UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

ANIMAÇÃO DO FUNCIONAMENTO DE UM NÚCLEO DE SISTEMA OPERACIONAL

MARCO ANTONIO RUTHES DOS SANTOS

BLUMENAU
2005

2005/1-36

MARCO ANTONIO RUTHES DOS SANTOS

ANIMAÇÃO DO FUNCIONAMENTO DE UM NÚCLEO DE SISTEMA OPERACIONAL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Mauro Marcelo Mattos, - Orientador

BLUMENAU
2005

2005/1-36

ANIMAÇÃO DO FUNCIONAMENTO DE UM NÚCLEO DE SISTEMA OPERACIONAL

Por

MARCO ANTONIO RUTHES DOS SANTOS

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Mauro Marcelo Mattos – Orientador, FURB
Membro:	Prof. Antônio Carlos Tavares – FURB
Membro:	Prof. Miguel Alexadre Wisintainer – FURB

Blumenau, 23 de Junho de 2005

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

À minha família, especialmente a minha mãe.

Ao meu professor orientador Mauro Marcelo Mattos pelo auxilio e compreensão.

Quem não luta por sua liberdade, não a merece.

Herbert de Souza

RESUMO

O ensino dos conceitos de sistemas operacionais é uma tarefa difícil. O mérito deste trabalho reside no fato de que se pretende construir um ambiente de animação onde seja possível demonstrar a dinâmica do movimento dos componentes do núcleo de um sistema operacional. A hipótese de pesquisa do trabalho reside no fato de que se espera que o aluno ao visualizar os componentes em movimento possa construir os conceitos de uma forma mais intuitiva do que simplesmente os decorando. A grande vantagem deste projeto de software educacional é permitir a criação de um ambiente híbrido de ensino/aprendizado, permitindo a experimentação das teorias apresentadas em sala de aula, no qual o aluno desempenha um papel ativo e essencial. Além disso, o ambiente de animação pode contribuir para que o aluno depure a sua versão do simulador de processos concorrentes a partir da visualização do comportamento anômalo da animação.

Palavras-chave: *PowerPoint. Automation.* Animação. Sistemas operacionais.

ABSTRACT

The teaching of the operating systems concepts is a difficult task. The merit of this work resides in the fact that it is intended to build an animation environment, where can be possible to demonstrate the movement dynamic of the nucleus components of an operating system. The hypothesis of the work research resides in the fact that it is hoped that the student when he visualize the components in movement, he can build the concepts in a more intuitive form than memorizing them only. The great advantage of this educational software project is to allow the creation of a teaching/learning hybrid environment, allowing the theories experimentation introduced in the class room, in which the student plays an active and essential role. Besides that, the animation environment can contribute to the student improves his simulator version of the competitive processes, from the visualization of the animation anomalous behavior.

Key-Words: PowerPoint. Automation. Animation. Operating systems.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A disciplina de Sistemas Operacionais é fundamental nos currículos de graduação em Ciência da Computação, ao lado de outras disciplinas, como Arquitetura de Computadores e Redes. O correto entendimento dos mecanismos presentes nos sistemas operacionais permite ao profissional de informática uma melhor compreensão de seu ambiente de trabalho, resultando no desenvolvimento de soluções com maior qualidade e eficiência.

Morselli Junior (2003) afirma que a disciplina de Sistemas Operacionais faz parte de um conjunto de disciplinas que são tradicionalmente oferecidas pelos cursos de Ciência da Computação. Pode-se obter uma ótima discussão sobre o conteúdo programático de Sistemas Operacionais em Anido (2000).

Nesse contexto, a disciplina de Sistemas Operacionais busca apresentar aos alunos os conceitos e mecanismos fundamentais usados na construção de sistemas operacionais e como esses conceitos podem ser usados – e influenciar – na construção de aplicações. Além dos tópicos clássicos da área, como gerência de processos, memória e arquivos, a ementa da disciplina pode ainda cobrir assuntos específicos como escalonamento de tempo real, coordenação distribuída, sistemas embarcados, máquinas virtuais, etc (MAZIERO, 2002).

Para Machado e Maia (2004), a dificuldade no ensino de Sistemas Operacionais já vem sendo discutida há algum tempo por pesquisadores como Downey (1999) e Jones e Newman (2002). No Brasil, existem poucos trabalhos acadêmicos publicados sobre o ensino-aprendizado de Sistemas Operacionais. Dentre os existentes pode-se destacar Anido (2000), Machado e Maia (2004) e Maziero (2002).

Como exposto em Perez-Davilla (1995), o sistema operacional constitui uma “ponte” entre o mundo abstrato das teorias e algoritmos e o mundo prático e concreto do *hardware*. Essa característica de ponte entre dois mundos tão distintos pode tornar o ensino de Sistemas

Operacionais uma tarefa trabalhosa e pouco efetiva. Notadamente, exige-se dos alunos uma grande capacidade de construir abstrações mentais dos mecanismos envolvidos para poder compreender os aspectos mais densos da disciplina. Com isso, observa-se que muitos alunos concluem a disciplina conhecendo os principais conceitos e algoritmos, mas têm muita dificuldade em integrar todos aqueles conceitos de forma coerente.

Conforme Morselli Junior (2003), existem publicações, tais como Maziero (2002), que abordam com extrema clareza as principais características das ferramentas utilizadas no processo de aprendizagem de uma disciplina de Laboratório de Sistemas Operacionais. Entretanto, com relação ao conteúdo puramente teórico, oferecido pela disciplina Sistemas Operacionais pouco se discute em relação às formas de utilização das técnicas de ensino.

Uma das estratégias para apoiar o ensino de conceitos na disciplina é o uso de simuladores. Segundo Machado e Maia (2004), simuladores envolvem a criação de modelos dinâmicos e simplificados do mundo real, sendo o potencial educacional deste tipo de ferramenta muito superior ao dos programas tradicionais.

Na área da ciência da computação existem simuladores que auxiliam no ensino de várias disciplinas, como Redes de Computadores, Técnicas de Programação, Arquitetura de Computadores e Sistemas Operacionais. Dentre os diferentes tipos de simuladores, cada um possui suas características próprias, vantagens e desvantagens. Em geral, os simuladores, apesar de mais simples que os sistemas reais, também oferecem dificuldades em utilizá-los. A maioria dos simuladores em Sistemas Operacionais apresenta uma elevada curva de aprendizado, exigindo também bons conhecimentos de Unix e programação (MACHADO; MAIA, 2004)

O professor Mauro Mattos, na disciplina de Sistemas Operacionais da FURB, desenvolve em sala de aula uma série de exercícios de fixação de alguns conceitos importantes da mesma. Um dos exercícios realizados pelos alunos consiste em representar

numa planilha os eventos que ocorrem durante a execução de um conjunto de processos em um núcleo de sistema operacional preemptivo. O resultado deste exercício é um *log* gerado em arquivo texto conforme apresentado na figura 1. Nesta figura é possível identificar que processo está executando. Também é possível destacar em que momento quais processos estavam prontos para execução ou bloqueados aguardando por algum evento. Maiores detalhes serão apresentados no capitulo 2.

Main:	E21	-	-	E22	-	-	E23	-	P	P	P	P	P	P	P
Leitor 1:	-	-	-	P	P	P	P	P	P	E 1	E 2	-	E 3	E 4	E 5
Leitor 2:	-	-	-	-	-	-	P	P	P	P	P	P	P	P	P
Escritor 1:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Leitor 3:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Escritor 2:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Escritor 3:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Cria P.:	-	E	E	-	E	E	-	E	-	-	-	-	-	-	-
P:	-	-	-	-	-	-	-	-	-	-	-	E	-	-	-
V:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Interrupção:	-	-	-	-	-	-	-	-	E	-	-	-	-	-	-
Idle:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Debug:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

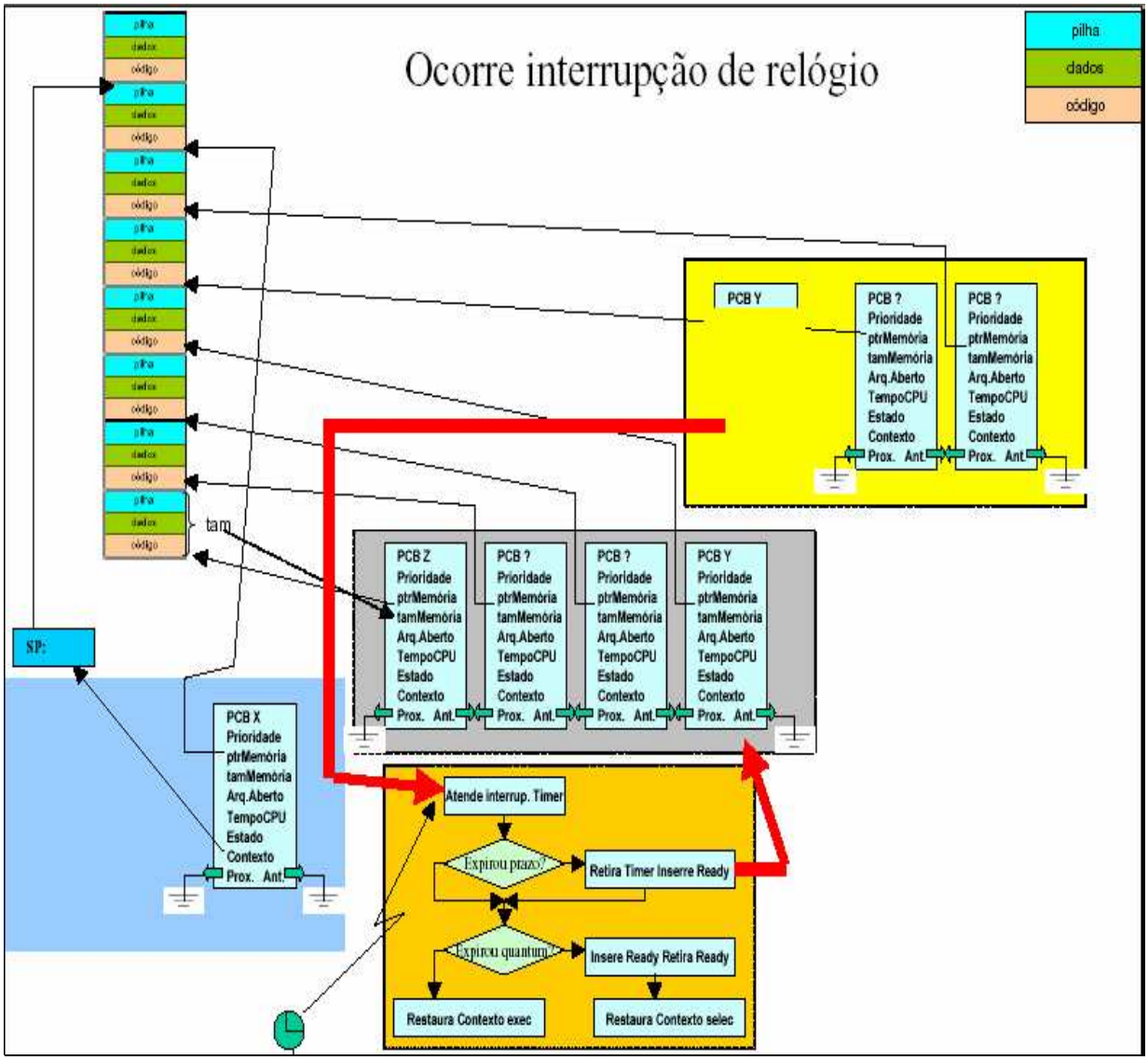
Main:	P	P	P	P	P	E25	-	-	E26	P	P	P	P	P	P
Leitor 1:	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
Leitor 2:	P	P	P	P	P	P	P	P	P	P	E 3	E 4	E 6	-	E 7
Escritor 1:	-	B	B	B	B	B	B	B	B	B	B	B	B	B	B
Leitor 3:	P	P	E 1	E 2	-	B	B	B	B	B	B	B	B	B	P
Escritor 2:	-	-	-	-	-	-	-	-	P	P	P	P	P	P	P
Escritor 3:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Cria P.:	-	-	-	-	-	-	E	E	-	-	-	-	-	-	-
P:	E	-	-	-	E	-	-	-	-	-	-	-	-	-	-
V:	-	-	-	-	-	-	-	-	-	-	-	-	-	E	-
Interrupção:	-	E	-	-	-	-	-	-	-	E	-	-	-	-	-
Idle:	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Debug:	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46

Fonte : Mattos (2004)

Figura 1 - Exemplo de *log* de execução de uma simulação produzida pelos alunos.

Tendo em vista permitir uma melhor visualização do comportamento do sistema simulado, Mattos (2004) desenvolveu uma série de *slides* em *PowerPoint*, para caracterizar o comportamento simulado de um núcleo de sistema operacional. A figura 2 apresenta um destes *slides*.

Embora contribua para o entendimento do funcionamento de um núcleo de sistema operacional, este conjunto de *slides* estático não é suficiente para adaptar-se aos exemplos gerados pelas simulações produzidas pelos alunos. Neste sentido, o que se propõe neste trabalho é a construção de uma ferramenta que possibilite a animação dos *logs* produzidos pelos alunos a partir dos estudos de caso propostos em aula.



Fonte: Mattos (2004)

Figura 2 - estado instantâneo do funcionamento do núcleo

1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho proposto é desenvolver um ambiente de animação de objetos gráficos para representar o funcionamento dos principais componentes de um núcleo de sistema operacional a partir das simulações produzidas pelos alunos.

Os objetivos específicos do trabalho são:

a) construir uma biblioteca que, utilizando-se do *PowerPoint Object Model* do pacote *Office Automation* da *Microsoft*, permita a configuração das características do ambiente de animação;

b) construir uma interface que permita o controle dos detalhes da animação pelo aluno através de um programa externo ao *PowerPoint*;

c) especificar uma interface padrão para aquisição dos eventos de animação produzidos pelas ferramentas de simulação desenvolvidas pelos alunos.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo deste trabalho tem-se a introdução, apresentação dos objetivos e a organização do mesmo.

No segundo capítulo é iniciada a fundamentação teórica com uma introdução a *PowerPoint automation*, uma breve descrição sobre Sistemas Operacional.

O terceiro capítulo é dedicado a processos consumidores no núcleo do Sistema Operacional. Nele são descritas as principais características e comportamentos breve descrição para cada um dos eventos.

O quarto capítulo mostra o desenvolvimento do trabalho. Aqui encontra-se a explicação dos principais requisitos e a sua especificação. São abordados os temas como implementação, técnicas e ferramentas utilizadas e operacionalidades da implementação. No final do terceiro capítulo é apresentada a operacionalidade da ferramenta desenvolvida.

O quinto capítulo foi destinado a simulação dos eventos do núcleo do Sistema Operacional. Nele são apresentadas explicações dos testes realizados assim como as telas capturadas no momento dos testes.

No sexto capítulo temos as conclusões finais e extensões para novos trabalhos.

2 MULTIPROGRAMAÇÃO E PROGRAMAÇÃO CONCORRENTE

O proposto do presente capítulo é apresentar a fundamentação dos conceitos relacionados ao ambiente de animação construído.

2.1 SISTEMA OPERACIONAL

De acordo com Oliveira (2001), o sistema operacional é uma camada de software colocada entre o hardware e os programas que executam tarefas para os usuários. Ele é responsável pelo acesso aos periféricos. Sempre que um programa necessita de algum tipo de operação de entrada e saída, ele a solicita ao sistema operacional. Desta forma, o programa não precisa conhecer os detalhes do *hardware*. Informações do tipo “como enviar um caractere para a impressora” ficam escondidas dentro do sistema operacional. Ao mesmo tempo, como todos os acessos aos periféricos são feitos através do sistema operacional, ele pode controlar qual programa está acessando qual recurso. É possível, então, obter uma distribuição justa e eficiente dos recursos.

Para Tanenbaum (2002), a interface entre sistema operacional e os programas de usuários é definida pelo conjunto de “instruções estendidas” que o sistema operacional proporciona.

2.2 MULTIPROGRAMAÇÃO

Em um sistema multiprogramado diversos recursos são mantidos na memória ao mesmo tempo. Neste sistema, supondo que existem 3 programas de usuário na memória principal, prontos para serem executados. Supondo que o sistema operacional inicia a

execução do programa 1. Após algum tempo, da ordem de milisegundos, o programa 1 faz uma chamada ao sistema. Ele solicita algum tipo de operação de entrada ou saída. Por exemplo, uma leitura do disco. Sem multiprogramação, o processo ficaria parado durante a realização do acesso. Em um sistema multiprogramado, enquanto o periférico executa o comando enviado, o sistema operacional inicia a execução de outro programa (OLIVEIRA, 2001).

2.2.1 O CONCEITO DE PROCESSO

Conforme Tanenbaum (2002, p. 26), um processo é basicamente um programa em execução, Associado com cada processo está seu espaço de endereçamento.

Segundo Oliveira (2001, p. 14), não existe uma definição objetiva, aceita por todos, para a idéia de processo. Na maioria das vezes, um processo é definido como “um programa em execução”. O conceito de processo é bastante abstrato, mais essencial no estudo de sistemas operacionais. O processo é um elemento ativo. Altera seu estado, à medida que executa um programa. É o processo que faz chamadas de sistema, ao executar os programas.

2.2.2 CICLOS DE UM PROCESSO

Processos são criados e destruídos, o momento e a forma pela qual os processos são criados e destruídos depende do sistema operacional em consideração. Alguns sistemas trabalham com um número fixo de processos. Por exemplo, um processo para cada terminal do computador. Neste caso, todos os processos são criados na inicialização do sistema. Eles somente são destruídos quando o próprio sistema é desligado

A forma mais flexível de operação é permitir que processos possam ser criados

livremente, através de chamadas de sistema. Alem da chamada de sistema “cria processo”, serão necessárias chamadas para “autodestruição do processo” e também para “eliminação de outro processo”, (OLIVEIRA, 2001).

2.2.3 ESTADOS DE UM PROCESSO

Tanenbaum (2002, p. 48), diz que embora cada processo seja uma entidade independente, com seu próprio contador de programas e estado interno, os processos freqüentemente precisam interagir entre si. Um processo pode gerar alguma saída que outro processo utiliza como entrada

Segundo Oliveira (2001, p. 16), após ser criado, o processo entra em um ciclo de processador. Ele precisa de processador para executar. Entretanto, o processador poderá estar ocupado com outro processo, e ele deverá esperar. Diversos processos podem estar neste mesmo estado.

Em máquinas multiprocessadoras existem diversos processadores. Neste caso, diversos processadores executam ao mesmo tempo. Porém, essa não é a situação mais comum. Vamos supor que exista um único processador no computador. Neste caso, é necessário manter uma fila com os processos aptos a ganhar o processador. Essa fila é chamada “fila de aptos”.

2.2.4 GERÊNCIA DE FILAS

Para Oliveira (2001, p. 18), quando uma solicitação é feita é preciso verificar a fila de periféricos. Se a fila estiver vazia, o periférico está livre. O pedido é inserido na fila, e o comando é enviado ao controlador. Caso contrario o periférico está ocupado, O pedido é inserido na fila, mais nada é enviado ao controlador. O primeiro pedido da fila pode ser

removido, pois o acesso foi concluído. O processo correspondente volta para a fila de aptos, para disputar processador. Se após a sua remoção, a fila ficou vazia, então o periférico esta livre. Caso contrario, é necessário enviar para o controlador do periférico o primeiro pedido da fila. Trata-se de uma solicitação feita anteriormente, mais que não fora enviada pois o periférico estava ocupado.

2.2.5 MECANISMO DE INTERRUPÇÕES

Para Vieira (1975, p. 6), o mecanismo de interrupções é um recurso comum dos processadores de qualquer porte. Ele permite que um controlador de periférico chame a atenção do processador. Fisicamente a determinada ocorrência, o barramento de controle é usado para o envio de sinais elétricos associados com a geração de uma interrupção.

De acordo com Oliveira (2001, p. 20), uma interrupção sempre sinaliza a ocorrência de algum evento. Quando ela acontece, desvia a execução da posição atual de um programa para uma rotina específica. Essa rotina, responsável por atender a interrupção é chamada de tratador de interrupção. O tratador realiza as ações necessárias em função da ocorrência da interrupção. Ele é, simplesmente, uma rotina que somente é executada quando ocorre uma interrupção. Quando o tratador termina, a execução volta para a rotina interrompida, sem que essa perceba que foi interrompida.

Existem momentos em que um programa não pode ser interrompido. Por exemplo, o programa pode estar alterando variáveis que também são acessadas pelo tratador de interrupção. Durante a alteração, o programa pode deixar essas variáveis temporariamente com valores inconsistentes. Se a interrupção ocorrer nesse instante, o tratador será ativado e irá acessar essas variáveis, que estão com valores incorretos. É preciso ter em mente que o instante exato em que vai ocorrer uma interrupção de hardware é imprevisível.

A solução seria desligar temporariamente o mecanismo de interrupção. Os processadores normalmente possuem instruções para habilitar e desabilitar interrupções. Enquanto as interrupções estiverem desabilitadas, elas serão ignoradas pelo processador. Elas não serão perdidas, apenas ficam pendentes. Quando o programa tornar a habilitar as interrupções, elas imediatamente serão atendidas pelo processador. Com as interrupções desabilitadas, o acesso as estruturas de dados pode ser feita de forma mais segura, (OLIVEIRA, 2001).

2.3 PROGRAMAÇÃO CONCORRENTE

De acordo com Oliveira (2001, p. 29), um programa concorrente é executado simultaneamente por diversos processos que cooperam entre si, isto é, trocam informações. É necessária a interação entre processos para serem considerados concorrentes. Embora a interação entre processos possa ocorrer através de acesso a arquivos comuns.

É comum em sistema multi-usuário que um mesmo programa seja executado simultaneamente por vários usuários. Por exemplo, um editor de texto. Entretanto, executar simultaneamente 10 instâncias do editor de texto não faz dele um programa concorrente. Apenas o código é possivelmente compartilhado pelos 10 processos. Cada processo executa sobre sua própria área de dados e ignora a existência de outras execuções do programa. Esses processos não cooperam entre si, isto é, não trocam informações. Neste exemplo, temos a execução de 10 instâncias do mesmo programa seqüencial, e não um programa concorrente (OLIVEIRA, 2001).

2.3.1 PROBLEMAS DA SEÇÃO CRÍTICA

A quantidade exata de memória compartilhada entre os processos pode variar conforme o programa. Processos podem compartilhar em todo o seu espaço de endereçamento, apenas um segmento de memória, algumas estruturas de dados ou algumas variáveis. O sistema operacional arranja para que processos acessem as mesmas posições de memória.

Seção critica é aquela parte do código de um processo que acessa uma estrutura de dados compartilhados. Por exemplo, o código do Escritor que insere nomes de arquivos na fila e o código do leitor que retira esse nomes são seções criticas. O problema da seção critica está em garantir que, quando um processo está executando sua seção crítica, nenhum outro processo entre na sua respectiva seção critica. No exemplo, isso significa que, enquanto o processo Escritor estiver inserindo um nome na fila, o processo Leitor não poderá retirar nomes da fila, e vice-versa.

Uma solução para o problema da seção crítica estará correta quando apresentar as seguintes propriedades:

a) A solução não depende das velocidade relativas dos processos;
b) Quando um processo P deseja entrar na seção crítica e nenhum outro processo está executando a sua seção crítica, o processo P não é impedido de entrar;
c) Nenhum processo pode ter seu ingresso na seção crítica postergado onde indefinidamente, ou seja, fica esperando pra sempre;
d) Existe exclusividade mútua entre os processos com referência a execução das respectivas seções críticas.

2.3.2 SEMÁFOROS

Segundo Silberschatz (2000, p. 130), semáforo é um mecanismo de sincronização muito empregado. Somente duas operações são permitidas sobre o semáforo. Elas são conhecidas como P (do holandês *proberen*, testar) e V (do holandês *verhogen*, incrementar).

Para que semáforos funcionem corretamente, é essencial que as operações P e V sejam atômicas. Isto é, uma operação P ou V não pode ser interrompida no meio e outra operação sobre o mesmo semáforo iniciada.

Semáforos tornam a proteção da seção critica muito simples. Para cada estrutura de dados compartilhada, deve ser criado um semáforo S inicializado com o valor 1. Todo processo antes de acessas essa estrutura, deve executar um P(S), ou seja, a operação P sobre o semáforo S associado com a estrutura de dados em questão. Ao sair da seção crítica, o processo executa V(S) .

2.4 GERÊNCIA DO PROCESSADOR

Segundo Mattos (2004), a gerência de processador é uma atividade que envolve a representação das entidades envolvidas no funcionamento de um núcleo de sistema operacional bem como seus respectivos estados.

2.4.1 BLOCO DESCRITOR DE PROCESSO

Existem várias informações que o sistema operacional deve manter a respeito dos processos. No “programa” sistema operacional, um processo é representado por um registro. Esse registro é chamado de bloco descritor de processo ou simplesmente descritor de processo

(DP). No DP, fica tudo que o sistema operacional precisa saber sobre o processo. Abaixo alguns campos que normalmente são encontrados no descritor de processo:

a) Prioridade do processo no sistema, usada para definir a ordem na qual os processos recebem o processador;
b) Localização e tamanho da memória principal ocupada pelo processo;
c) Identificação dos arquivos abertos no momento;
d) Informação para contabilidade, como tempo de processador gasto, espaço de memória ocupada, etc;
e) Estado do processo: apto, executando, bloqueado;
f) Contexto de execução quando o processo perde o processador, ou seja, conteúdo dos registradores do processador quando o processo é suspenso temporariamente;
g) Apontadores para encadeamento dos blocos descritores de processo.

Na prática, descritores não são copiados. Todas as filas são implementadas como listas encadeadas. A passagem do descritor de uma fila para a outra é feita através da manipulação de apontadores (OLIVEIRA, 2001).

2.4.2 CHAVEAMENTO DE CONTEXTO

Em um sistema multiprogramado, é necessário interromper processos para continuá-los mais tarde. Essa tarefa é chamada de chaveamento de processo, ou chaveamento de contexto de execução.

O contexto de execução é formado basicamente pelos registradores do processador. O programa do usuário não sabe que será interrompido diversas vezes durante a sua execução. Logo, não é possível deixar para o programa a tarefa de salvar os registradores. Isso deve ser

feito pelo próprio sistema operacional.

Os conteúdos dos registradores são salvos toda vez que um processo perde o processador. Eles são realocados, quando o processo volta a executar. Desta forma o processo não percebe que foi interrompido. Em geral, salvar o contexto de execução do processo em execução é a primeira tarefa do sistema operacional, ao ser acionado. Da mesma forma a ultima tarefa do sistema operacional ao entregar o processador para um processo é repor o seu contexto de execução. O modulo do sistema operacional que realiza a reposição do contexto é chamado de *dispatcher* (OLIVEIRA, 2001).

2.4.3 ALGORITMOS DE ESCALONAMENTO

Na escolha de um algoritmo de escalonamento, utiliza-se como critério básico o objetivo de aumentar a produção do sistema e, ao mesmo tempo, diminuir o tempo de resposta recebida pelos usuários. Esses dois objetivos podem tornar-se conflitantes em determinadas situações.Para aumentar a produção do sistema, é necessário manter o processador ocupado todo o tempo. Dessa forma, o sistema produz mais em menos tempo. Também é importante oferecer um baixo tempo de resposta ao usuário. Isto é obtido, no caso da gerencia do processador, cm um baixo tempo médio de espera na fila do processador.

Não adianta um baixo tempo médio de resposta com variância elevada. Variância elevada significa que a maioria dos processos recebe um serviço satisfatório, enquanto alguns são bastante prejudicados. Provavelmente, será melhor sacrificar o tempo médio de resposta para homogeneizar a qualidade do serviço que os processos recebem.

2.4.3.1 ORDEM DE CHEGADA (FIFO - First-in first-out)

Esse é o algoritmo de implementação mais simples. A fila do processo é uma fila simples. Os processos são executados na mesma ordem em que chegaram na fila. Um processo somente libera o processador quando realiza uma chamada de sistema ou quando ocorre algum erro na execução. As chamadas de sistema incluem a própria indicação de término do processo.

O problema desse algoritmo é o desempenho. Quando um processo está na frente da fila, todos os processos devem esperar que ele termine seu ciclo de processador para então executar.

3 MODELO DE SIMULAÇÃO

Conforme citado anteriormente, o presente trabalho implementa um ambiente de animação do funcionamento de um núcleo de sistema operacional baseado na especificação original apresentada nas notas de aula do Prof. Mauro Mattos (MATTOS,2004).

Assim sendo, faz-se necessário contextualizar a ferramenta de simulação descrita em Mattos (2004).

O método de ensino dos conceitos básicos de um ambiente de multitarefa adotado pelo Prof. Mauro Mattos envolve a utilização de slides desenvolvidos em *PowerPoint* e planilhas desenvolvidas em *Excel*.

Os slides (figuras 3 a 8) desenvolvidos em *PowerPoint* são empregados para caracterizar a dinâmica de comportamento de um núcleo de sistema operacional multitarefa.

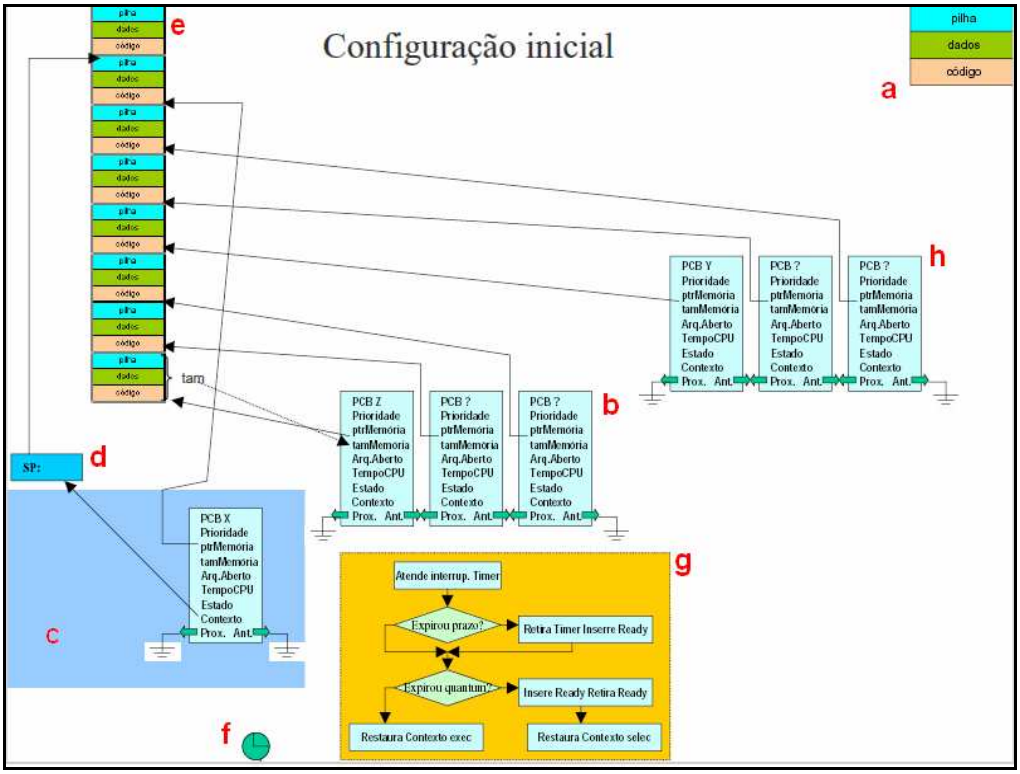


Figura 3 - Estado inicial de um núcleo de sistema operacional multitarefa

Na configuração inicial (figura 03) é possível identificar:

a) legenda indicando uma área de memória alocada para determinado processo onde se caracteriza que esta região de memória é sub dividida em: área de código, área de dados e área de pilha de processo;
b) fila de processos aptos a receber a CPU. É caracterizada pelo enfileiramento de blocos descritores de processos, os quais identificam, através de setas, qual região da mesma esta alocada para determinado processo;
c) bloco descritor do processo atualmente utilizando a CPU;
d) apontador para tipo de pilha do processo atualmente usado na CPU;
e) memória da máquina contendo as regiões demarcadas segundo a legenda *a*;
f) um ícone de um relógio representando o relógio de tempo real de sistema o qual gera sinais de interrupção de tempos em tempos;
g) fluxograma representando a seqüência lógica dos passos a serem executados pelo tratador de interrupção do relógio, ou seja, pelo código do núcleo do sistema operacional responsável para atendimento de interrupção do relógio;
h) fila de descritores de processos que estão bloqueados aguardando o término de alguma atividade – por exemplo : aguardando o término de uma operação de E/S..

A Figura 4 representa a seqüência lógica dos passos a serem executados quando ocorre uma interrupção do relógio e o tratador desta interrupção detecta que o prazo que um determinado processo desejava aguardar (através de uma chamada de Sistema Operacional - *delay*) esgotou-se. Com isso o código do tratador da interrupção retira o descritor do processo da fila de bloqueados e insere no final da fila de prontos.

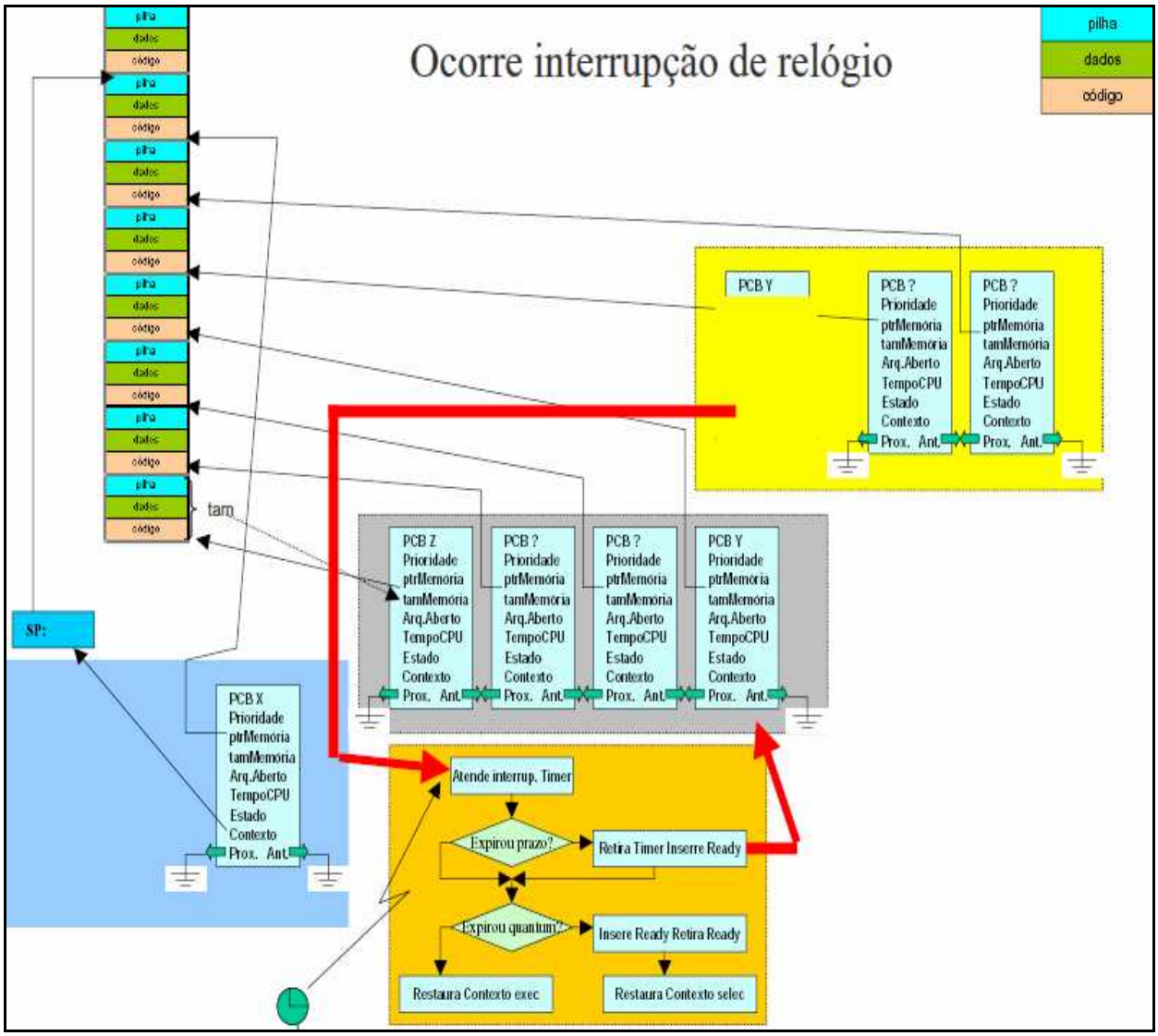


Figura 4 - Ocorrência da interrupção de relógio

A figura 5 caracteriza a situação onde, segundo o fluxograma, o tratador detecta que a fatia de tempo a que um processo tinha direito de executar esgotou-se.

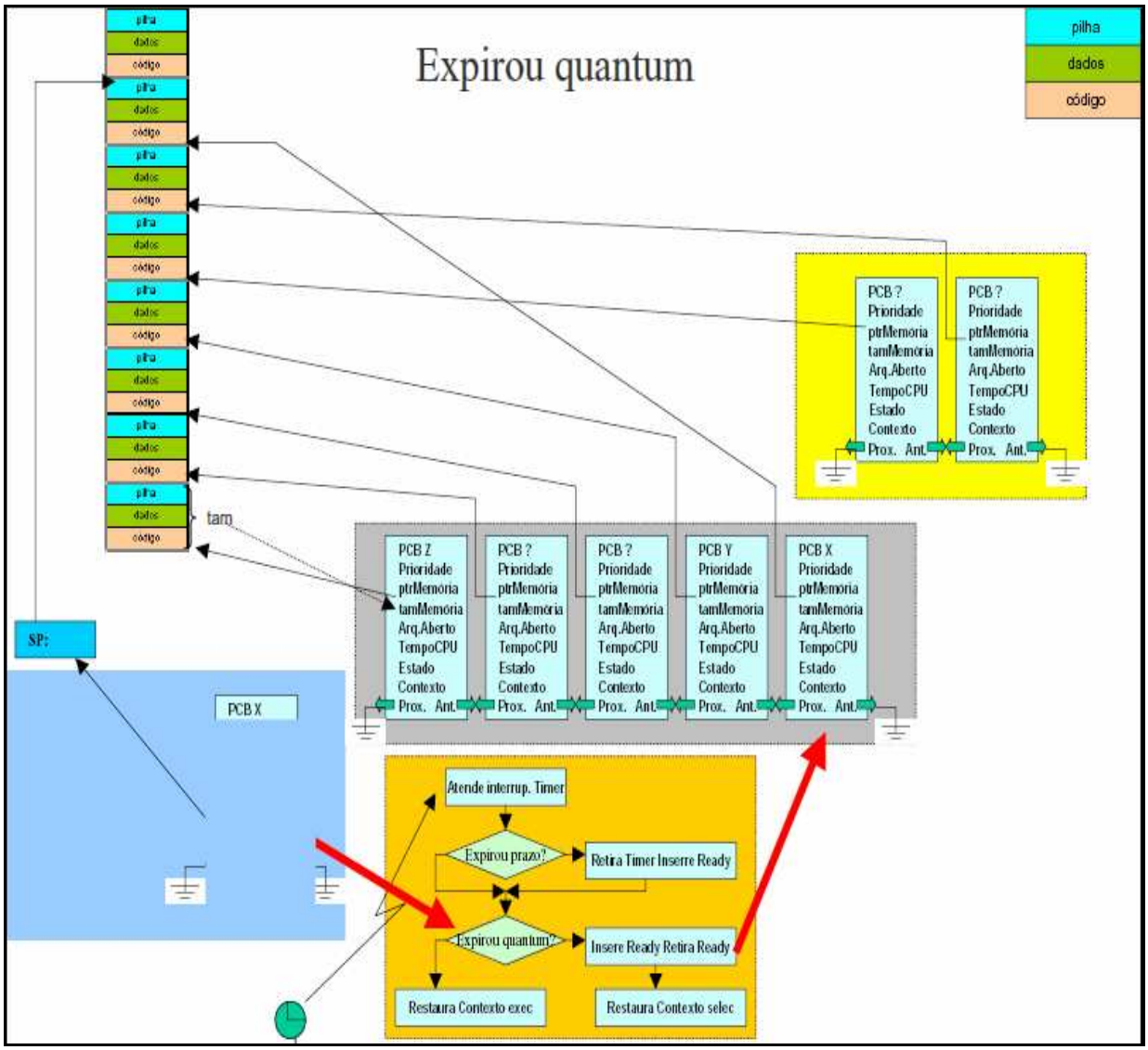


Figura 5 - Quantum expirou e assume nova posição

Isto faz com que, o tratador ative o escalonador de processos para deslocar o descritor do processo que estava executando quando a interrupção ocorreu e inseri-lo no final da fila de prontos.

A figura 6 caracteriza o término do processo de escalonamento quando o escalonador retira o descritor de fila de prontos e entrega a CPU a ele.

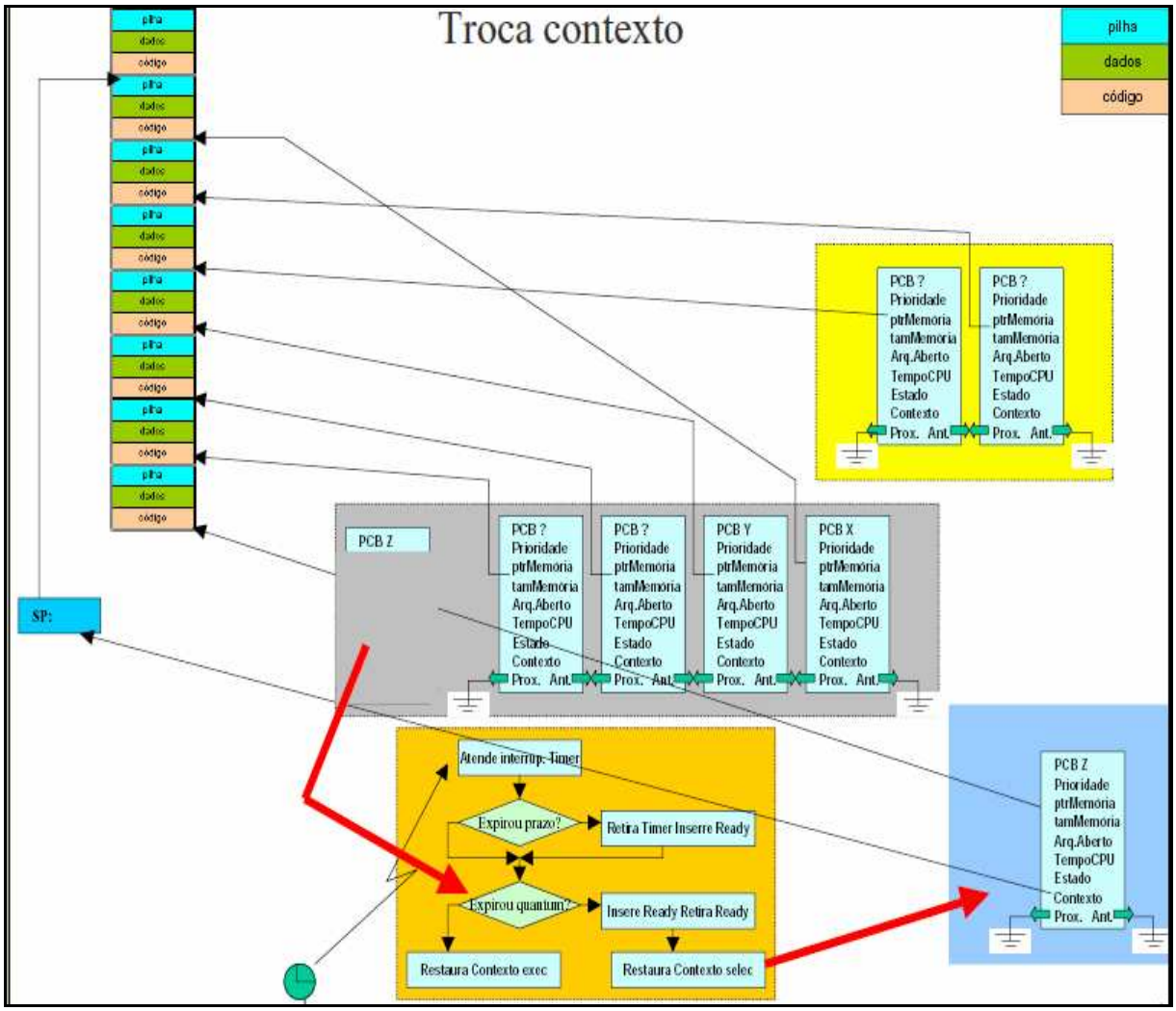


Figura 6 - Na troca de contexto um novo processo assume a CPU

A figura 7 caracteriza a situação onde a interrupção do relógio não provoca uma troca de contexto e após verificar se esgotou o prazo de espera de algum processo que realiza a chamada a primitiva *delay*, o tratador do relógio retorna o controle ao processo atual para que continue a sua execução.

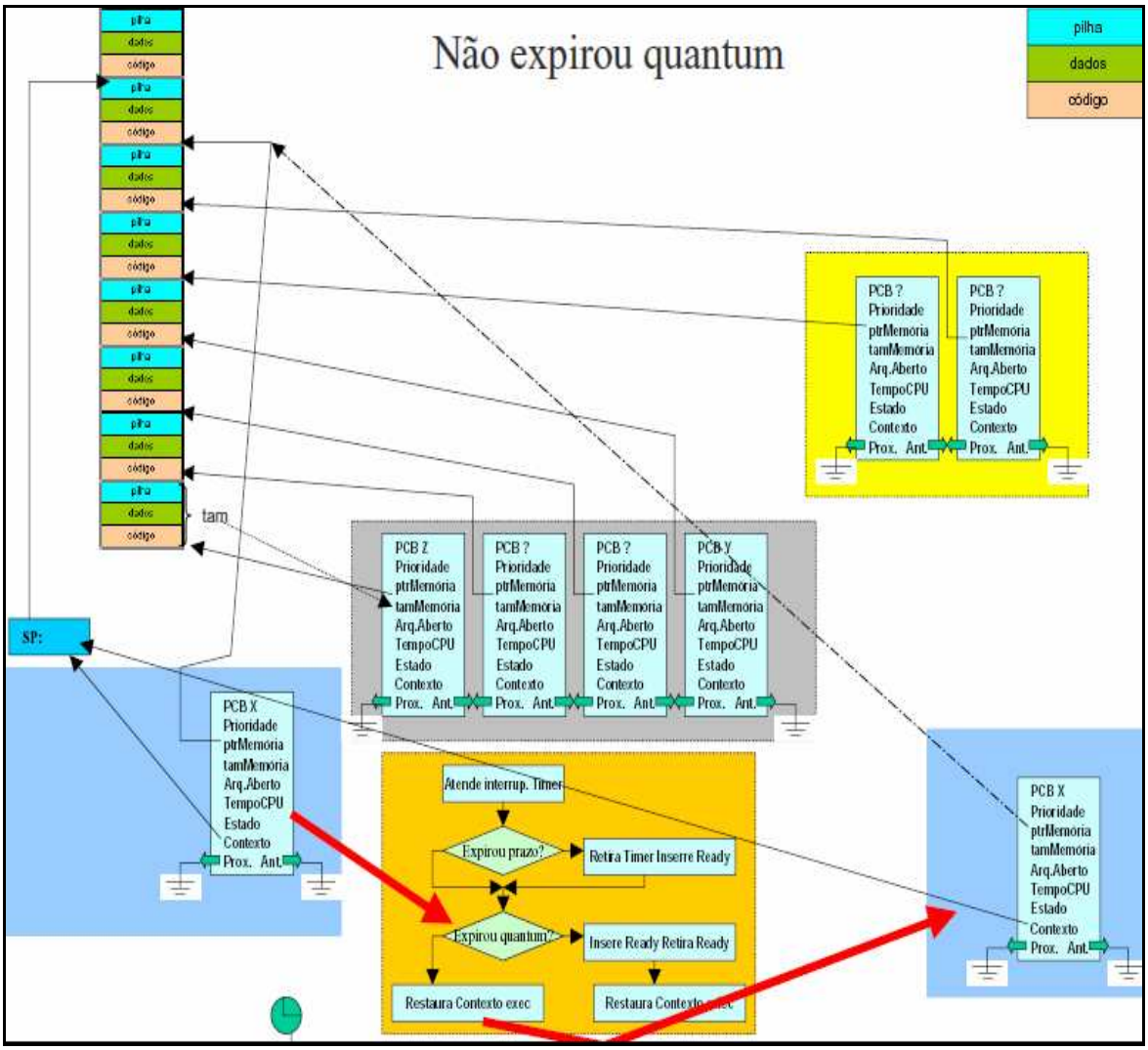


Figura 7 - Não expirou o quantum

A figura 8 caracteriza uma chamada de sistema, ou *system call*.

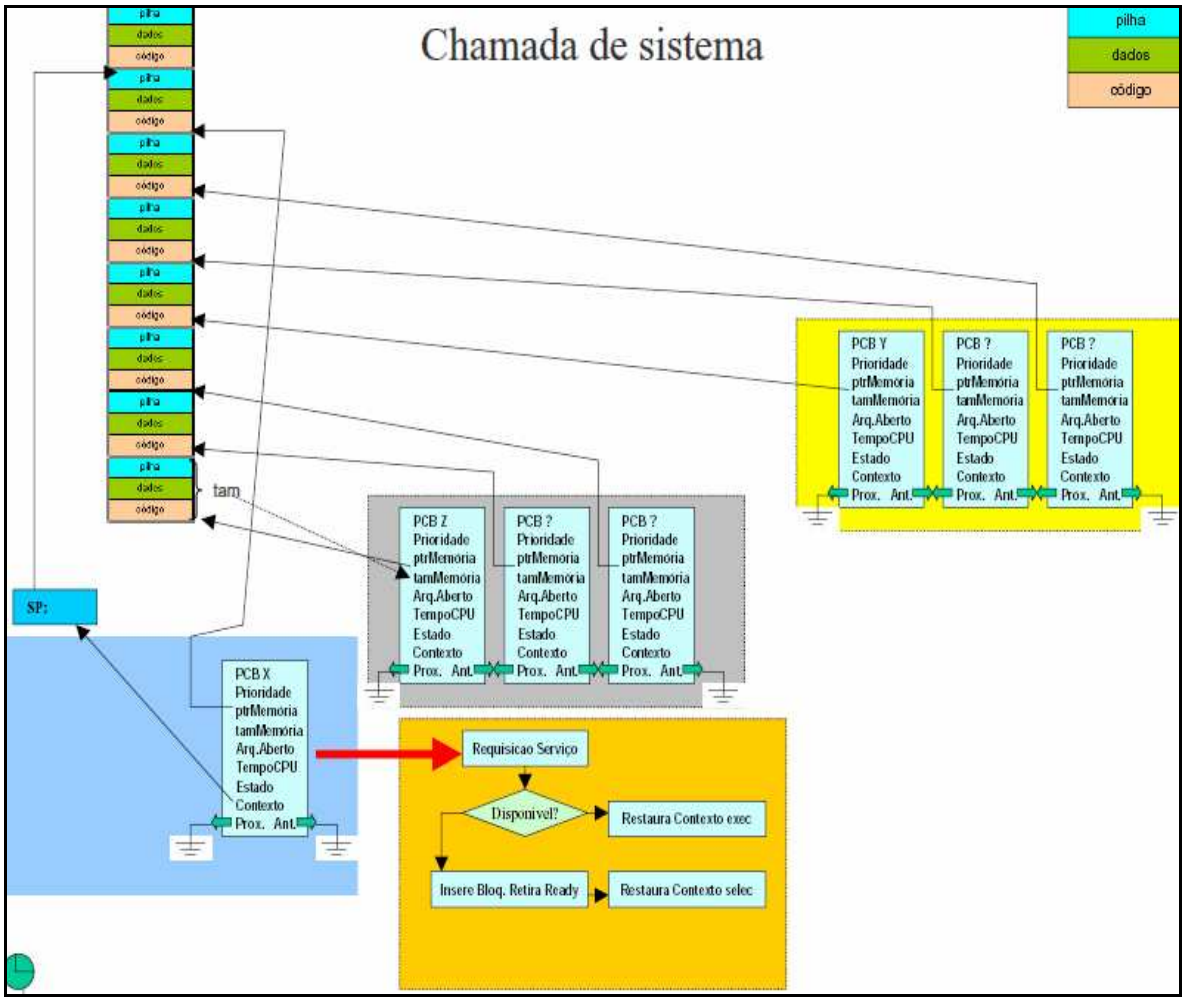


Figura 8 - Nova chamada do sistema

A figura 9 caracteriza uma chamada de sistema sem bloqueio, ou seja, uma chamada de sistema onde, o sistema executa uma sub-rotina interna e retorna imediatamente o resultado.

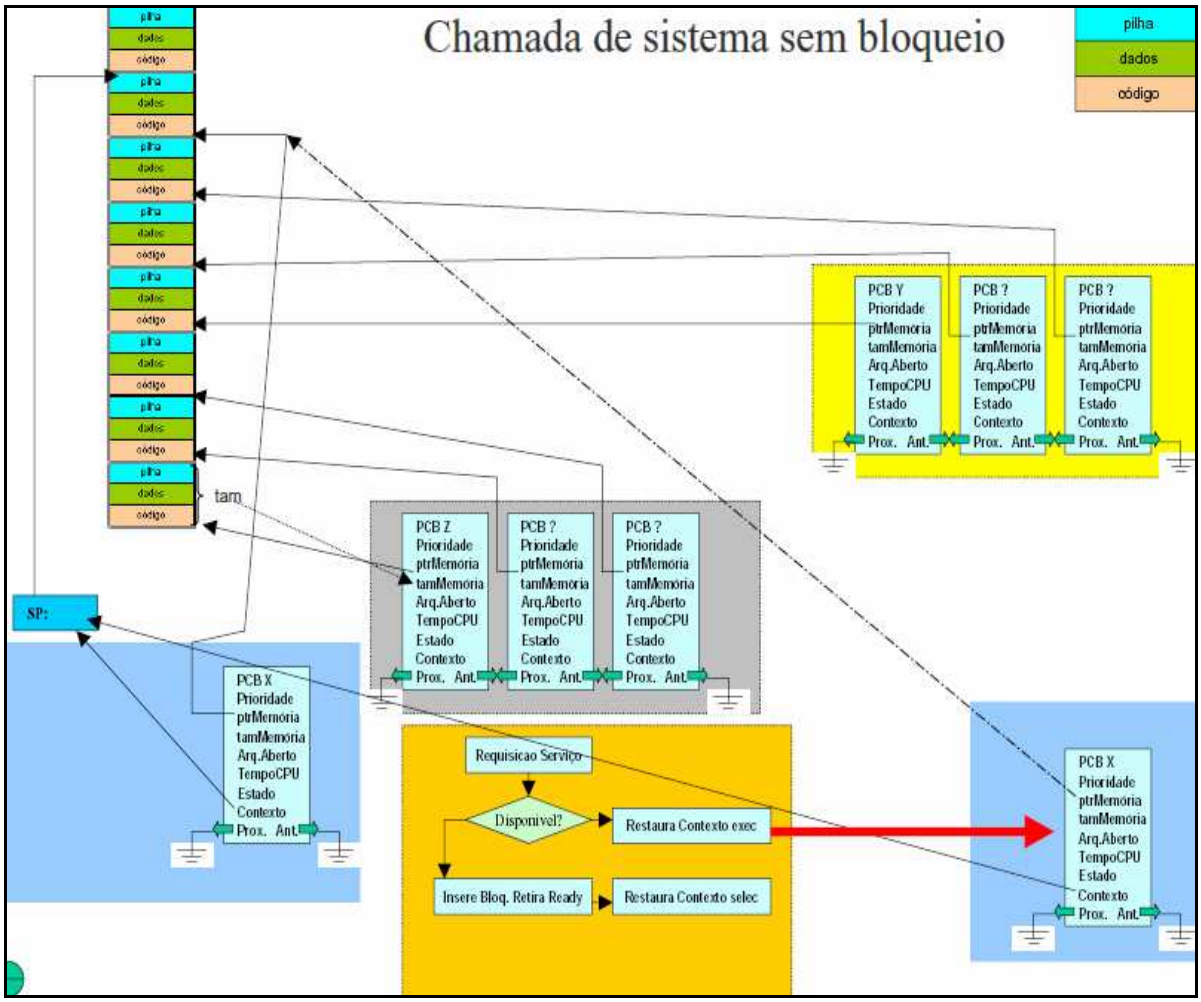


Figura 9 – Atendimento da requisição do sistema sem bloqueio

A figura 10 caracteriza uma situação onde o processo realizou uma *system call* e pode atender prontamente a requisição. Neste caso o descritor do processo é inserido na fila de bloqueados e a requisição (provavelmente um acesso a algum periférico) é disparada.

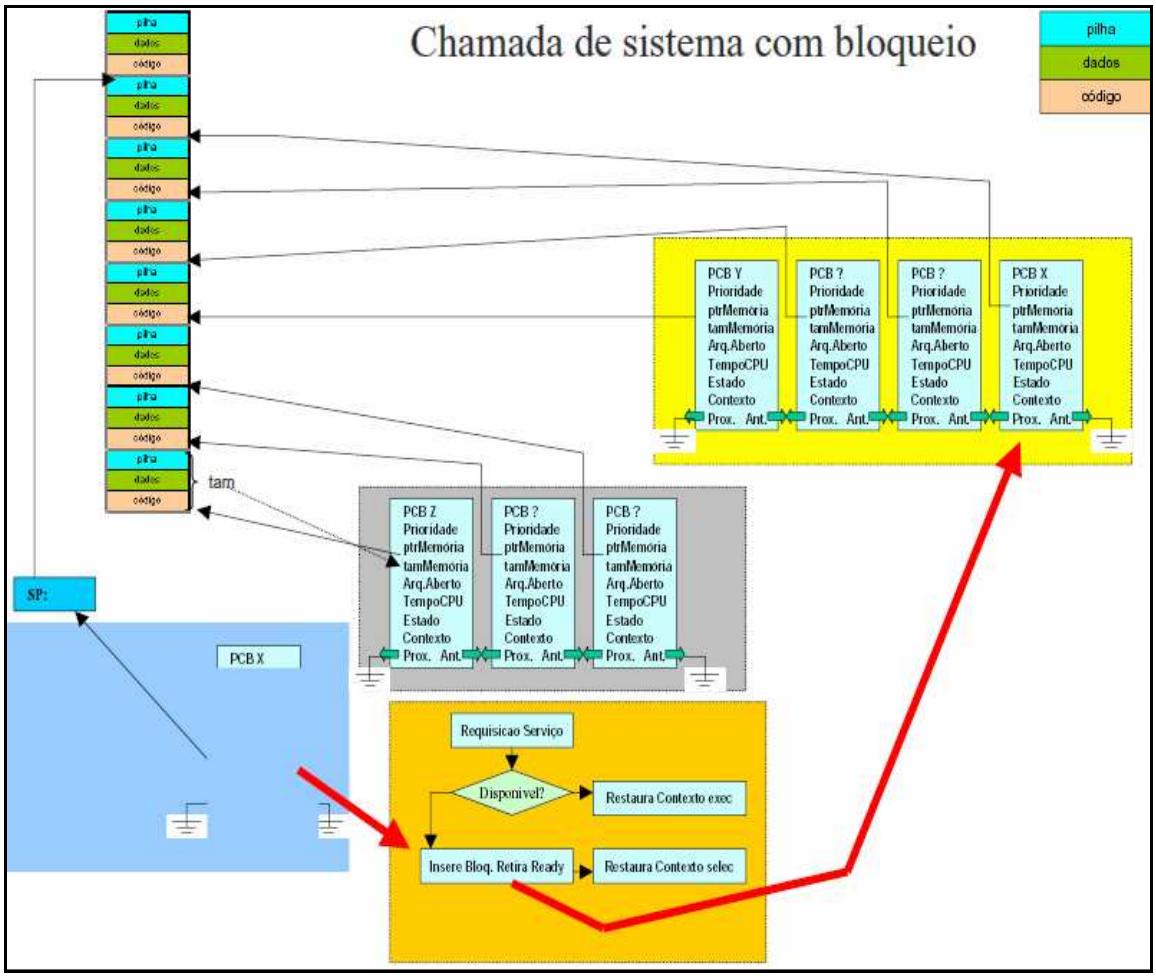


Figura 10 - Ocorrência de requisição do sistema com bloqueio

As Figuras 11 e 12 caracterizam o momento em que o hardware interrompe o processador indicando o termino da requisição de E/S. O tratador desta interrupção retira o descritor do processo da fila de bloqueados (Figura 11) e insere-o no final da fila de prontos(Figura 12).

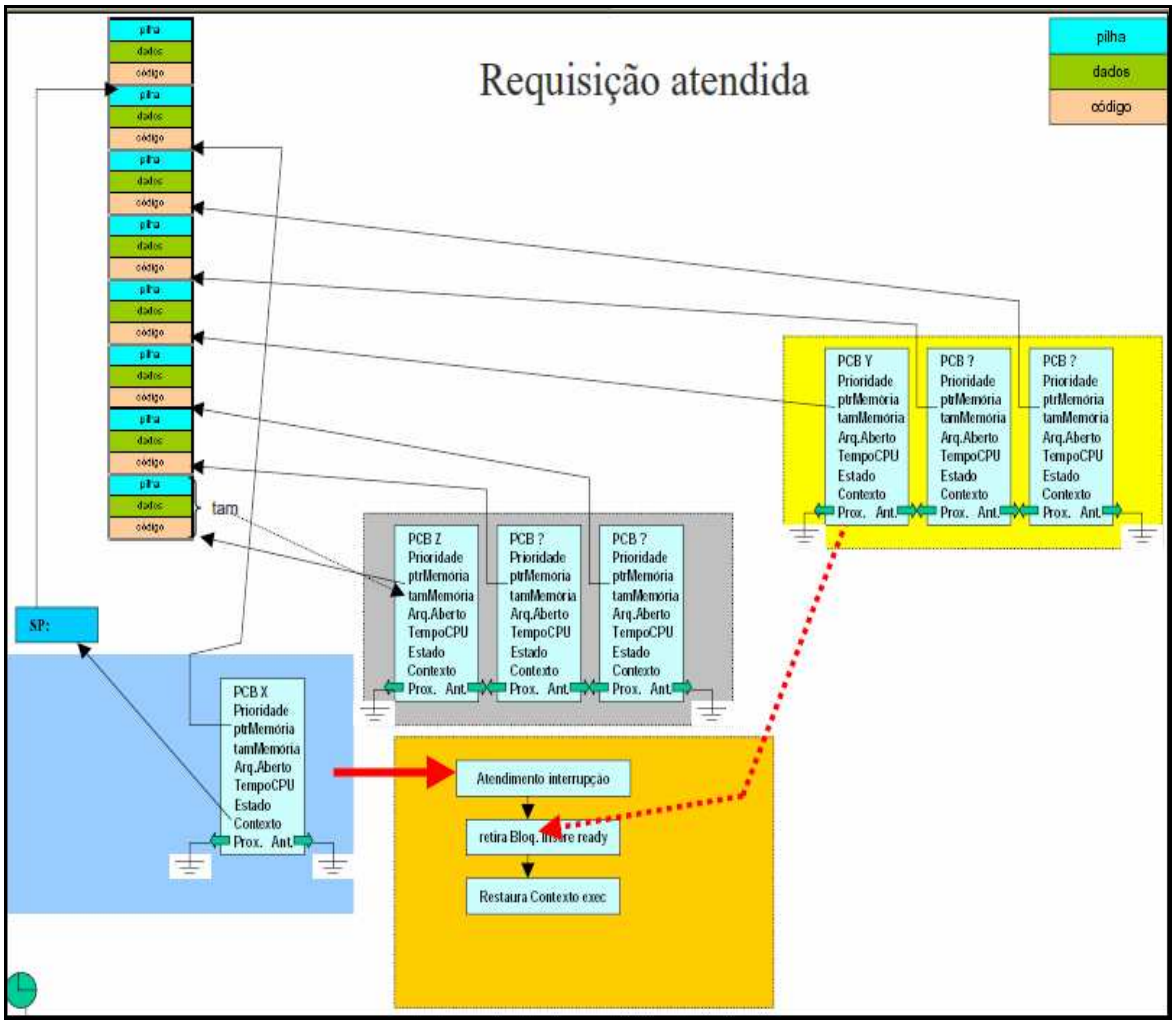


Figura 11 - Momento que a requisição de processamento é atendida

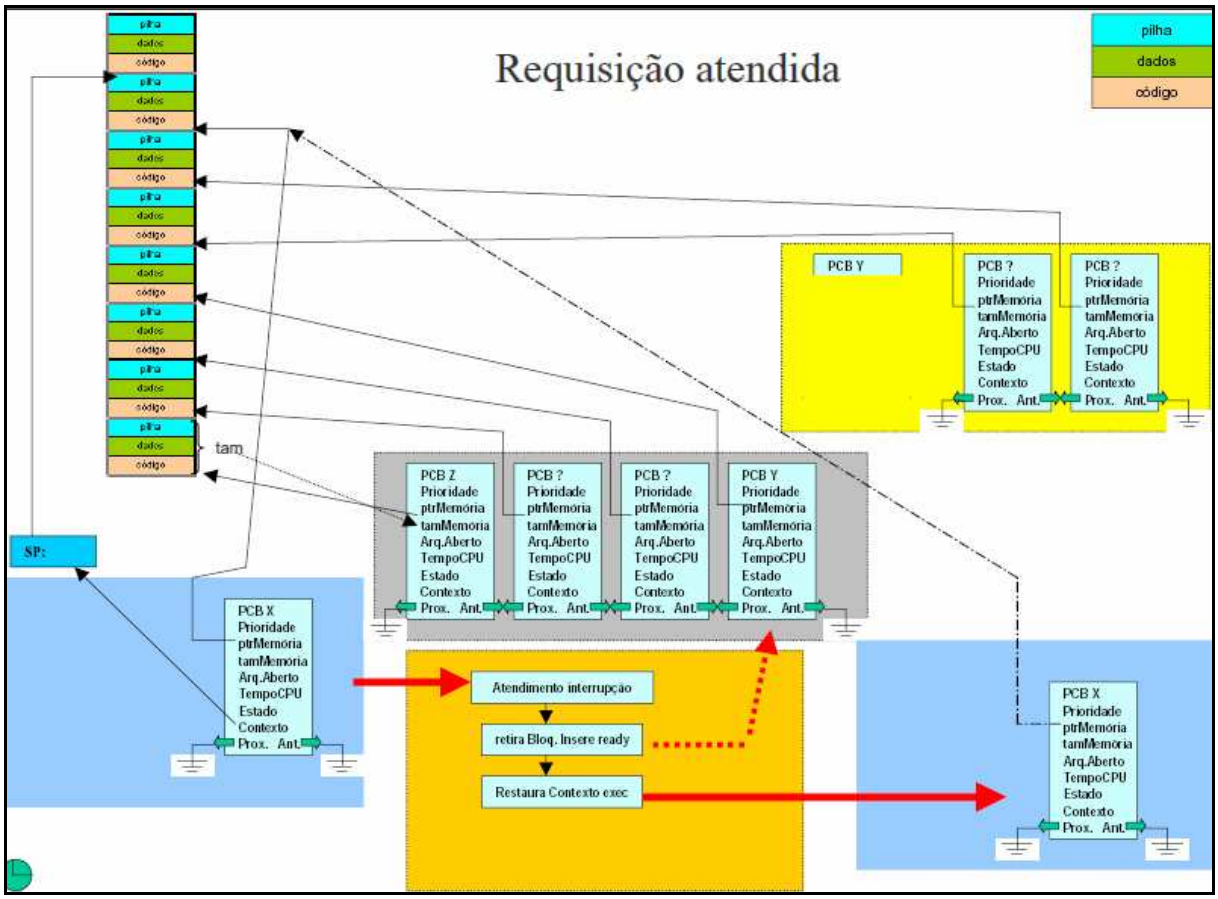


Figura 12 - Após a solicitação, a requisição é atendida e retorna ao final da fila de prontos

Uma vez que ficou caracterizada a lógica do funcionamento de um núcleo de sistema operacional, o professor Mauro Mattos lança mão de um outro recurso – planilhas de excel – para caracterizar aspectos específicos do funcionamento de um núcleo de multitarefa.

A figura 13 apresenta um exemplo de uma planilha que permite que o aluno registre os eventos que ocorrem em um núcleo de multitarefa hipotético. Este recurso permite ao aluno deparar-se com situações inusitadas como a ocorrência simultânea de eventos de hardware entre outros. Conforme destaca o Prof. Mauro Mattos, “embora estas situações sejam descritas na literatura o aluno tem dificuldades em visualizá-las”.

A Figura 13 caracteriza um cenário onde existem quatro processos: p1, p2, p3 e p4. Nesta planilha é registrado o tempo nominal de execução de cada processo supondo que o mesmo possui 100% dos recursos disponíveis, executa em um ambiente monotarefa e, os

periféricos e chamadas de sistema respondem instantaneamente. Além disso, a planilha permite o registro do momento aproximado (resolução de 5ms) onde ocorrerá, por exemplo, uma operação de leitura (r) ou escrita (w). No exemplo da Figura 13 está caracterizado que, após 30ms do início da sua execução (e, posteriormente após 60ms) , o processo P1 fará uma requisição de leitura à disco o que provavelmente o colocará na fila de bloqueados até o término da operação de E/S. Registra também que os processos P2 e P4 são *cpu-bound* pois não realizam operações de E/S e, caracteriza que o processo P3 realizará 3 operações de escrita respectivamente à 55ms, 85ms e 115ms do início de sua execução.

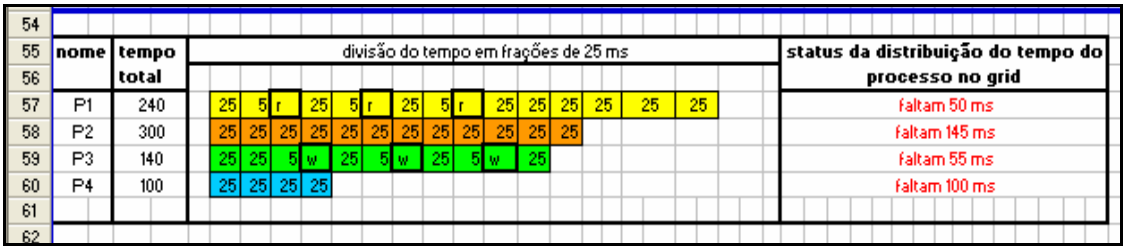


Figura 13 - Planilha que demonstra a configuração de 4 processos

A Figura 14 apresenta um exemplo de utilização da planilha. A meta do aluno é registrar, para cada processo, em que momento ele estava em estado “pronto” (P), executando (E) ou bloqueado (B). Esta planilha é dividida em duas áreas: uma área onde se registra o estado do processo em determinado momento caracterizando assim o tempo de CPU consumido em modo usuário e, uma área onde se registra o tempo gasto executando primitivas do sistema operacional (ex: criação de processo, atendimento de interrupção de relógio/periféricos, P e V, etc.). A planilha é dividida em unidades de 5 milisegundos por célula.

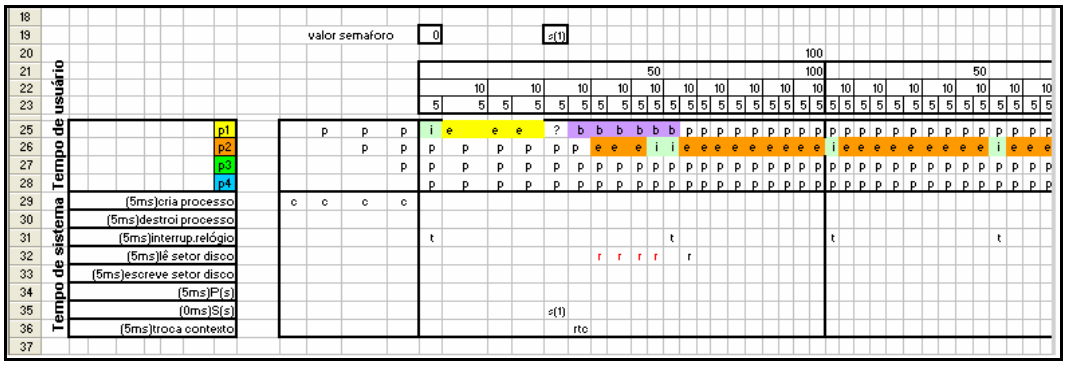


Figura 14 - Início da execução do processo de simulação

A coluna “status de distribuição do tempo do processo no grid” apresentada na Figura 13 é utilizada como forma de o aluno validar a sua resposta, ou seja, verificar se não alocou mais tempo de CPU que aquele necessário para a distribuição no tempo do tempo necessário para a finalização de determinado processo.

A Figura 15 apresenta as constantes utilizadas como referência para a demarcação de cada campo da planilha.

constantes	Tempo
time-slice	100
fracao	5
tempo méd.leitura	20
tempo méd.escr.	35
relogio	5
troca contexto	5
	0
	0
	0
	0
	0

legenda		
t	5	interrup.relógio
e	5	execute
p	0	pronto
b	0	bloqueado
bw	5	busywait
d	5	destroi processo
tc	5	troca contexto
rtc	5	progr.leitura e troca contexto
wtc	5	progr.escr. e troca contexto
c	5	cria processo
R	5	lê setor disco
W	5	escreve setor disco
P	5	P(s)
S	0	S(s)

Figura 15 - Constantes para refêrencias demarcações do campo da planilha

A Figura 16 caracteriza o início da execução do processo de simulação. Nesta planilha é possível identificar:

a) o valor dos semáforos utilizados na lógica de uma simulação de processos concorrentes;

b) as ocorrências de eventos síncronos (como relógio);

c) as ocorrências de eventos assíncronos (como chamadas de operações de E/S);

d) a demarcação de que processo está executando em determinado momento e o estado de execução dos demais processos ativos no sistema em determinado momento;

e) a indicação do que acontece quando duas interrupções ocorrem simultaneamente (exemplo: relógio e hardware indicando o término de uma operação de E/S);

f) o paralelismo entre o funcionamento da CPU e o funcionamento dos periféricos;

g) os tempos de execução em modo usuário e em modo sistema.

A Figura 16 caracteriza a execução entremeada de quatro processos em um ambiente de multitarefa e permite a caracterização da necessidade de um processo *IDLE* para manter o sistema operacional em funcionamento na eventualidade de todos os processos terem encerrado a sua execução ou ficarem bloqueados.

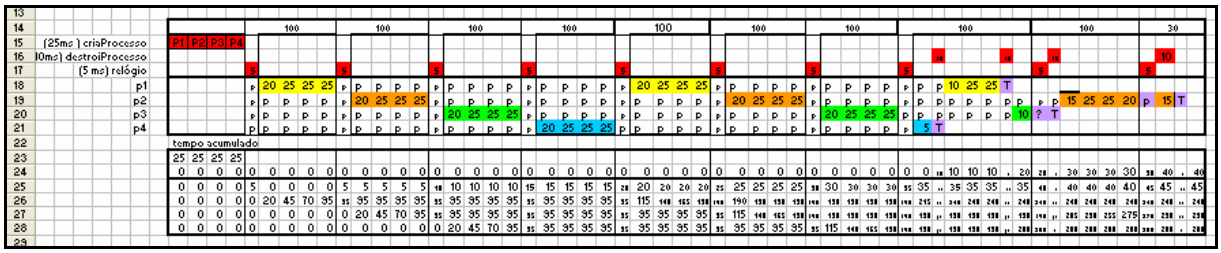


Figura 16 - Execução entremeada de quatro processos em um ambiente de multitarefa

Segundo afirmação do Prof. Mauro Mattos, a utilização das planilhas permite a visualização por parte do aluno (em um ambiente controlado pelo mesmo), da ocorrência dos eventos de um núcleo de sistema operacional preemptivo e a constatação de aspectos sutis que ficam camuflados durante a apresentação teórica dos conceitos ou durante o desenvolvimento de exercícios de programação concorrente em um sistema operacional real.

Ainda segundo o Prof. Mauro Mattos, “a aplicação das planilhas de simulação permitiu uma evolução na qualidade das discussões sobre questões de sincronização, *time-slice*,

eventos assíncronos, descritor de processos, filas de sistema e semáforos – questões estas que facilitaram o esclarecimento de aspectos específicos relativos ao funcionamento de um núcleo de sistema operacional, os quais, de outra forma tinham que ser abstraídos pelos alunos o que dificultava a compreensão do assunto e a assimilação do conteúdo".

4 POWERPOINT AUTOMATION

O Powerpoint faz parte da família de produtos da Microsoft, sendo parte integrante do pacote de ferramentas do Office. O Powerpoint é um programa de apresentação de imagens: um software que lhe ajuda na criação de apresentação de slides. Ele facilita a geração e a organização das idéias. Fornece ferramentas que podem ser utilizadas para criar os objetos que formam uma apresentação de slides eficaz (UNONIUS, 1997).

Permite compartilhar a apresentação com outras pessoas, independente de terem instalado o Powerpoint, também incluem ferramentas poderosas para gerenciamento da aplicação de slides, dando-lhe total controle sobre ela (UNONIUS, 1997).

O *automation* é uma característica do modelo de objeto componente (*Component Object Model* (COM)), uma tecnologia proprietária que utilizam seus objetos, métodos, ferramentas de desenvolvimento, macros, e às outras aplicações (RICE, 2005).

Quando uma aplicação suporta a automatização, os objetos que a aplicação expõe podem ser acessados com *Visual Basic(VB)*. É possível utilizar *Visual Basic* para manipular os objetos e métodos invocados, ou ajustar propriedades dos objetos. A fim compreender o *Automation*, é necessário compreender alguns conceitos e terminologia (RICE , 2005):

Object	Item que pode ser programado ou manipulado. Por exemplo, inclusão de objetos como textbox, combo box, botões, documentos Word e outros. Microsoft Office 2000 applications incluem outros 500 objetos.
Object Property	Uma propriedade é uma característica de um objeto. Por exemplo, as propriedades de um textbox incluem: Nome, visível, ForeColor e outros.
Object Method	Ação em que seja possível verificar o objeto. Por exemplo, o método de acesso para finalizar um form **Close**.
Automation Server	O *Automation Server* é a aplicação que mostra a automatização dos objetos.
Automation Client	O *"Automation Client"* é a aplicação que se decide quais objetos vai usar e quando os usar. Por exemplo, ao pressionar o botão de impressão para imprimir um arquivo a partir do Powerpoint: • Automation Server = Powerpoint • Automation Client = Access
Binding	Ajusta o tipo do objeto à variável do objeto.
Late Binding	*Late Binding* ocorre quando é declarado variáveis do objeto com uma classe específica e o *Late Binding* ocorre quando o código funciona *slower* (mais lento).
Early Binding	*Early Binding* ocorre quando é declarado variáveis do objeto com uma classe específica e o *Early Binding* ocorrer quando você compilar o código *faster* (mais rapidamente).

FONTE : (RICE, 2005)

O Powerpoint Automation trouxe soluções para acabar com trabalhos repetitivos do cotidiano. Este recurso disponibiliza duas formas de interação com o acesso ao PowerPoint. A primeira é criar uma apresentação de PowerPoint através dos dados de uma tabela do acesso usando a automatização, a segunda, e indicar e manipular uma apresentação existente de PowerPoint dentro de um formulário do acesso, também usando a automatização. A automatização lhe dá a agilidade de controlar uma aplicação a partir de outra, manipulando as propriedades expostas e métodos da aplicação controlada, e respondendo aos eventos .

Usar técnicas como estas permite ao usuário automatizar as tarefas que foram executadas manualmente no passado, assim ganhando tempo e adicionando as suas apresentações um toque profissional, (RICE, 2005).

4.1 MICROSOFT POWERPOINT OBJECT MODEL

De acordo com May (2004), no PowerPoint, o termo animação refere-se a efeitos sonoros e visuais que o usuário adiciona a um texto ou objeto. Efeitos de animação permitem mover o texto, figuras e outros conteúdos dos *slides*. Além disso, a adição de movimento permite ao usuário conduzir o foco da audiência para enfatizar pontos importantes, implementar transição entre slides e maximizar o espaço dos slides movendo componentes para dentro e para fora do mesmo.

Estes efeitos podem incluir como o objeto (*shape*) ou seus componentes entram no slide, o que o objeto faz quando aparece e como ele aparece quando a seqüência de animação move-se para o próximo objeto. Por exemplo, o trecho de código apresentado no quadro 3, exemplifica como selecionar um objeto como sendo o primeiro objeto a ser animado quando um *slide* é carregado e especifica para animar aquele objeto automaticamente 5 segundos após o *slide* ter sido carregado.

```
Dim objShape As Shape
With ActivePresentation.Slides(1).Shapes

  Set objShape = .Item("Title")
     With objShape.AnimationSettings
       .EntryEffect = ppEffectBlindsHorizontal
       .AnimationOrder = 1
       .AdvanceMode = ppAdvanceOnTime
       .AdvanceTime = 5
      End With
End With
```

Fonte: May (2004)

Figura 17 - Exemplo de controle de animação do PowerPoint através de programação

Para o desenvolvimento da ferramenta foram utilizadas inúmeras funções para alterar as propriedades, dos objetos do PowerPoint automation.

Estas propriedades são responsáveis pela alteração de cores dos objetos,

movimentação, incremento dos diversos contadores, alteração da posição dos objetos, mostrando as facilidades de se trabalhar com estas propriedades. A seguir são relacionadas, algumas destas citadas acima.

Criação de um objeto :

a) Quadrado:
    a. ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeRectangle, 558#, 78#, 48#, 102#).Select;
b) WordArt:
    a. ActiveWindow.Selection.SlideRange.Shapes.AddTextEffect(msoTextEffect2, "Furb", "Arial Black", 36#, msoFalse, msoFalse, 319.12, 223.5).Select;
c) Selecionar e soltar:
    a. ActiveWindow.Selection.SlideRange.Shapes("Rectangle 443").Select;
    b. ActiveWindow.Selection.Unselect;

d) Manipulação de objeto:

e) Alteração de Cores:
    a. Fill.ForeColor.SchemeColor = ppFill;
    b. Fill.ForeColor.SchemeColor = ppAccent1;
f) Posicionamento:
    a. .Left = Horizontal_1;
    b. .Top = Vertical_1;
g) Chamada de Função:
    a. Call;
h) Controle de tempo:
    a. Sleep (1000);
i) Efeitos de texto:
    a. .TextEffect.Text = ArmLength;
    b. .TextFrame.TextRange.Text = "write";

5 DESENVOLVIMENTO DO TRABALHO

Neste capítulo são apresentados os requisitos, a especificação, tecnologias utilizadas, implementação, operacionalidade do sistema e funcionalidades da ferramenta criada, resultado e discussões.

5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

A seguir estão classificados em funcionais e não funcionais os principais requisitos para o desenvolvimento do protótipo.

O quadro 1 apresenta os requisitos funcionais previstos para o sistema, identificando os requisitos que deverão ser implementados.

Requisitos Funcionais
RF01: o software deverá permitir ao usuário configurar as características do ambiente de animação.
RF02: o software deverá fazer a leitura de um *log* contendo estados dos vários componentes do ambiente que serão previamente definidos.
RF03: o software deverá interpretar o *log* lido.
RF04: o software deverá possibilitar demonstrar o movimento dos componentes do núcleo de um sistema operacional.

Quadro 1 - Requisitos Funcionais

O quadro 2 lista os requisitos não funcionais previstos para o sistema, identificando os requisitos que serão contemplados na implementação.

Requisitos Não Funcionais
RNF01: confiabilidade: o ambiente de simulação deve refletir os resultados da simulação da ferramenta que produz os *logs* a serem processados pelo simulador.
RNF02: performance: a gerência dos eventos de simulação não deve impactar no tempo de resposta da simulação para não comprometer o objetivo final que é a apropriação dos conceitos por parte do aluno.
RNF03: usabilidade: o simulador deve possuir uma interface gráfica de aparência “amigável” de modo a despertar interesse do aluno na utilização do mesmo.
RNF04: a ferramenta utilizada para o desenvolvimento do ambiente será o Delphi.
RNF05: a ferramenta utilizará a API do pacote *PowerPoint Office Automation* da *Microsoft*.

Quadro 2 - Requisitos não funcionais

A ferramenta criada tem como objetivo auxiliar o entendimento do fluxo das informações no núcleo do sistema operacional, tornando-se mais dinâmico pelo apelo visual e abstração de eventos já conhecidos.

O acadêmico usuário do sistema deverá configurar o comportamento dos processos antes de iníciar a animação.

É de responsabilidade do acadêmico informar através dos botões da tela inicial se o comportamento dos processos será de execute, *read*, *write*, P, V.

O Acadêmico submete suas coordenadas a verificação e só então torna-se possível ver o cenário de animação e os elementos que o compõe.

Todo o fluxo da informação toma como partida um *log* que foi gerado a partir das informações inseridas pelo acadêmico.

Neste *log* é possível registrarem-se quatro situações que nortearão o mecanismo de animação:

a) situação 1: o sistema encontra um identificador chamado de "troca de contexto" e alterna as posições, colocando o processo que estava utilizando a CPU na última posição da fila de processos prontos e o primeiro da fila de prontos recebe a CPU. Isto causa um deslocamento de todos os demais processos uma posição à frente. Este ciclo de troca de contexto é iniciado sempre que encontra-se um identificador de "troca de contexto" no *log* e, o tempo do relógio chega a zero;
b) situação 2: o sistema encontra uma instrução representando uma operação de entrada/saída (*read* ou *write).* Imediatamente o descritor do processo atual é inserido na fila de bloqueados e um contador regressivo de tempo decorrido para o término da operação de E/S é iniciado. Ao final deste tempo decorrido para a execução da operação de E/S, o descritor do processo é removido da fila de bloqueados e inserido no final da fila de prontos;
c) situação 3: o sistema encontra uma instrução representando uma operação bloqueante da primitiva “P”. Neste caso o descritor do processo é inserido na fila do semáforo e lá ficará até que uma operação “V” seja encontrada no *log*. Cabe destacar que, chamadas à primitiva P não bloqueante (quando o semáforo permite a continuidade da execução) não são representadas na animação;
d) situação 4: o sistema encontra uma instrução que representa uma operação de chamada à primitiva “V”. Neste caso o primeiro descritor de processos da fila de semáforos é retirado da mesma e inserido no final da fila de prontos. A situação onde “V” é

chamado e nada ocorre por não haver processos na fila de processos do semáforo não é representada na animação.

5.2 ESPECIFICAÇÃO

A seguir são apresentados os casos de uso que descrevem o funcionamento do sistema sob a perspectiva do usuário.

5.2.1 CASOS DE USO

Na lista a seguir, são identificados os acontecimentos desde a configuração inicial do sistema:

a) acadêmico configura os processos;

b) acadêmico dispara a animação.

No primeiro cenário, o aluno configura a seqüência de escalonamento de acordo com o exercício a ser animado. Isto implica em cadastrar os processos que definem um grupo de problemas. No cadastro de sub-processos, é detalhada uma situação para o evento “pai” do processo, onde será informada também a solução que deve ser tomada para este sub-processo, caso a ação “processo” ocorra para este sub-processo.

No segundo cenário dispara-se a animação. Neste momento a ferramenta escrita em *Delphi* faz validação do *log* antes de carregar a apresentação.

Uma vez iniciada a animação, a ferramenta escrita em *Visual Basic* lê o *log* e desliza os objetos iniciando a animação dos processos.

5.3 IMPLEMENTAÇÃO

O sistema foi concebido na forma de dois módulos: a interface de simulação e a apresentação PowerPoint.

5.3.1 INTERFACE DE SIMULAÇÃO

A proposta original envolvia o desenvolvimento de uma interface de animação do funcionamento do núcleo de um sistema operacional a partir de um *log* gerado por uma aplicação específica a ser construída na forma de um TCC. Contudo, tendo em vista que o TCC não foi concluído, surgiu a necessidade do desenvolvimento de uma interface que simulasse o funcionamento de tal programa.

Assim sendo, desenvolveu-se uma aplicação a qual é responsável por permitir ao professor a configuração da seqüência de execução dos eventos de simulação.

A saída desta interface é um arquivo texto o qual contém *tags* que identificam os seguintes eventos:

a) Troca contexto : evento que chaveia o simulador para executar uma troca de contexto entre o processo que está executando e o primeiro processo da fila de prontos;
b) Read : indica que o processo atual realizou uma chamada de sistema para a execução de uma operação de leitura em disco. “Fila de bloqueados”
c) Write : indica que o processo atual realizou uma chamada de sistema para a execução de uma operação de escrita em disco. “Fila de bloqueados”
d) P : indica que o processo realizou uma chamada de sistema para a execução da primitiva P (seção 2.3.2).
e) V : indica que o processo realizou uma chamada de sistema para a execução da

primitiva V (seção 2.3.2).

A Figura 18 apresenta a tela da interface de geração do *log* de execução.

Ao clicar em qualquer botão insere-se um novo evento no *log.* É possível limpar todo o *log* ou edita-lo para melhor visualização, e ao clicar no botão "*EXECUTE POWERPOINT*", as instruções serão validadas e a apresentação carregada com o estado inicial dos processos e aguardando o *click* ao botão "simulação".

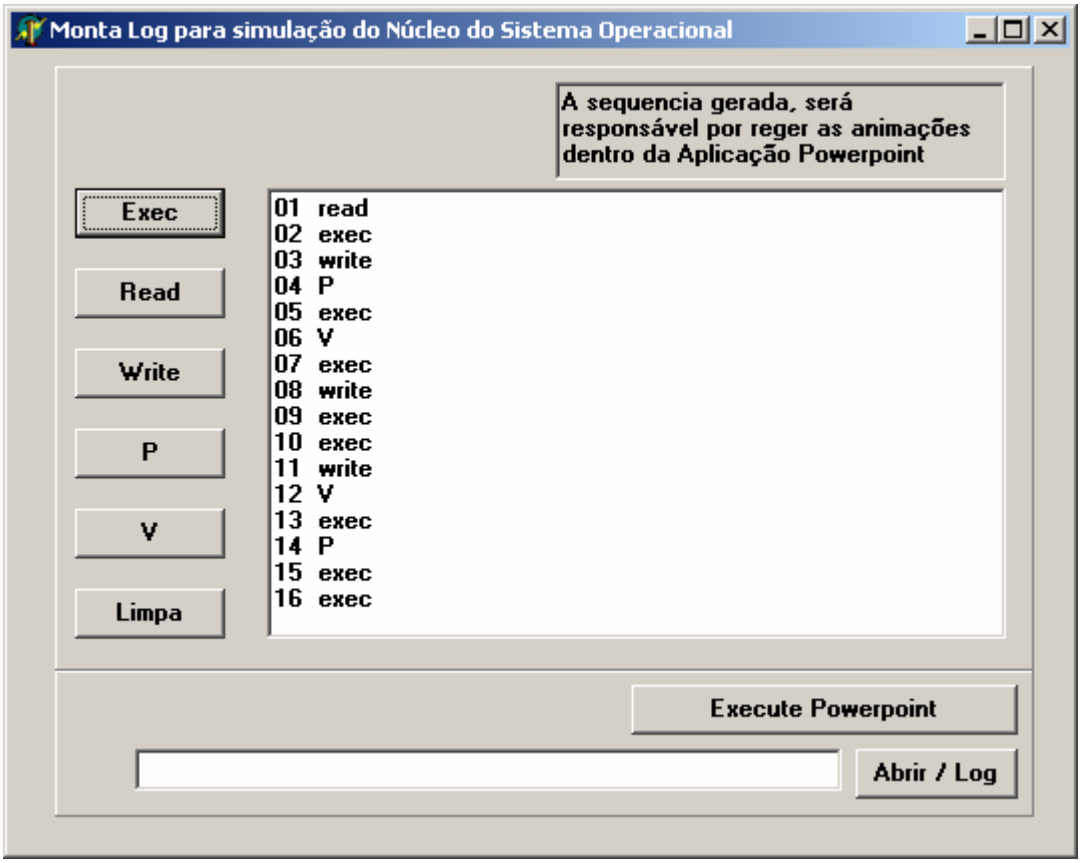


Figura 18 - Tela responsável em montar um *log* para seqüência de animação

No exemplo da Figura 18, ao iniciar a simulação, o processo que está executando realiza uma operação de leitura. Isto implica que uma troca de contexto será efetuada (*exec*). O próximo processo inicia a sua execução e também realiza uma chamada de sistema (*write*) para a execução de uma operação de E/S levando-o a ficar bloqueado. O núcleo escalona o próximo processo o qual realiza uma chamada de sistema para a execução da operação P. O processo que estava na CPU fica aguardando até que uma operação V seja requisitada.

O funcionamento completo de uma sessão de simulação será descrito a seguir.

5.3.2 APRESENTAÇÃO *POWERPOINT*

A apresentação foi desenvolvida utilizando recursos de programação *do Visual Basic for Applications* (VBA) , para implementar a leitura e lógica dos movimentos da animação.

A Figura 19 apresenta o cenário de simulação descrito em Mattos (2004).

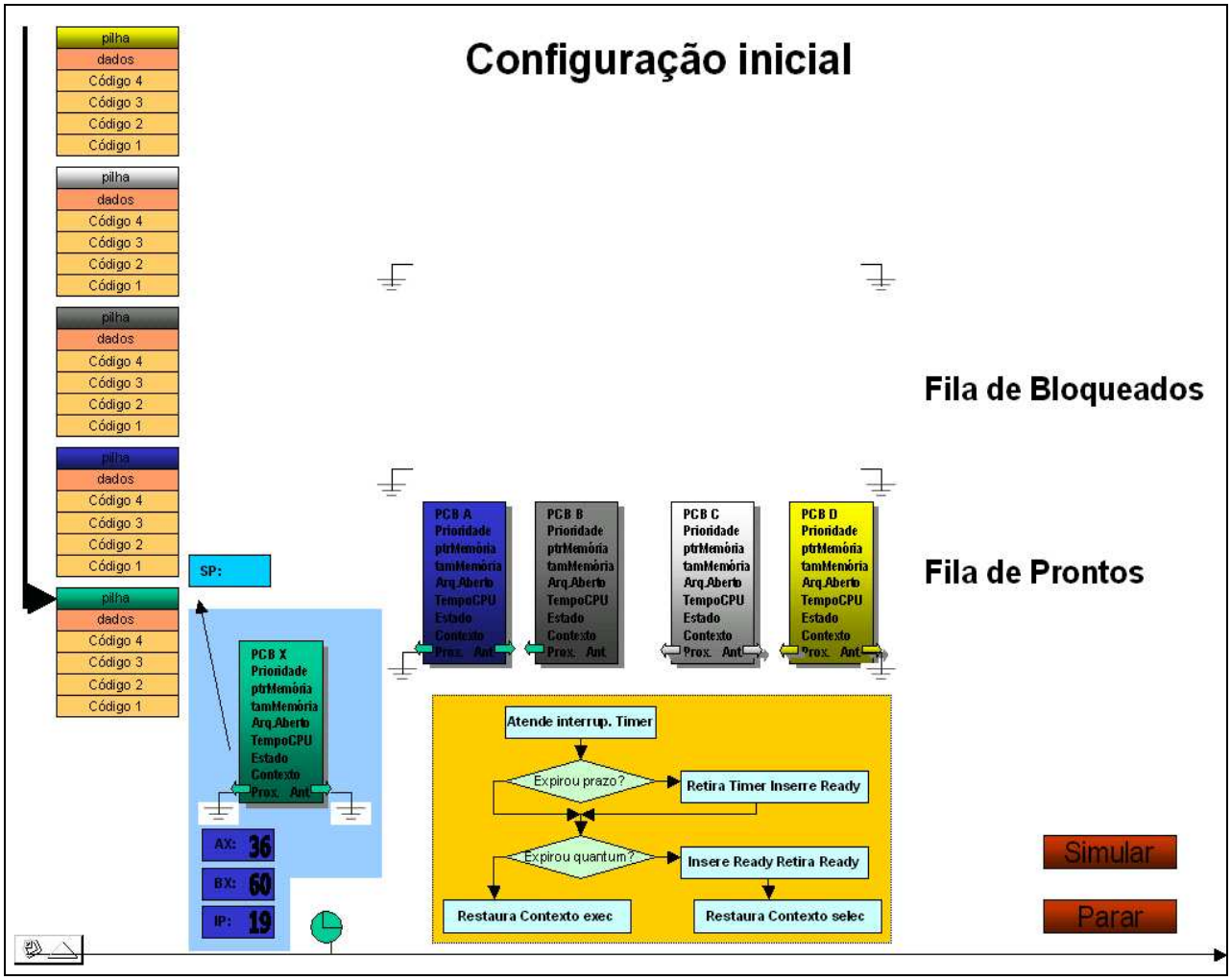


Figura 19 - Estado inicial do modelo antes de iniciar animação

Na figura é possível identificarem-se os seguintes elementos:

a) Hardware:

- Processador;

- Registradores: AX.BX,IP (*instruction pointer*) e SP (*stack pointer*);
- Memória física contendo a indicação para código, dados e pilha de execução de cada um dos processos envolvidos na simulação;
- Um *slider* indicando a área de memória sendo referenciada pelo processo atual;
- Um relógio que sinaliza uma interrupção do *clock* do sistema.

b) Software:

- Blocos descritores de processos (a,b,c,d,e,f), respectivamente representados pelos retângulos verde, azul, cinza, branco e amarelo;
- A fila de processos prontos para a execução;
- A fila de processos bloqueados por eventos de E/S;
- A fila de processos bloqueados pela execução de chamadas de primitivas de sincronização (P e V);
- A lógica do tratador de uma interrupção de relógio.

O cenário inicial apresenta um processo executando e quatro processos aguardando na fila de prontos.

Um dos principais elementos da animação é o relógio de tempo real. Este tem função de indicar a fatia de tempo que cada processo tem direito na CPU.

Após pressionar o botão de Simular os processos são iniciados e a animação começa. O exemplo da Figura 20 retrata a simulação da operação de “Troca contexto”. Isto implica na substituição entre o bloco descritor do processo que perdeu a CPU pelo primeiro processo da fila de prontos. Esta figura destaca um processo (identificador 1) que estava ocupando a CPU, foi movido para o final da fila de processos prontos, deixando a CPU para o primeiro processo da fila de prontos (representado pelo identificador 2).

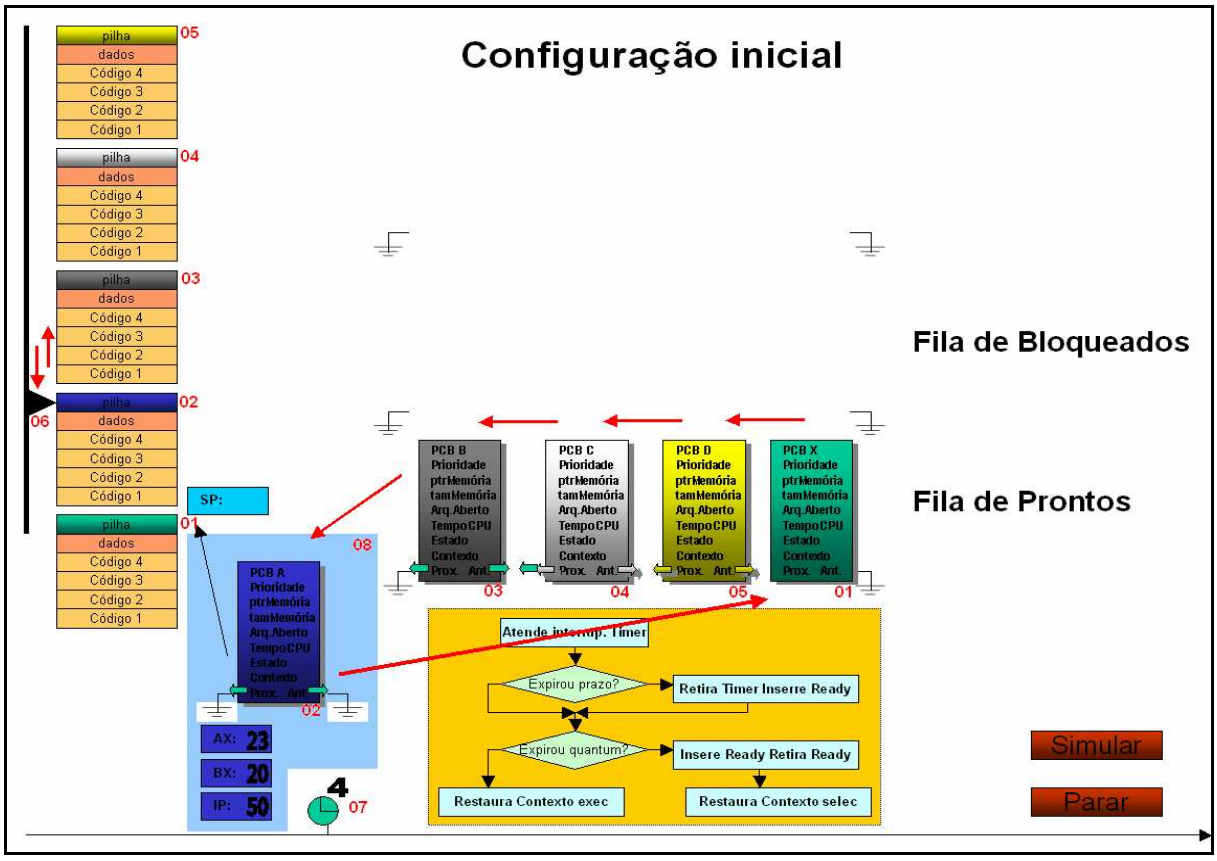


Figura 20 - Estado da animação para o ciclo "Troca contexto"

Observe-se que esta ação faz com que todos os demais processos ocupem novas posições na fila, o processo B(identificador 03) no lugar do A (identificador 02), o C (identificador 04) no lugar do B (identificador 03), o D (identificador 05) no lugar do C (identificador 04) e o X (identificador 01) no lugar do D (identificador 05).

Para apontar para a área de memória referenciada pelo processo que está executando no momento, existe uma barra deslizante que na figura 20 esta destacada com o (identificador 6). Os movimentos desta barra acontecem apenas verticalmente.

O relógio que está destacado com o identificador 07 na figura, é quem determina o momento da troca de contexto. Este contador deve decrescer de 05 até 00 (valor este configurável), quando deve começar um novo ciclo de execução.

Quando a aplicação VBA lê o *log* que identifica a seqüência de execução ela adianta um movimento para saber qual será sua próxima ação. Isto se faz necessário para o adequado tratamento da operação "*read*".

Quando isto acontece o descritor do processo atual é inserido na fila de bloqueados e não há a necessidade de aguardar o relógio atingir o valor 00 para ser disparada a ação de troca de contexto.

A Figura 21 apresenta o cenário em que um processo foi bloqueado por uma chamada de sistema para a realização de uma operação de entrada/saída. Observe-se que, no momento em que um processo é bloqueado nesta situação, inicia-se uma contagem regressiva representando o tempo restante até que a operação seja completada. Este contador é identificado por um rótulo (identificador 8) indicando o tipo de operação e um contador (identificador 9) caracterizando o tempo de espera para o término da operação.

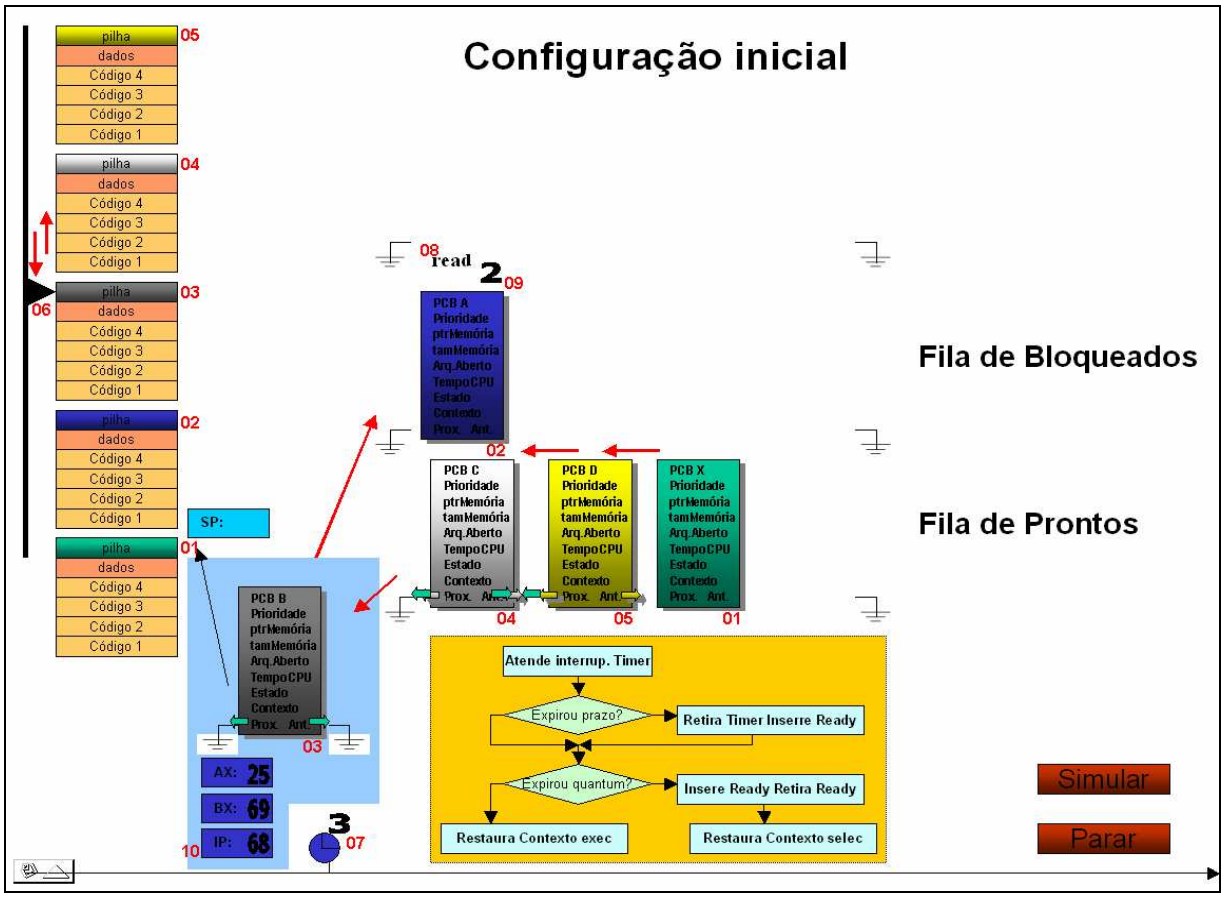


Figura 21 - Funcionamento do ciclo "*read*"

Este contador é regressivo e começa em 07 e vai até 00, quando o processo volta à fila de prontos.

O relógio do sistema (identificador 7) possui um contador regressivo o qual indica a quantidade de tempo que ainda resta ao processo atual antes da retomada do processador pelo núcleo do sistema, este contador ao chegar a zero executa uma nova operação. Na figura 21 está o estado final de uma operação que demonstra o processo representado pelo identificador 02 na figura 21, na área de fila de bloqueados, ou seja, acabou de deixar a CPU, que passa a ser ocupada pelo processo de numero 03, fazendo com que os processos de numero 04, 05 e 06 se deslocarem para a esquerda preenchido o espaço deixado pelo processo 03.

Os contadores representados pelo indicador de numero 10, apenas simulam o funcionamento da CPU, utilizando números aleatórios que representam a alteração de conteúdo dos registradores da CPU durante a execução de um processo.

Pode-se ter outros processos também aguardando na fila de bloqueados. Existe um *timer* para cada processo o qual ao atingir o valor zero indica que este processo será inserido na fila de prontos.

A figura 21 demonstra os caminhos da informação no caso de uma operação "*read*". A diferença da animação do "*read*" para o "*write*", é o tempo do *timer* individual de cada processo, como mostram os identificadores 10 e 11 da Figura 22.

Um aspecto importante a ser destacado refere-se ao fato de que mais de um processo pode estar na fila de bloqueados, cada um com seu *timer* individual, o que caracteriza um funcionamento assíncrono dos dispositivos de E/S (Figura 22 rótulos 8 e 9).

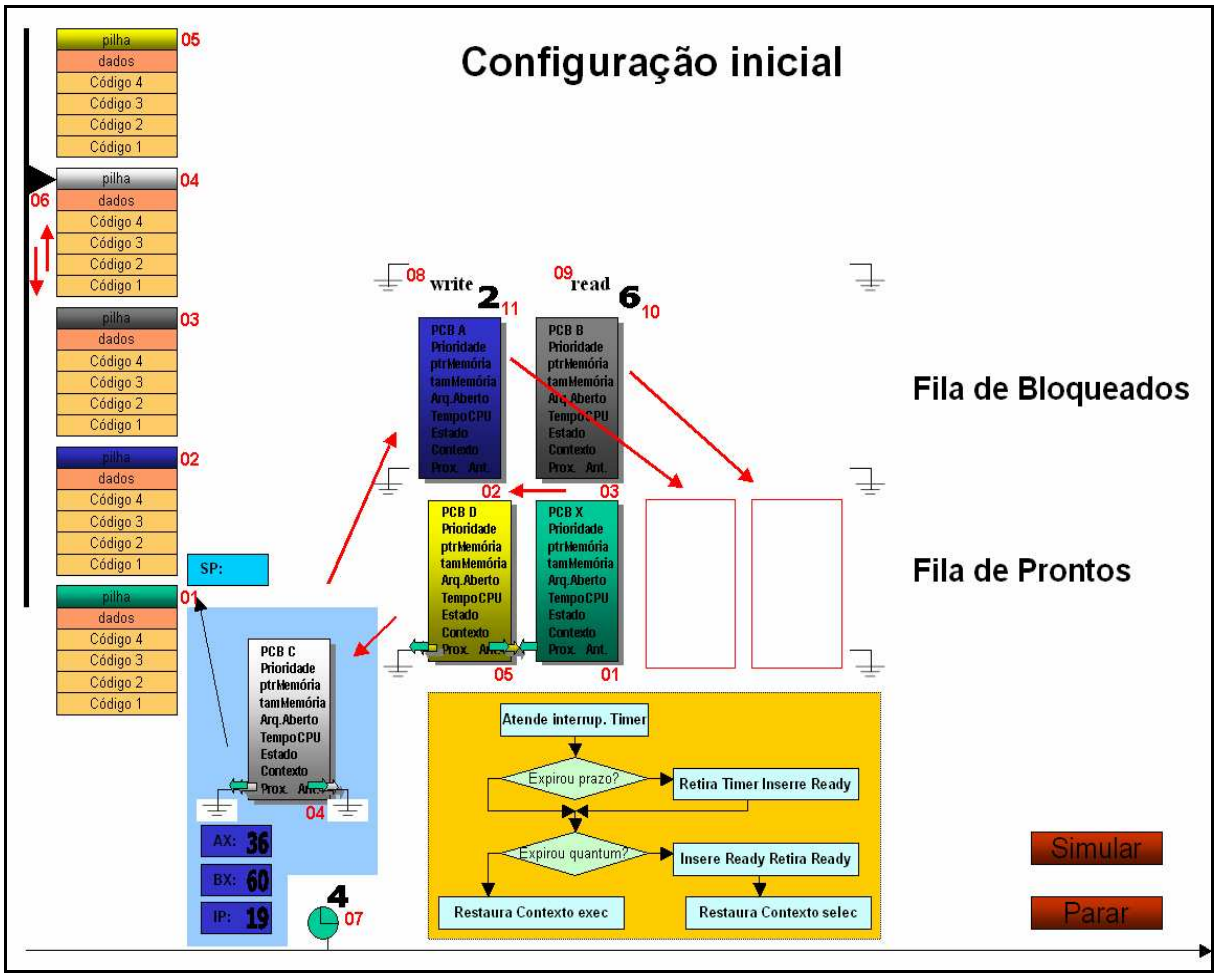


Figura 22 - Funcionamento do ciclo "*write*"

A Figura 23 caracteriza o momento em que cria um processo bloqueado após ter sido executado uma operação "P" o que significa que o semáforo esta fechado bloqueando o processo que chamava a primitiva.

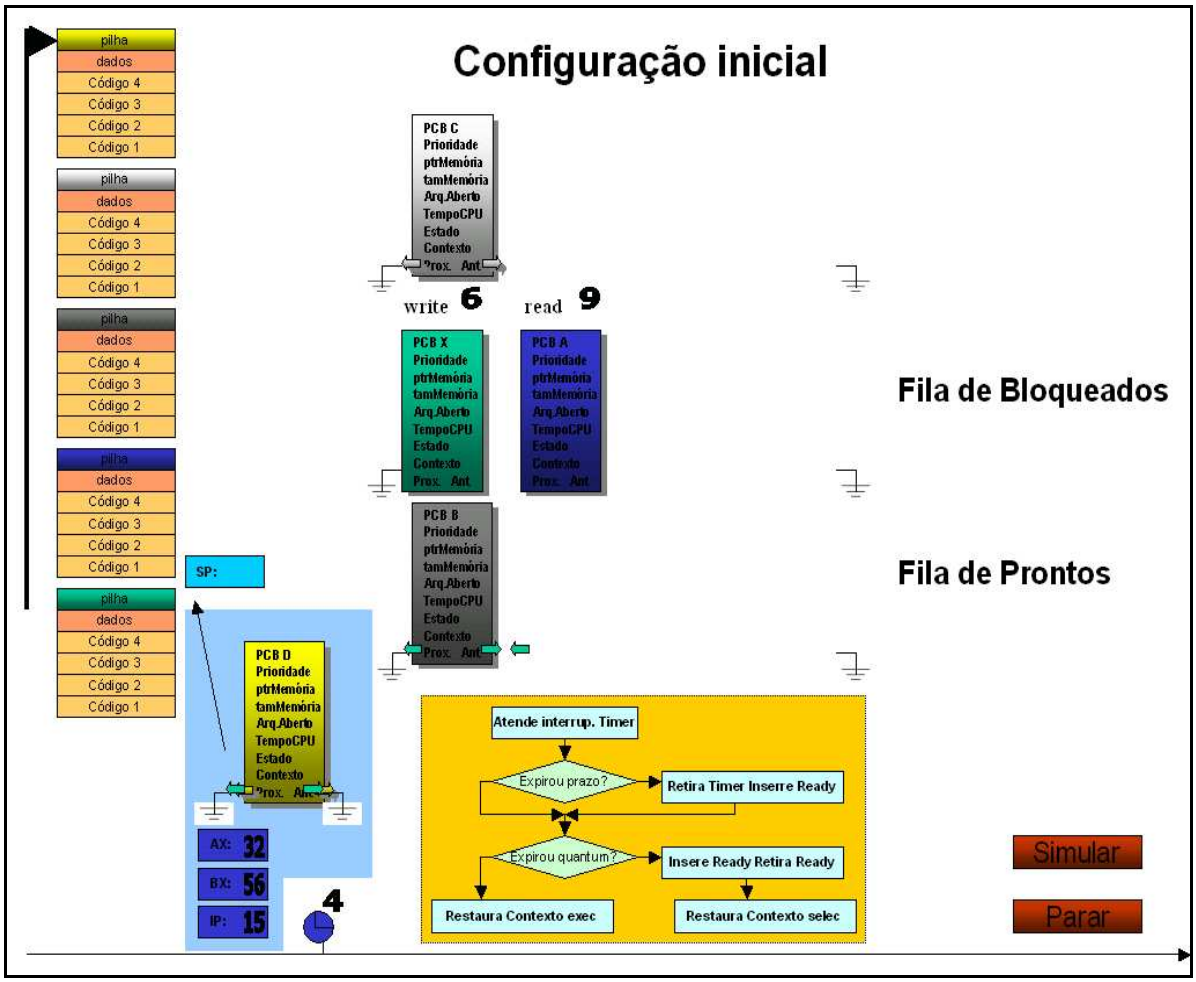


Figura 23 – Processo que executou uma operação P bloqueante

5.3.3 MACROS DESENVOLVIDA

O trecho de código da Figura 24, exibido abaixo, foi implementado utilizando o *Borland Delphi 5* e mostra como é realizada a chamada do aplicativo *Powerpoint*.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  PowerPointApp: OLEVariant;
begin
  try
    PowerPointApp := CreateOleObject('PowerPoint.Application');
  except
    ShowMessage('Error...');
    Exit;
  end;

  // Powerpoint visivel
  PowerPointApp.Visible := True;

  // Abre a apresentação
  PowerPointApp.Presentations.Open(ExtractFilePath(Edit1.Text)+'Nucleo_fonte.ppt', False,
False, True);

  // executa apresentação
  PowerPointApp.ActivePresentation.SlideShowSettings.Run;

end;
```

Figura 24 Código referente a chamada do Aplicativo Powerpoint

A apresentação *Powerpoint*, foi desenvolvida utilizando-se macros, e valendo-se dos recursos da biblioteca *Office* A*utomation.*

Como a Figura 25 demonstra, o código fonte desenvolvido em VBA que faz a abertura do arquivo, e começa a lê-lo seqüencialmente até o fim, quando a animação se encerrará. Note que a leitura está sempre antecipando o movimento para prever os casos de "*read*" e "*write*", que não aguardam que o relógio do sistema atinja o valor 0 (zero) para realizar uma troca de contexto.

```
 CONTA_LINHA = 1
 'Abrir o arquivo
 Open Path & "SEQUENCIA.TXT" For Input As #CONTA_LINHA
 Line Input #1, linha
 Animacao = Mid(linha, 4, 15)
 Animacao_Proxima = Animacao
 'fazendo o loop
 Do While Not EOF(1)

  Do Until Stopped
   'pega as 15 primeiras posiçòes e joga na variavel
   Animacao = Animacao_Proxima
   CONTA_LINHA = CONTA_LINHA + 1
   Line Input #1, linha
   Animacao_Proxima = Mid(linha, 4, 15)

If Animacao = "exec" Then ' A CPU VAI PARA O FINAL DA FILA
```

Figura 25 - Abre o arquivo e lê mesmo seqüencialmente a cada passada do *loop*

Através da montagem do *layout*, com o posicionamento das caixas retangulares que representam os processos, é definido o início do desenvolvimento das rotinas *automation*, pois utiliza a gravação de macros para capturar o nome e a posição inicial dos objetos gerados com o código que é mostrado na figura 26. O código desta mesma figura demonstra que um retângulo foi criado e logo após selecionado, desta forma fica explicito o posicionamento e o nome gerado pelo própria gravação automática da macro.

```
Sub Macro1()
'
' Macro gravada em 23/6/2005 por Microsoft
'
    ' Posição dos objetos
    ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeRectangle, _
                                                      318#, 186#, 60#, 132#).Select
    ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeRectangle, _
                                                      390#, 186#, 60#, 132#).Select
    ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeRectangle, _
                                                      462#, 186#, 60#, 132#).Select
    ActiveWindow.Selection.SlideRange.Shapes.AddShape(msoShapeRectangle, _
                                                      228#, 318#, 72#, 144#).Select

    ' nome dos objetos gerado pelo proprio automation
    ActiveWindow.Selection.SlideRange.Shapes("Rectangle 5").Select
    ActiveWindow.Selection.SlideRange.Shapes("Rectangle 2").Select
    ActiveWindow.Selection.SlideRange.Shapes("Rectangle 3").Select
    ActiveWindow.Selection.SlideRange.Shapes("Rectangle 4").Select
    ActiveWindow.Selection.SlideRange.Shapes.AddTextEffect _
                    (msoTextEffect2, "Furb", "Arial Black", _
                     36#, msoFalse, msoFalse, 316.12, 212.88).Select
    With ActiveWindow.Selection.ShapeRange
        .IncrementLeft -106.12
        .IncrementTop -152.88
    End With
End Sub
```

Figura 26 - Informações geradas automática de uma macro. Ex: nome e posição do objeto

Inicialmente para cada objeto da apresentação que deve ser deslocado, foram atribuídos valores para as posições verticais e horizontais. Antes mesmo de ler o arquivo que define os caminhos dos movimentos, é feito uma chamada de todos os objetos passando seus valores iniciais, para que assumam suas posições de saída. Como é um arquivo de *PowerPoint*, interromper a qualquer momento faz com que os objetos fiquem parados

exatamente onde foram interrompidos. Esta primeira chamada garante que todos estarão exatamente em seus lugares na leitura da primeira linha do *log (*Figura 27*).*

```
.
.
.
Call ShowHand(ActivePresentation.Slides(1).Shapes("Group 24"), _
            Amarelo_h, Amarelo_v, 114, 24)

Call ShowHand(ActivePresentation.Slides(1).Shapes("Group 8"), _

            Branco_h, Branco_v, 114, 24)

Call ShowHand(ActivePresentation.Slides(1).Shapes("Rectangle 29"), _
            Cinza_h, Cinza_v, 114, 24)

Call ShowHand(ActivePresentation.Slides(1).Shapes("Rectangle 253"), _
            Azul_h, Azul_v, 114, 24)

Call ShowHand(ActivePresentation.Slides(1).Shapes("Rectangle 257"), _
            Verde_h, Verde_v, 114, 24)

Call ShowHand(ActivePresentation.Slides(1).Shapes("Group 291"), _
            504, 180, w_cor_relogio, 1)

Call ShowHand(ActivePresentation.Slides(1).Shapes("WordArt 325"), _
            492, 192, 99, 0)

Call ShowHand(ActivePresentation.Slides(1).Shapes("WordArt 330"), _
            162, 288, 99, 0)

Call ShowHand(ActivePresentation.Slides(1).Shapes("WordArt 391"), _
            162, 358, 99, 0)

Call ShowHand(ActivePresentation.Slides(1).Shapes("WordArt 392"), _
            162, 450, 99, 0)

Call ShowHand(ActivePresentation.Slides(1).Shapes("Line 364"), _
            330, 18, 88, 0)
.
.
.
```

Figura 27 - Chamada para alinhar os objetos em suas posições iniciais

O código que aparece na Figura 28 é referente a rotina de “Troca contexto”, onde um novo processo assume a CPU e o que estava nela no momento vai para o final da Fila de

Prontos. A figura mostra a comparação da posição 5, que é a última posição caso todos os processos estivessem na fila de Prontos, exceto o que está na CPU.

O processo utiliza uma triangulação entre outra variável (posição_5v), comparando o valor da variável com os nomes dos objetos para saber qual processo virá para a posição 5. Em seguida verifica para onde deve deslocar o ponteiro vertical que indica a área de memória utilizada pelo processo que está usando a CPU, e abre um *loop* que fará as várias chamadas causando o movimento da tela, até que cada objeto assuma sua nova posição. Este procedimento ocorre para todas as posições da tela com este mesmo tratamento para cada um dos processos.

```
If POSICAO_5v = "0" Then ' qual processo irá para a posição 5
        POSICAO_5v = POSICAO_1
        POSICAO_1 = POSICAO_2
        POSICAO_2 = POSICAO_3
        POSICAO_3 = POSICAO_4
        POSICAO_4 = POSICAO_5
        POSICAO_5 = POSICAO_5v
        POSICAO_5v = "1"
End If
' ...
If POSICAO_5 = "PCB_X" Then  ' barra vertical que indica o
processo da CPU
   OBJETO_POSICAO_5 = "Rectangle 257"
End If
' ...
If POSICAO_1 = "PCB_X" Then
   MOVE_PONTEIRO = 330
End If
' ...
While MOVE_5_V <> 456    ' loop que altera os valores fazendo
o movimento

  If (MOVE_1_H <> 354) Then
     MOVE_1_H = MOVE_1_H + 2
  End If
  If (MOVE_1_V <> 138) Then
     MOVE_1_V = MOVE_1_V - 2
  End If
  ' ...
  If OBJETO_POSICAO_5 <> "0" Then
     Call
ShowHand(ActivePresentation.Slides(1).Shapes(OBJETO_POSICAO_5
), _
     MOVE_5_H, MOVE_5_V, 114, 24)
  End If
  For I = 1 To 1
     Sleep (10)
     DoEvents
  Next
 Wend
End If
```

Figura 28 - Acerta os valores antes da chamada da rotina que movimenta os objetos

A Sub-rotina "*showhand*" (Figura 29) é a responsável por receber os parâmetros passados a ela e alterar as propriedades de cada objeto, tantas vezes quantas for solicitada até que cada objeto esteja no seu devido lugar. É nesta rotina que todas as propriedades são alteradas, como por exemplo; posição de cada processo, cor do relógio, contadores, barra vertical que indica processo na CPU, etc.

Os dois últimos parâmetros passados para a sub-rotina são parâmetros de controle interno, que determinam uma ação de acordo com o valor passado, por exemplo: é através destes parâmetros que os rótulos “*read*” ou “*write*” são escritos no campo respectivo ao lado do contador, ou se é necessário alterar a cor do relógio de verde para azul na ocorrência de uma interrupção de relógio.

```
Sub ShowHand(Hand As Shape, ByVal Degrees As Integer, _
    ByVal HandCenterX As Integer, ByVal HandCenterY As
Integer, _
    ByVal ArmLength As Integer)

    Dim Vertical_1 As Integer
    Dim Horizontal_1 As Integer
    Dim Top As Integer
    Dim Height As Integer
    Dim Width As Integer

    With Hand
        Horizontal_1 = HandCenterX
        Vertical_1 = Degrees

        If HandCenterY = 0 Then
          'Muda para azul a cor do relogio
          .Fill.ForeColor.SchemeColor = ppFill
        End If

        If HandCenterY = 1 Then
          'Muda para verde a cor do relogio
          .Fill.ForeColor.SchemeColor = ppAccent1
        End If
        'responsável pelo movimento
        .Left = Horizontal_1
        .Top = Vertical_1

        If HandCenterY = 99 Then
          .TextEffect.Text = ArmLength
          If ArmLength <= 0 Then
            .Fill.ForeColor.SchemeColor = ppBackground
            .Line.ForeColor.SchemeColor = ppBackground
            .Left = 0
            .Top = 0
          End If
          If ArmLength > 0 Then
            .Fill.ForeColor.SchemeColor = ppForeground
            .Line.ForeColor.SchemeColor = ppForeground
          End If
        End If
        .
        .
        .
```

Figura 29- Sub-rotina que altera propriedades dos objetos

5.3.4 FUNCIONAMENTO DO SIMULADOR DE ANIMAÇÃO

O fluxograma a seguir (Figura 30 e Figura 31) demonstra o funcionamento do simulador de animação do núcleo do sistema operacional.

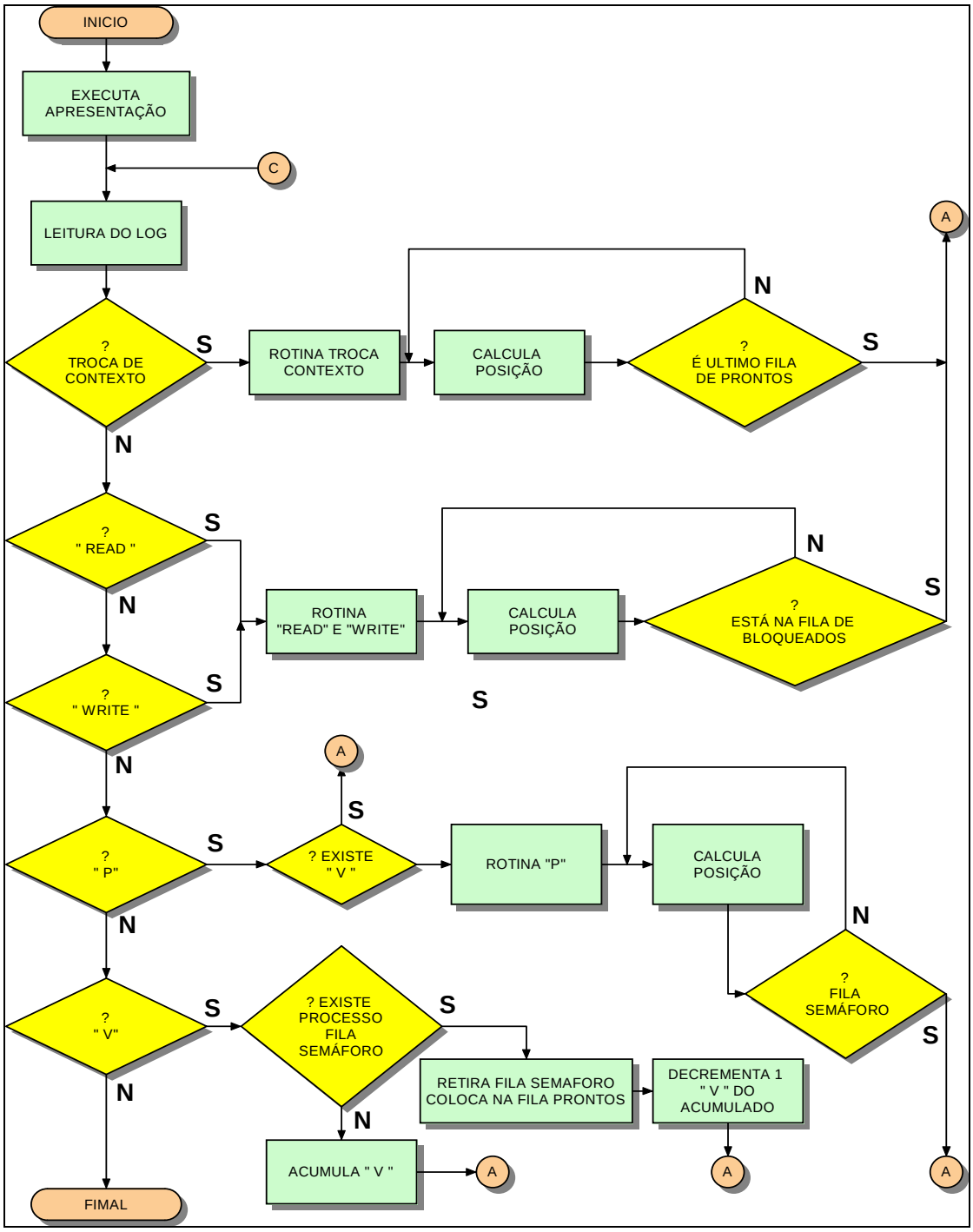


Figura 30 - Fluxograma do simulador

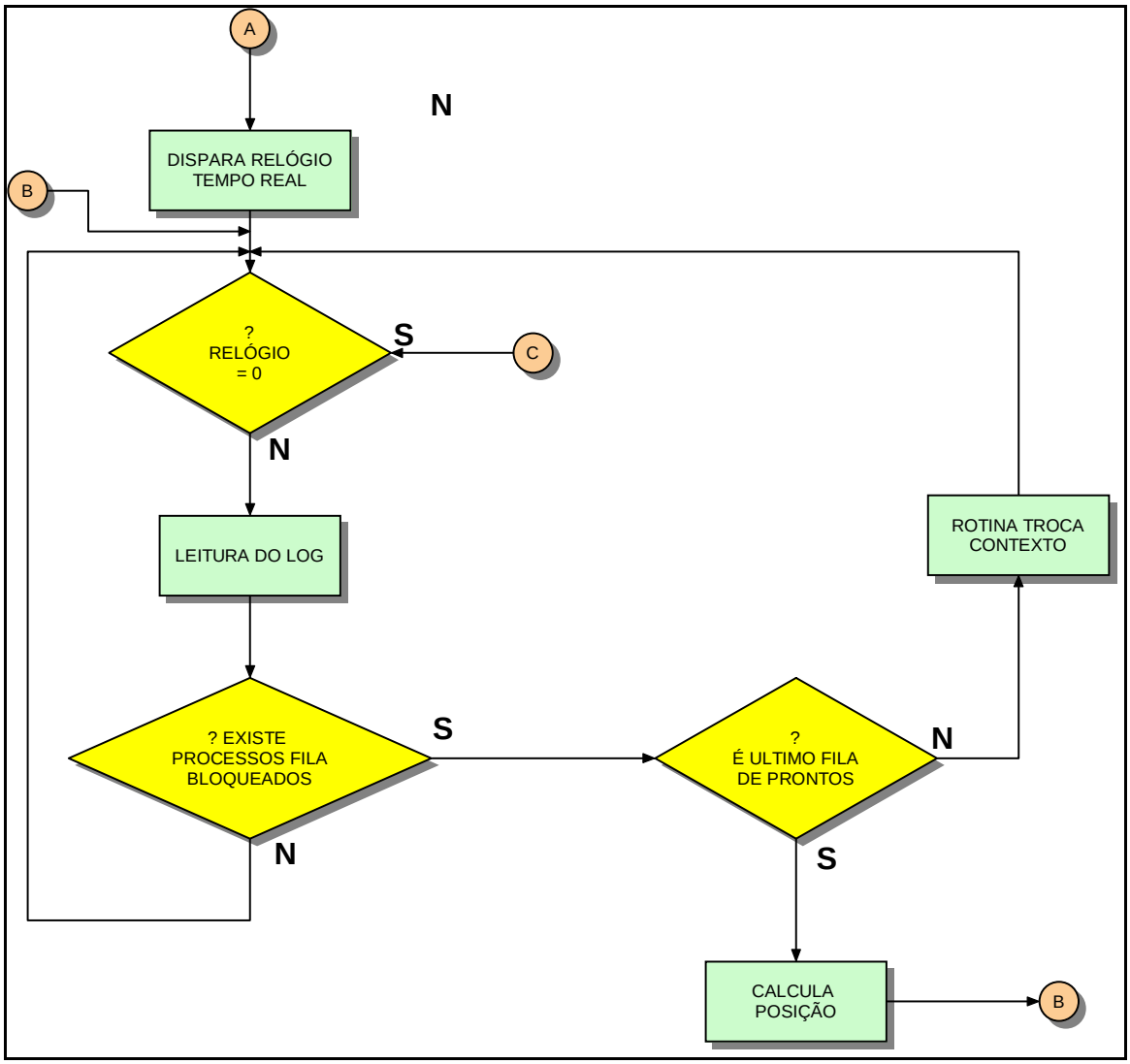


Figura 31 - Parte do fluxograma que representa a rotina de timer

Ao executar a apresentação *PowerPoint* o programa fará a leitura do *log* que contém as seqüências que devem ser seguidas no decorrer da animação. A leitura é feita linha a linha e um nova operação é executada a cada leitura.

Ao verificar o valor de cada linha do *log* se pode encontrar as operações seguintes: “troca de contexto”, “*read*”, “*write*”, “P”, “V” ou final de arquivo.

O primeiro teste que o programa realiza é a comparação da linha lida no *log*, com os possíveis estados pré-estabelecidos pelo programa. O primeiro a ser testado é a “troca de contexto”, se a resposta da comparação for falsa, deve-se seguir perguntando até achar o estado correto, mais se a resposta for verdadeira, a rotina segue e entra no procedimento das

rotinas de troca de contexto. A partir deste momento cálculos determinam a nova posição que cada objeto ocupará no cenário antes de começar a proceder o deslocamento, que levará o processo que pertencia a CPU par ocupar a última posição na fila de prontos. Após o processo que ocupava a CPU chegar ao final da fila de prontos e um novo processo assumir a CPU, o relógio de tempo real é disparado.

O programa verifica se o valor do *timer* de tempo real é zero. Se a resposta for verdadeira então o programa volta ao início e uma nova linha do *log* será lida. Se a resposta a comparação do relógio de tempo real for diferente de zero, o contador do relógio é decrementado em uma unidade de tempo, e passa-se a verificar se existem *timers* na fila de bloqueados. Caso não haja nenhum processo na fila de bloqueados torna-se a perguntar se o relógio de tempo real é igual a zero.

Para os casos onde existem processos na fila de bloqueados após decrementar o valor do relógio de tempo real, o programa verifica o valor do relógio referente ao processo que está bloqueado. Se este for maior que zero, decrementa o contador do descritor do processo na fila de bloqueados e volta a verificar o valor do relógio de tempo real. Se o valor do *timer* que está na fila de bloqueados chegar a zero, o processo volta para a ultima posição na fila de prontos, e a rotina volta a verificar o *timer* de tempo real para verificar se já chegou a zero.

Um processo passa a ocupar a fila de bloqueados apenas quando as operações são pertinentes a "*read*" ou "*write*", ou seja, o programa inicia, lê o *log* e encontra uma operação de "*read*", logo após executa as rotinas que são responsáveis em determinar quais serão as novas posições dos processos e começa a cálcular a nova posição até que o processo seja deslizado a fila de bloqueados, a rotina de cálculo de novo posicionamento é executada até que esteja em sua nova posição.

Caso a leitura do *log* referencia a uma operação "*write*", o procedimento será o mesmo do referenciado a operação "*read*", com uma pequena diferença entre os valores de timer.

Para um processo ocupar a fila de semáforos é necessário que a leitura do *log* identifique uma operação “P”, neste caso existem uma verificação para garantir que não exista uma operação “V” pendente, o que faria com que ao invés de ir para a fila de semáforos fosse para o final da fila de prontos, passaria a executar as rotinas de timer e todas as suas validações e voltaria a ler nova linha no *log*.

Caso o procedimento encontre um “V” e não existam processos na fila de semáforos, o procedimento que acumula “V” é acionado, e será utilizado assim que uma operação “P”, seja submetida liberando, passando o processo para o final da fila de prontos e liberando a operação que havia ficado armazenada. Novamente as rotinas de timer são acionadas e após suas validações novas consultas ao *log* são submetidas até que atinja o fim do documento.

6 CONCLUSÕES

A integração entre uma aplicação *Delphi* e os recursos do *PowerPoint Object Model* através do mecanismo provido pelo recurso *Office Automation* da *Microsoft* possibilitou uma dinâmica superior àquela obtida com a tradicional apresentação de slides.

Utilizando-se de um ambiente de animação, para compreensão dos conceitos de Sistema Operacional, desenvolveu-se uma ferramenta onde é possível demonstrar de forma dinâmica o comportamento dos componentes do núcleo de um sistema operacional hipotético. Desenvolvido como um sistema híbrido de ensino/aprendizagem, permite a compreensão das teorias apresentadas em sala passando ao aluno um papel ativo essencial mostra-se bastante eficiente, pois trabalha com conceitos abstratos deixando o aluno com um papel mais ativo e essencial em todo o decorrer do aprendizado. A partir da visualização do comportamento anômalo da animação o ambiente pode contribuir para que o aluno depure a sua versão do simulador de processos concorrentes.

6.1 EXTENSÕES

Certamente o término do TCC que gera o *log* contribuirá sobremaneira para ampliar o escopo desta ferramenta. Sugerem-se como extensões o incremento no realismo da animação através do desenvolvimento das seguintes funcionalidades:

a) identificação do valor dos semáforos quando usando primitivas P e V;

b) identificação dos periféricos acessados;

c) incorporação da simulação de acesso a mecanismos de memória virtual e área de *swap*;

d) incorporação de diferentes algoritmos de escalonamento.

REFERÊNCIAS BIBLIOGRÁFICAS

ANIDO, R. Uma proposta de plano pedagógico para a matéria sistemas operacionais. In: CURSO DE QUALIDADE, WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 2., 2000, Curitiba. **Anais…** Curitiba: UFPR, 2000. p. 125–148.

DOWNEY,A.B. Teaching experimental design in an operating system class. Proceedings of the 30th ACM SIGCSE, 1999

JONES, D.; NEWMAN, A. **A constructivist-based tool for operating system with animations**. [S.I.], 2002. Disponível em : <http/cq-pan.cqu.edu.au/david-jones/publications/papers_and_Books/RCOS.java_2002/>. Acesso em: 23 nov. 2004.

MACHADO, F. B.; MAIA, L. P.. Um framework construtivista no aprendizado de sistemas operacionais- uma proposta pedagógica com o uso do simulador SOSim. In: XII WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO. 1., 2004, Salvador. **Anais...** Salvador: XXIV Congresso da Sociedade Brasileira de Computação (SBC), 2004.

MAIA, L.P. **SOsim: Simulador para o ensino de sistemas operacionais:** este documento apresenta as motivações do trabalho, os fundamentos educacionais, o modelo proposto, a arquitetura e implementação, e conclusões. 2001. Tese de mestrado, NCE/UFRJ, Rio de Janeiro.

MATTOS, Mauro Marcelo. **Sistemas operacionais**, 2004. Notas de aula (Disciplina de Sistemas Operacionais, Bacharelado em Sistemas de Informação). Centro de Exatas e Naturais , Departamento de Sistemas e Computação, Universidade Regional de Blumenau.

MAY, Andrew. **Creating animation sequences in PowerPoint 2002 and PowerPoint 2003**. [S.l],2004. Disponível em: <http://msdn.microsoft.com/library/en-us/odc_pp2003_ta/html/odc_PP_TimeLine_Pt1.asp?frame=true>. Acesso em: 29 nov. 2004.

MAY, Andrew. **Animating shapes in PowerPoint 2000 and PowerPoint 97**. [S.l], 2004. Disponível em: <http://msdn.microsoft.com/library/en-us/dnpower2k2/html/odc_PP_AnimationSettings2000.asp?frame=true>. Acesso em: 29 nov 2004.

MAZIERO, C. A..Reflexões sobre o ensino prático de Sistemas Operacionais. In: X WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO, 1., 2002, Florianópolis. **Anais...** Florianópolis: UFSC, 2002.

MORSELLI JUNIOR, J. C. M. Um estudo sobre a utilização de procedimentos de ensino na disciplina de sistemas operacionais In: II WEIMIG: WORKSHOP EM EDUCAÇÃO EM COMPUTAÇÃO E INFORMATICA, 1., 2003, Poços de Caldas. **Anais...** Poços de Caldas: PUC Minas, 2003.

OLIVEIRA, Rômulo de. **Sistemas operacionais**: série livros didáticos número 11. Porto Alegre: Ed. Sagra Luzzatto , 2001.

PEREZ-DAVILLA, A. OS – bridge between academia and reality. ***ACM SIGCSE Bulletin***, 27(1), p. 146–148, 1995.

RICE, Frank **Microsoft Powerpoint**. São Paulo, [2005]. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dno2k3ta/html/OfficeAccess2Powerpoint.asp>. Acesso em: 05 jun. 2005.

SILBERSCHATZ, Galvin Gagne. **Sistemas operacionais**: conceitos e aplicações. Rio de Janeiro: Ed. Campus Ltda , 2000.

TANENBAUM, Andrew S.; WOODHULL, Albert S**. Sistemas operacionais:** projeto e implementação. Porto Alegre: Artes Médicas Sul Ltda, 2002.

UNONIUS, Lars Gustav Erik. Microsoft Powerpoint 97. In**: Sem Mistério**. São Paulo: Berkeley, 1997

VIEIRA, Enio Robert. **Introdução a sistemas operacionais**, 1975. Seminário Introdução a sistemas operacionais. Faculdade de Ciências Econômicas de Blumenau. Departamento de Informática, Universidade Regional de Blumenau.