12 typedef struct {
13     TipoChave indiceProcesso;
14     int prioridade;
15     /* outros componentes */
16 } TItem;
17
18 typedef struct Celula* Apontador;
19
20 typedef struct Celula {
21     TItem Item;
22     struct Celula* pProx; /* Apontador pProx; */
23 } TCelula;
24
25 typedef struct {
26     Apontador pPrimeiro;
27     Apontador pUltimo;
28     int Contador;
29 } TLista;
30
31
32 void FLVazia(TLista* pLista);
33 int LEhVazia(TLista* pLista);
34 void LInsere(TLista *pLista, TItem* pItem);
35 //int LRetira(TLista* pLista, TItem* pItem);
36 int LRetira (TLista* pLista, int index);
37 void LImprime(TLista* pLista);
38 void LBuscaPrimeiro (TLista* pLista, int *indiceProcesso, int *prioridade);
39
40
41
42 #endif // FILA_H_INCLUDED
```



CÓDIGO - CPU

```
12  typedef struct Cpu{
13      int time;
14      int variavel;
15      int pc;
16      int fatiaTempo; //qtd de tempo máxima para o processo executar
17      int tempoUsado; //tempo já usado na execução
18  }Cpu;
19
20  typedef struct retornoCPU{
21      int comando;
22      int n;
23      char arquivo[30];
24
25  }retornoCPU;
26  void executaProxInst(Processo *p, Cpu *cpu, retornoCPU* retorno);
27  void executaInstrucao(Processo *p, String instrucao, Cpu *cpu, retornoCPU* retorno);
28
29  #endif // CPU_H_INCLUDED
```



CÓDIGO - Reporter

```
7  typedef struct Reporter{
8      int *pipeEntrada;
9      int *pipeSaida;
10
11  }Reporter;
12
13  void imprimeEstado(Manager *m, char* retorno);
14
15  #endif // REPORTER_H_INCLUDED
16  |
```

5

Execução

Hora de executar o trabalho



EXECUÇÃO

TP 02 SO

```
usuario@computador:~$ q
```

Comando Q

Executando instrução: S 10000

```
usuario@computador:~$ p
```

Comando P

-----INFORMAÇÕES DO SISTEMA-----

Tempo do Sistema: 23

Processos:

0 - ID:0; ID Pai: -1; Prioridade: 2; PC: 18; Tempo Início: 0; Tempo CPU: 14; Variavel: 1100

1 - ID:2; ID Pai: 0; Prioridade: 3; PC: 11; Tempo Início: 15; Tempo CPU: 0; Variavel: 1060

PROCESSO EM EXECUÇÃO:

1 - ID:2; ID Pai: 0; Prioridade: 3; Variavel: 1060

PROCESSOS PRONTOS:

Processos com prioridade 0

Processos com prioridade 1

Processos com prioridade 2

Processos com prioridade 3

PROCESSOS BLOQUEADOS:

Processos com prioridade 0

Processos com prioridade 1

Processos com prioridade 2

0 - ID:0; ID Pai: -1; Prioridade: 2; Variavel: 1100

Processos com prioridade 3

Informações da CPU:

PC 1

Variável: 10000

Tempo de Uso: 2

Fatia de tempo: 8

```
usuario@computador:~$ █
```


6

Conclusão

O que aprendemos no trabalho?



CONCLUSÃO

- Trabalho bem extenso e com muitas funcionalidades
- Desenvolvimento de processos simultâneos
- Grande trabalho em equipe
- Alta complexidade.