Universidade Federal de Viçosa - Campus Florestal

Disciplina: Sistemas Operacionais

Professor: Daniel Mendes Barbosa

Trabalho Prático 2

Eduardo Vinicius – 3498

Pablo Ferreira – 3480

Samuel Sena – 3494

Florestal – MG

2020

SUMÁRIO

1. Introdução.....3

2. Decisões do projeto.....4

2.1. Escalonador do Grupo 4

2.2. Processo Impressão.....5

3. Testes de execução.....7

4. Referência Bibliográfica.....13

1. Introdução

O trabalho consiste em criar estruturas e métodos capazes de fazer o gerenciamento de processos, e de suas simulações com respectivas entradas de processos no programa. Realizamos a escrita de todo o trabalho na linguagem de programação C.

Na criação do trabalho e do código de implementação se fez muito importante as chamadas dos sistemas, fork() para a criação de processos filhos , e pipe() para a comunicação entre os processos, tornando possível a construção ambiente de simulação.

Além disso o trabalho especifica algumas decisões que deveriam ser tomadas por o grupo em determinadas funções, essas vão ser mostradas nos tópicos abaixo.

Para compilar o programa, criamos um arquivo *makefile*, portanto basta iniciar uma instancia de um terminal (em algum sistema operacional Linux) e compilar o programa com o uso do comando *make*. Em seguida, para executar o programa, é possível com o uso do comando *make run* ou simplesmente: *./EXEC* .

O código se encontra com diversos comentários que auxiliam no entendimento do funcionamento e fluxo de execução.

2. Decisões do Projeto

2.1 Escalonador do Grupo

O escalador que nosso grupo optou por utilizar a mesma premissa do escalonamento por filas múltiplas, entretanto a mudança que foi feita e que se o processo executar em um tempo menor que sua fatia de tempo destinada, a sua prioridade ira aumentar , e consequentemente em sua próxima execução ele terá um tempo menor na CPU, isso foi feito com objetivo de otimizar o tempo médio de ciclo.

```c
void ExecutaCPU2(Cpu *cpu, Time *time, PcbTable *pcbTable, EstadoEmExec *estadoexec, EstadoBloqueado *estadobloqueado, EstadoPronto *estadopronto, Processo *processo){


  RodaInstrucao(cpu, time, estadoexec, pcbTable, estadobloqueado, estadopronto, processo);

  switch (processo->prioridade){
      case 0:
          cpu->fatiaTempoUsada += 1; break;
      case 1:
          cpu->fatiaTempoUsada += 2; break;
      case 2:
          cpu->fatiaTempoUsada += 4; break;
      case 3:
          cpu->fatiaTempoUsada += 8; break;
      default:
          printf("Erro ao atualizar fatia de tempo CPU!\n");
  }

  //Atualizando processo simulado
  processo->Estado_Processo.Cont = cpu->contadorProgramaAtual;
  processo->CotaCPU = cpu->fatiaTempoUsada;
  processo->Estado_Processo.Alocado_V_inteiros = cpu->Alocado_V_inteiros;
  processo->Estado_Processo.Quant_Inteiros = cpu->Quant_Inteiros;
  for (int i = 0; i < processo->Estado_Processo.Tam; i++) {
      strcpy(processo->Estado_Processo.Programa[i], cpu->programa.instrucoes[i]);
  }
  if(cpu->Alocado_V_inteiros)
      processo->Estado_Processo.Inteiro=cpu->valorInteiro;
  pcbTable->vetor[estadoexec->iPcbTable] = *processo;
  if(cpu->fatiaTempoUsada >= cpu->fatiaTempo){ // caso a fatia ultrapassar a cota estabelecida, programa é bloqueado e aguarda novo escalonamento.
      if(processo->prioridade> 0 && processo->prioridade<=3){
            printf("\n --- Prioridade de processo de PID: %d mudada de %d para %d --- \n",processo->pid,processo->prioridade,pcbTable->vetor[estadoexec->iPcbTable].prioridade+1
            processo->prioridade++;
            pcbTable->vetor[estadoexec->iPcbTable].prioridade++;
          }
      EnfileiraBloqueado(estadobloqueado, processo);
      *processo = ColocaOutroProcessoCPU(cpu,estadopronto);//Ja que o processo atual foi bloqueado, colocaremos outro na CPU
    }
  else if(cpu->fatiaTempoUsada < cpu->fatiaTempo) {
      if(processo->prioridade> 0 && processo->prioridade<=3){
            printf("\n --- Prioridade de processo de PID: %d mudada de %d para %d --- \n",processo->pid,processo->prioridade,pcbTable->vetor[estadoexec->iPcbTable].prioridade-1
            processo->prioridade--;
            pcbTable->vetor[estadoexec->iPcbTable].prioridade--;
          }
  }
   else if(!strcmp(processo->estado,"BLOQUEADO")){ //Caso uma instrucao de bloqueio tenha sido realizada
     *processo = ColocaOutroProcessoCPU(cpu,estadopronto);
```

2.2. Processo Impressão

O nosso grupo optou por trazer as informações, relacionadas a processos na CPU, os processos na fila de pronto, processos na fila de bloqueados, processos na tabela de processos, e também o tempo médio do ciclo, ao termino da simulação, como pode ser visto nos códigos abaixo:

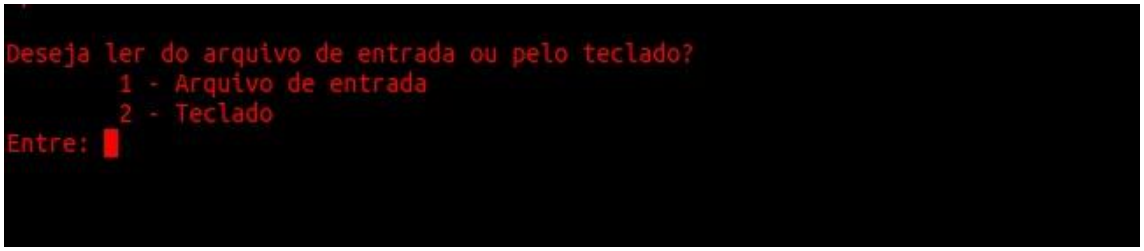
```c
void ImprimeSistemaCompleto(Cpu* cpu, PcbTable *pcbTable, EstadoBloqueado *estadobloqueado, EstadoPronto *estadopronto){
  printf("\n--Inicio impresssao do estado do sistema---\n\n");
  ImprimirCPU(cpu);
  printf("\n\n");
  ImprimePronto(estadopronto);
  printf("\n\n");
  ImprimeBloqueado(estadobloqueado);
  printf("\n\n");
  ImprimePcbTable(pcbTable);
  printf("\n\n--Fim da impresssao do estado do sistema---\n\n");
}

void ImprimeTempoMedioCiclo(Time *time){
  double tempomedio=0;
  tempomedio = time->time/time->QuantProcessosCriados;
  printf("\n---- O tempo de ciclo medio foi de %.2lf ----\n\nFinalizando...\n",tempomedio);
}
```

```c
  void ImprimeBloqueado(EstadoBloqueado *estadobloqueado){
    int Aux;
    printf("\n\tFila de Processsos Bloqueados:\n");
    for (Aux = estadobloqueado->Frente; Aux <= (estadobloqueado->Tras - 1); Aux++){
        printf("----------------- Processo %d -----------------\n",Aux );
        printf("PID: %i\n", estadobloqueado->vetor[Aux].pid);
        printf("PID Pai: %i\n", estadobloqueado->vetor[Aux].pid_do_pai);
        printf("Estado: %s\n", estadobloqueado->vetor[Aux].estado);
        printf("Tempo CPU: %d\n", estadobloqueado->vetor[Aux].CotaCPU);
        printf("Tempo Inicio: %d\n", estadobloqueado->vetor[Aux].startupTime);
        printf("Prioridade: %d\n", estadobloqueado->vetor[Aux].prioridade);
        printf("Quantidade de variaveis: %d\n",estadobloqueado->vetor[Aux].Estado_Processo.Quant_Inteiros);
        if(estadobloqueado->vetor[Aux].Estado_Processo.Alocado_V_inteiros)
          for(int i=0;i<estadobloqueado->vetor[Aux].Estado_Processo.Quant_Inteiros;i++){
              printf("Valor inteiro da variavel ( %d ): %d\n", i,estadobloqueado->vetor[Aux].Estado_Processo.Inteiro[i]);
            }
        printf("Contador de Programa: %d\n", estadobloqueado->vetor[Aux].Estado_Processo.Cont);
        printf("Programa do Processo: \n");
        for (int i = 0; i < estadobloqueado->vetor[Aux].Estado_Processo.Tam; i++)
            printf("\t%s", estadobloqueado->vetor[Aux].Estado_Processo.Programa[i]);
        printf("-----------------------------------------------\n");
    }
    printf("\n\t--Fim da fila de Processos Bloqueados--\n");
  }
  void ImprimePcbTable(PcbTable *pcbTable){
    int Aux;
    printf("\n\tLista de Processos na Tabela:\n");
    for (Aux = pcbTable->Primeiro; Aux <= (pcbTable->Ultimo - 1); Aux++) {
        printf("----------------- Processo %d -----------------\n",Aux );
        printf("PID: %i\n", pcbTable->vetor[Aux].pid);
        printf("PID Pai: %i\n", pcbTable->vetor[Aux].pid_do_pai);
        printf("Estado: %s\n", pcbTable->vetor[Aux].estado);
        printf("Tempo CPU: %d\n", pcbTable->vetor[Aux].CotaCPU);
        printf("Tempo Inicio: %d\n", pcbTable->vetor[Aux].startupTime);
        printf("Prioridade: %d\n", pcbTable->vetor[Aux].prioridade);
        printf("Quantidade de variaveis: %d\n",pcbTable->vetor[Aux].Estado_Processo.Quant_Inteiros);
        if(pcbTable->vetor[Aux].Estado_Processo.Alocado_V_inteiros)
          for(int i=0;i<pcbTable->vetor[Aux].Estado_Processo.Quant_Inteiros;i++){
              printf("Valor inteiro da variavel ( %d ): %d\n", i,pcbTable->vetor[Aux].Estado_Processo.Inteiro[i]);
          }
        printf("Contador de Programa: %d\n", pcbTable->vetor[Aux].Estado_Processo.Cont);
        printf("Programa do Processo: \n");
        for (int i = 0; i < pcbTable->vetor[Aux].Estado_Processo.Tam; i++)
            printf("\t%s", pcbTable->vetor[Aux].Estado_Processo.Programa[i]);
        printf("-----------------------------------------------\n");
    }
    printf("\n\t--Fim da lista de Processos na Tabela--\n");
```

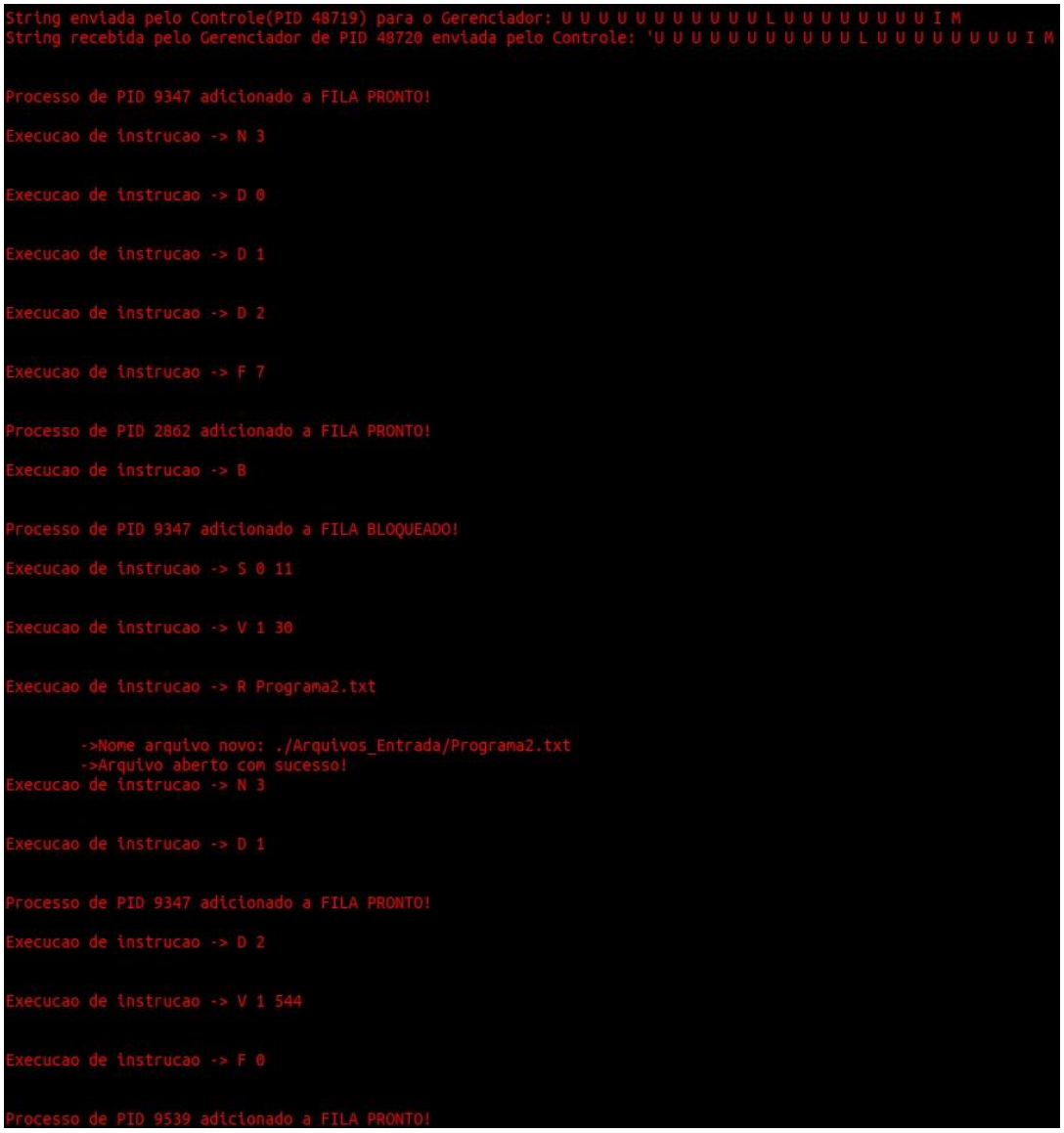
```c
void ImprimirCPU(Cpu *cpu){
      printf("\n\tInformacoes da CPU: \n");
      printf("\nPrograma na CPU: \n");
      for (int i = 0; i < cpu->programa.Tam; i++)
          printf("\t%s", cpu->programa.instrucoes[i]);
      printf("\n");
      printf("Contador de Programa Atual: %d\n", cpu->contadorProgramaAtual);
      printf("Quantidade de variaveis: %d\n",cpu->Quant_Inteiros);
      if(cpu->Alocado_V_inteiros)
        for(int i=0;i<cpu->Quant_Inteiros;i++){
          printf("Valor inteiro da variavel ( %d ): %d\n", i,cpu->valorInteiro[i]);
        }
      printf("Fatia de Tempo Disponivel: %d\n", cpu->fatiaTempo);
      printf("Fatia de Tempo Usada: %d\n", cpu->fatiaTempoUsada);
      printf("\n\t--Fim das informacoes na CPU--\n");
}
void ImprimePronto(EstadoPronto *estadopronto){
  int Aux;
  printf("\n\tFila de Processsos Prontos:\n");
  for (Aux = estadopronto->Frente; Aux <= (estadopronto->Tras - 1); Aux++){
      printf("----------------- Processo %d -----------------\n",Aux );
      printf("PID: %i\n", estadopronto->vetor[Aux].pid);
      printf("PID Pai: %i\n", estadopronto->vetor[Aux].pid_do_pai);
      printf("Estado: %s\n", estadopronto->vetor[Aux].estado);
      printf("Tempo CPU: %d\n", estadopronto->vetor[Aux].CotaCPU);
      printf("Tempo Inicio: %d\n", estadopronto->vetor[Aux].startupTime);
      printf("Prioridade: %d\n", estadopronto->vetor[Aux].prioridade);
      printf("Quantidade de variaveis: %d\n",estadopronto->vetor[Aux].Estado_Processo.Quant_Inteiros);
      if(estadopronto->vetor[Aux].Estado_Processo.Alocado_V_inteiros)
        for(int i=0;i<estadopronto->vetor[Aux].Estado_Processo.Quant_Inteiros;i++){
            printf("Valor inteiro da variavel ( %d ): %d\n", i,estadopronto->vetor[Aux].Estado_Processo.Inteiro[i]);
        }
      printf("Contador de Programa: %d\n", estadopronto->vetor[Aux].Estado_Processo.Cont);
      printf("Programa do Processo: \n");
      for (int i = 0; i < estadopronto->vetor[Aux].Estado_Processo.Tam; i++)
          printf("\t%s", estadopronto->vetor[Aux].Estado_Processo.Programa[i]);
      printf("-----------------------------------------------\n");
  }
  printf("\n\t--Fim da fila de Processos Prontos--\n");
}
```

3. Testes de execução

Ao inicio do programa como foi pedido o usuário poderá escolher qual tipo de entrada ele deseja.



Abaixo esta uma simulação com determinados comandos e instruções:


String enviada pelo Controle(PID 48719) para o Gerenciador: U U U U U U U U U U U L U U U U U U U U I M
String recebida pelo Gerenciador de PID 48720 enviada pelo Controle: 'U U U U U U U U U U U L U U U U U U U U I M
Processo de PID 9347 adicionado a FILA PRONTO!
Execucao de instrucao -> N 3
Execucao de instrucao -> D 0
Execucao de instrucao -> D 1
Execucao de instrucao -> D 2
Execucao de instrucao -> F 7
Processo de PID 2862 adicionado a FILA PRONTO!
Execucao de instrucao -> B
Processo de PID 9347 adicionado a FILA BLOQUEADO!
Execucao de instrucao -> S 0 11
Execucao de instrucao -> V 1 30
Execucao de instrucao -> R Programa2.txt
->Nome arquivo novo: ./Arquivos_Entrada/Programa2.txt
->Arquivo aberto com sucesso!
Execucao de instrucao -> N 3
Execucao de instrucao -> D 1
Processo de PID 9347 adicionado a FILA PRONTO!
Execucao de instrucao -> D 2
Execucao de instrucao -> V 1 544
Execucao de instrucao -> F 0
Processo de PID 9539 adicionado a FILA PRONTO!

```
        Lista de Processos na Tabela:
----------------- Processo 0 -----------------
PID: 9539
PID Pai: 2862
Estado: PRONTO
Tempo CPU: 2
Tempo Inicio: 13
Prioridade: 0
Quantidade de variaveis: 3
Valor inteiro da variavel ( 0 ): 100
Valor inteiro da variavel ( 1 ): 0
Valor inteiro da variavel ( 2 ): 0
Contador de Programa: 2
Programa do Processo:
        N 3
        D 1
        D 2
        V 1 544
        F 0
        D 0
        V 0 100
        A 1 10
        S 4 20
        V 3 52
        A 2 15
-----------------------------------------------
----------------- Processo 1 -----------------
PID: 2862
PID Pai: 9347
Estado: PRONTO
Tempo CPU: 0
Tempo Inicio: 4
Prioridade: 0
Quantidade de variaveis: 3
Valor inteiro da variavel ( 0 ): 100
Valor inteiro da variavel ( 1 ): 0
Valor inteiro da variavel ( 2 ): 0
Contador de Programa: 7
Programa do Processo:
        N 3
        D 0
        D 1
        D 2
        F 7
        B
        A 0 10
        S 0 11
        V 1 30
        R Programa2.txt
        T
-----------------------------------------------
----------------- Processo 2 -----------------
PID: 9539
PID Pai: 2862
Estado: PRONTO
Tempo CPU: 0
Tempo Inicio: 13
```

```
--Inicio impresssao do estado do sistema---


        Informacoes da CPU:

Programa na CPU:
        N 3
        D 1
        D 2
        V 1 544
        F 0
        D 0
        V 0 100
        A 1 10
        S 4 20
        V 3 52
        A 2 15

Contador de Programa Atual: 2
Quantidade de variaveis: 3
Valor inteiro da variavel ( 0 ): 100
Valor inteiro da variavel ( 1 ): 0
Valor inteiro da variavel ( 2 ): 0
Fatia de Tempo Disponivel: 10
Fatia de Tempo Usada: 2

        --Fim das informacoes na CPU--



        Fila de Processsos Prontos:

        --Fim da fila de Processos Prontos--



        Fila de Processsos Bloqueados:
----------------- Processo 1 -----------------
PID: 2862
PID Pai: 9347
Estado: BLOQUEADO
Tempo CPU: 10
Tempo Inicio: 4
Prioridade: 1
Quantidade de variaveis: 3
Valor inteiro da variavel ( 0 ): 100
Valor inteiro da variavel ( 1 ): 0
Valor inteiro da variavel ( 2 ): 0
Contador de Programa: 7
Programa do Processo:
        N 3
        D 1
        D 2
        V 1 544
        F 0
        D 0
```

```
        Lista de Processos na Tabela:
----------------- Processo 0 -----------------
PID: 9539
PID Pai: 2862
Estado: PRONTO
Tempo CPU: 2
Tempo Inicio: 13
Prioridade: 0
Quantidade de variaveis: 3
Valor inteiro da variavel ( 0 ): 100
Valor inteiro da variavel ( 1 ): 0
Valor inteiro da variavel ( 2 ): 0
Contador de Programa: 2
Programa do Processo:
        N 3
        D 1
        D 2
        V 1 544
        F 0
        D 0
        V 0 100
        A 1 10
        S 4 20
        V 3 52
        A 2 15
-----------------------------------------------
----------------- Processo 1 -----------------
PID: 2862
PID Pai: 9347
Estado: PRONTO
Tempo CPU: 0
Tempo Inicio: 4
Prioridade: 0
Quantidade de variaveis: 3
Valor inteiro da variavel ( 0 ): 100
Valor inteiro da variavel ( 1 ): 0
Valor inteiro da variavel ( 2 ): 0
Contador de Programa: 7
Programa do Processo:
        N 3
        D 0
        D 1
        D 2
        F 7
        B
        A 0 10
        S 0 11
        V 1 30
        R Programa2.txt
        T
-----------------------------------------------
----------------- Processo 2 -----------------
PID: 9539
PID Pai: 2862
Estado: PRONTO
Tempo CPU: 0
Tempo Inicio: 13
```

Exemplo de entrada por Teclado:

4. Referência Bibliográfica

TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010. 653 p.