Aula de Projeto

Detalhamento da Implementação de um Simulador
(de Sistema Operacional), dirigido por eventos

(Baseado no artigo McDougall-70)

Material de apoio ao projeto

Como material de apoio, é recomendado o seguinte tutorial clássico sobre o assunto, no qual se baseia esta apresentação.

McDougall, M. H.

Computer System Simulation - an Introduction

ACM Computing Surveys, vol. 2 n. 3, setembro de 1970, p. 191-209

Esta apresentação

Esta apresentação é constituída de duas partes.

- A primeira delas mostra o conteúdo básico do artigo McDougall-70, recordando os principais conceitos e a técnica de simulação guiada por eventos.
- A segunda preocupa-se mais com a descrição e com a análise dos detalhes da simulação particular a ser desenvolvida neste projeto

PRIMEIRA PARTE – o artigo

Simulador estocástico de processamento multiprogramado, dirigido por eventos

- Usando como base o artigo do McDougall, construir um simulador estocástico dirigido por eventos
- Construir uma lista inicial adequada de eventos independentes, comprovar o funcionamento correto do simulador exercitando-o, com base nessa lista, de modo que, ao menos uma vez, sejam executadas corretamente, por completo, cada uma das suas funções internas.
- Este programa não é o objetivo final, mas constitui o arcabouço do trabalho propriamente dito, discutido adiante, na 2ª parte desta apresentação.

Entradas do simulador

- o **instante inicial de acionamento do relógio** de tempo real,
- o **instante final de simulação** e
- os **instantes de chegada** de cada um dos elementos de um conjunto de programas independentes, para cada qual esteja especificado:
  - o **tempo total estimado** de processamento,
  - a **quantidade estimada de memória** de que necessitará
  - o **número estimado de operações de entrada/saída** que deverá realizar.

Saídas esperadas

Como saída, imprima um **log de simulação**, na forma de uma lista classificando todos os **instantes previstos e executados** para **todos os eventos que ocorrerem na simulação** de cada um dos programas.

Em cada linha de impressão imprima as seguintes informações:

- **<instante>** é o valor corrente do relógio do simulador (**instante corrente de simulação**)
- **<tipo de evento>** indica qual foi a **ocorrência** observada neste instante de simulação
- **<identificação do programa>** indica **qual dos programas** simulados provocou o evento
- **<ação executada>** indica **qual rotina foi executada** como reação do sistema à ocorrência do evento
- **<resultado>** indica os **efeitos dessa reação** sobre a situação corrente do programa simulado

Execute esse procedimento para várias listas iniciais.

Analise, compare e comente os resultados.

Recomendações

- Desenvolva o programa construindo primeiramente um **núcleo básico**, e vá adicionando as diversas funcionalidades, e testando-as **uma por vez**, para evitar dificuldades desnecessárias.
- Vá **documentando** o projeto à medida que for construindo o programa, para evitar omissões de coisas importantes no relatório devido a inevitáveis esquecimentos.
- Use uma **linguagem familiar de programação**, para evitar atrasos no projeto.

Componentes do sistema, simulados no software descrito no artigo do McDougall

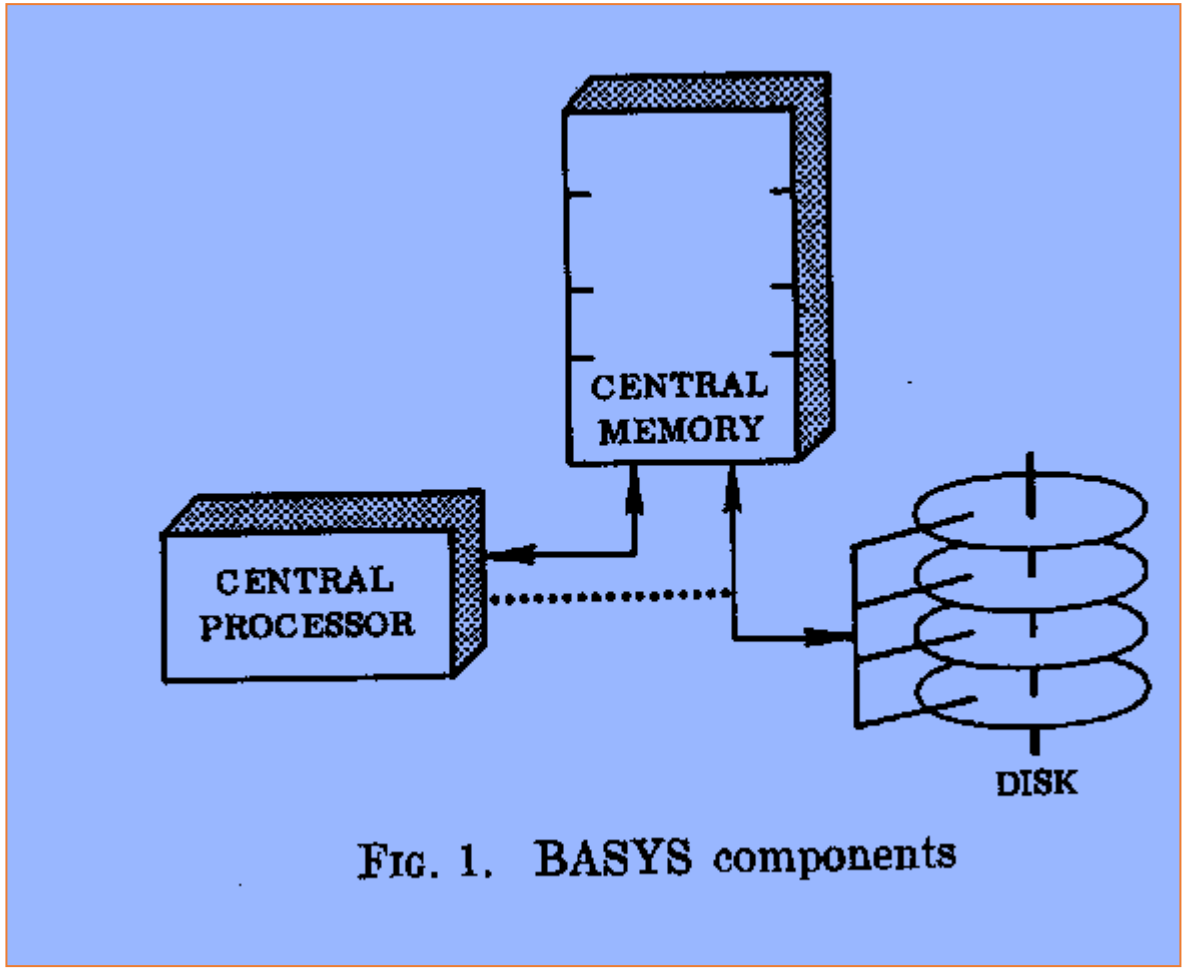
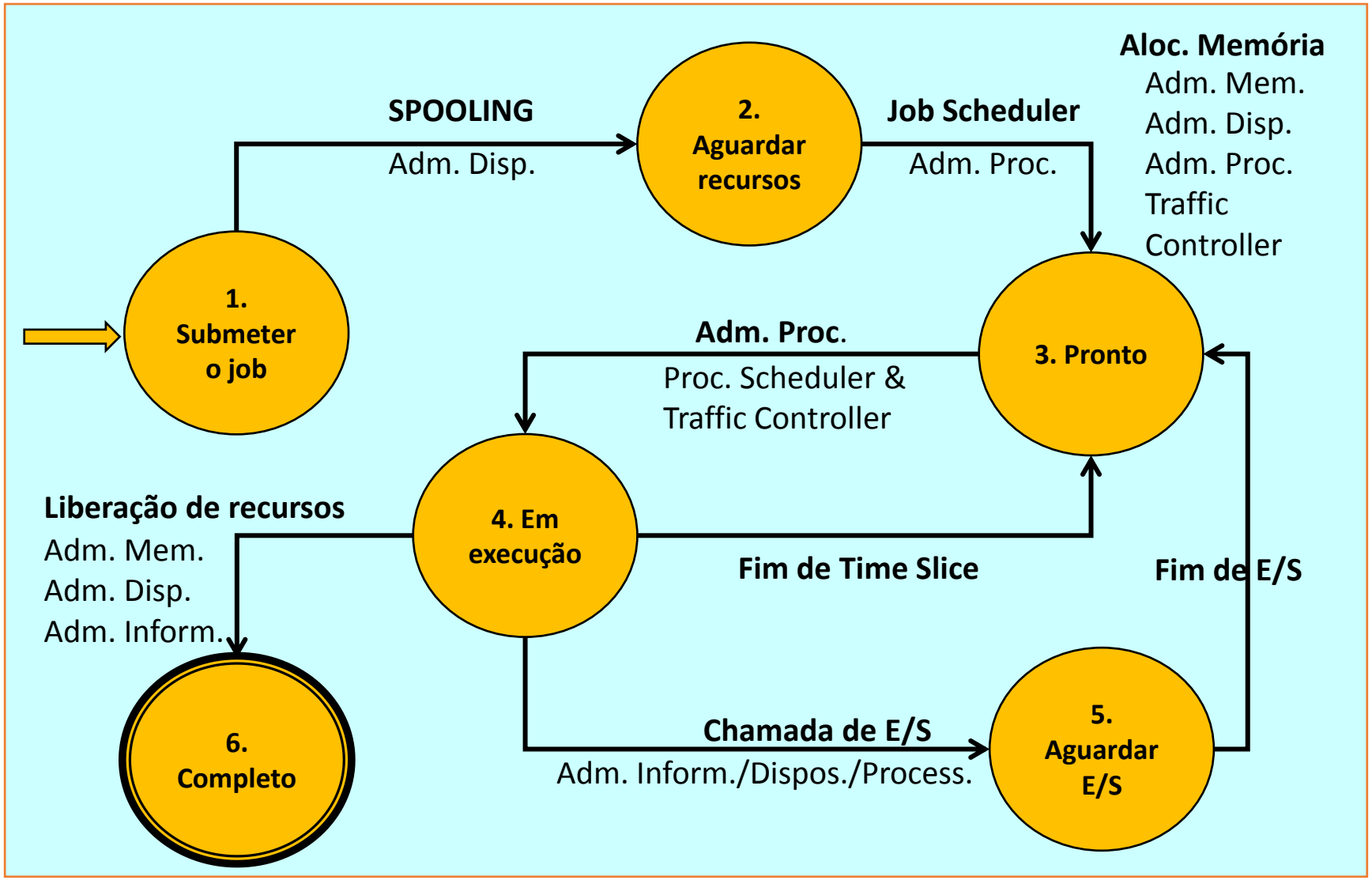


FIG. 1. BASYS components

Configuração a simular

- Adotar para a simulação uma configuração de sistema que apresente, no mínimo, os seguintes componentes:
  - **Memória** principal (**segmentada**)
  - **Processador** (um)
  - **Disco** (um) com **sistema de arquivos simples**
  - **Leitoras** físicas (**duas**)
  - **Impressoras** físicas (**duas**)

Modelo de estados



Modelo de execução dos jobs no artigo (blocos representam estados)

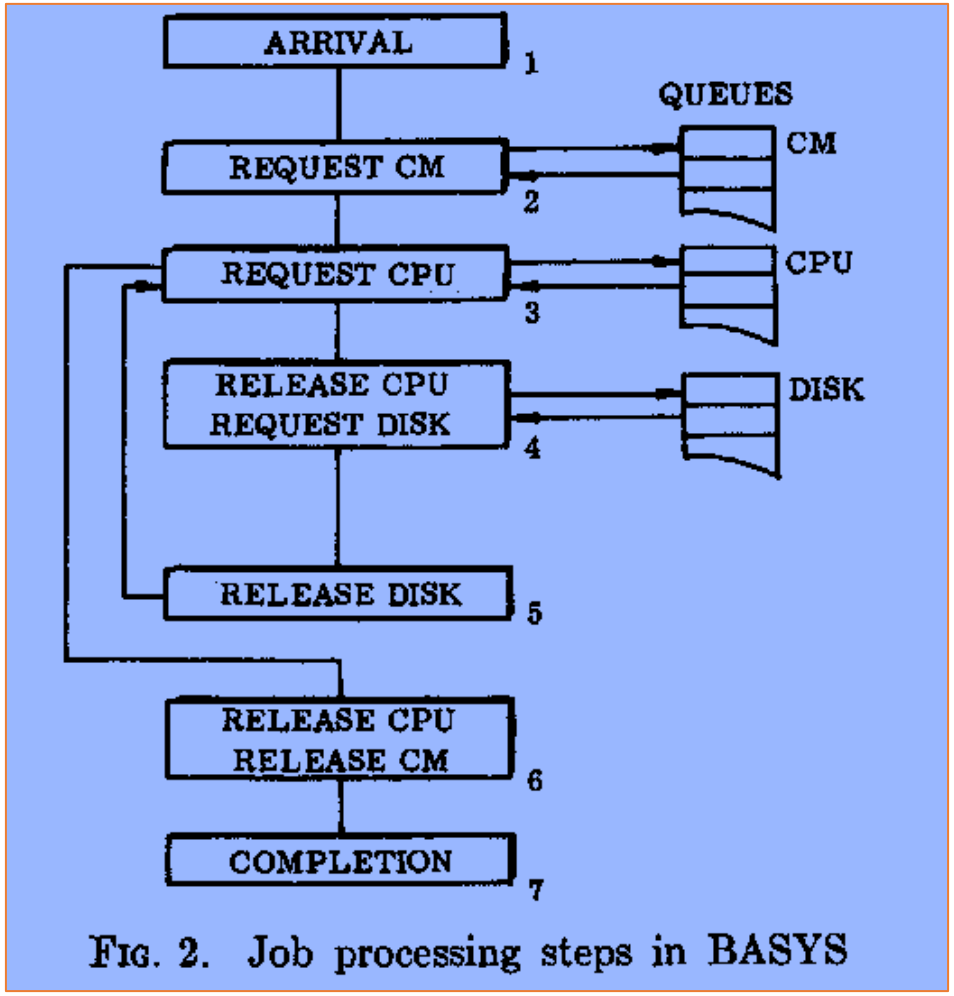


FIG. 2. Job processing steps in BASYS

Modelo de execução a adotar

- Antes de tudo, **identificar as partes** do modelo estudado em aula **com as do artigo**
- Naturalmente, é preciso adaptar o diagrama para comportar **novos estados**, referentes ao gerenciamento dos recursos incorporados
- É preciso ainda escolher critérios e **incorporar a lógica** correspondente, para implementar as **políticas de gerenciamento** que forem adotadas.

Distribuição dos jobs de acordo com as necessidades de memória principal

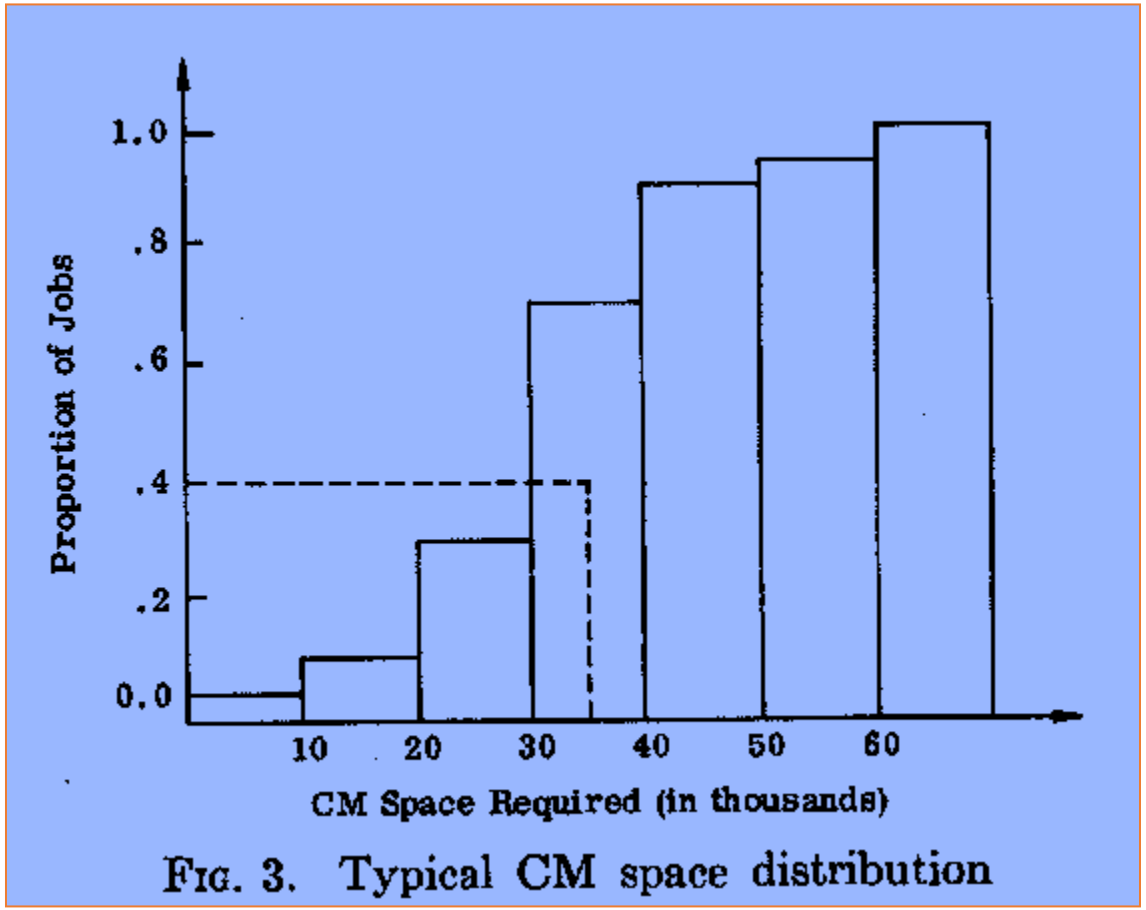


FIG. 3. Typical CM space distribution

Observações

- Convém criar **modelos** como o do gráfico anterior **para representar o comportamento** dos **jobs**, e do uso dos **periféricos**: disco, leitoras e impressoras.
- Para que os resultados de simulação se apresentem um tanto realísticos, escolher parâmetros descritivos dos periféricos que **imitem os atributos** usuais de periféricos **reais**.
- **Não adotar hipóteses** para a simulação que sejam **incompatíveis com a realidade**, pois os resultados de simulação tenderão a ser incoerentes.

Estrutura de dados para a implementação da lista de eventos

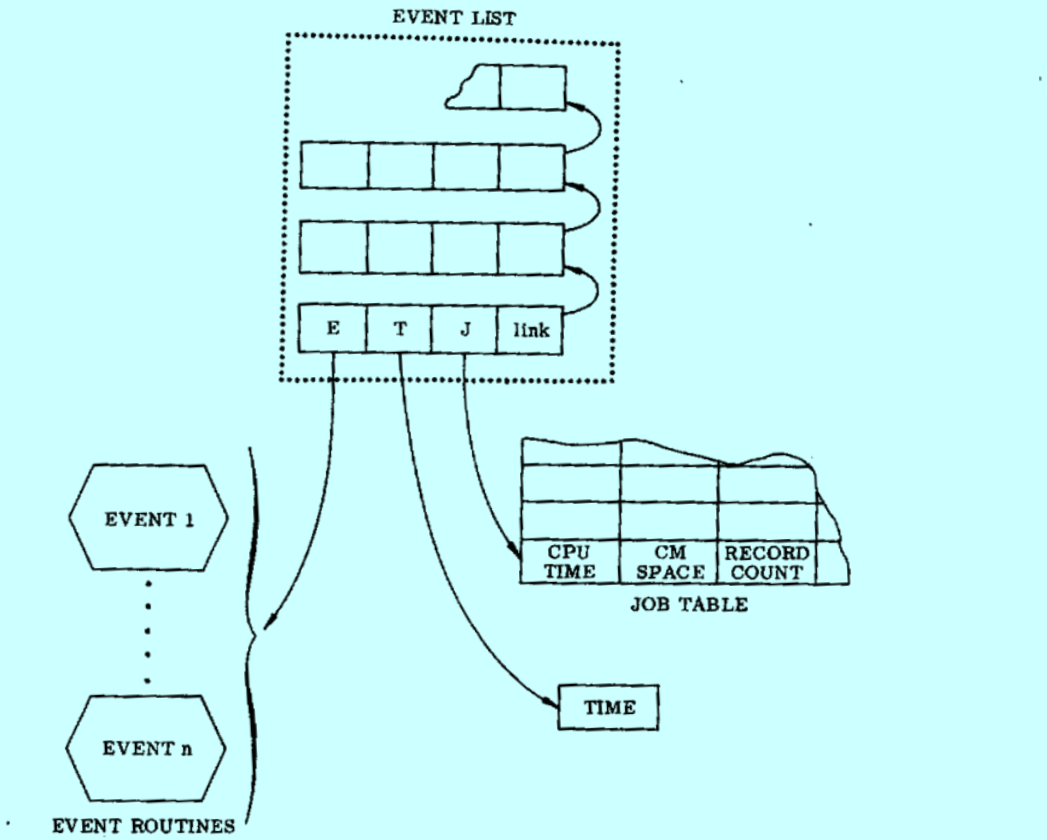


FIG. 5. Event list structure: E, event identifier; T, event time; J, job table entry pointer

Eventos e seu tratamento

- Em sua quase totalidade, os **eventos** neste projeto simulam **interrupções** na máquina, logo há interrupções de vários tipos e com diversas origens, assim como os eventos que os representam.
- Para cada uma dessas interrupções/eventos deve haver uma **rotina de tratamento** correspondente.
- O instante da ocorrência das interrupções na prática é aleatório, mas no simulador deve ser programado, por meio do estabelecimento do **instante previsto de simulação em que o evento deve ocorrer**.

Exemplo de estrutura de dados para a implementação de uma fila

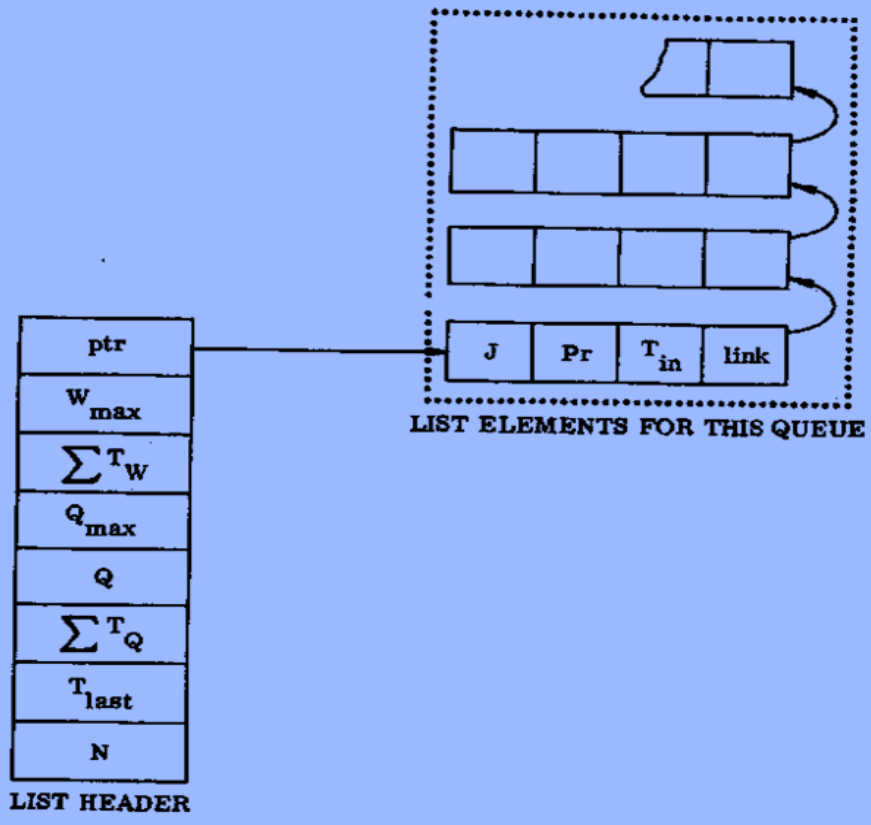


FIG. 6. Queue structure example:

ptr	pointer to head of list
W_{max}	maximum waiting time
ΣT_W	waiting time accumulator
Q_{max}	maximum queue length
Q	current queue length
ΣT_Q	length · time product accumulator
T_{last}	time of last entry/removal
J	job table entry pointer
Pr	priority or other ranking attribute
T_{in}	time of entry in queue
N	entry count

A mesma figura, ampliada

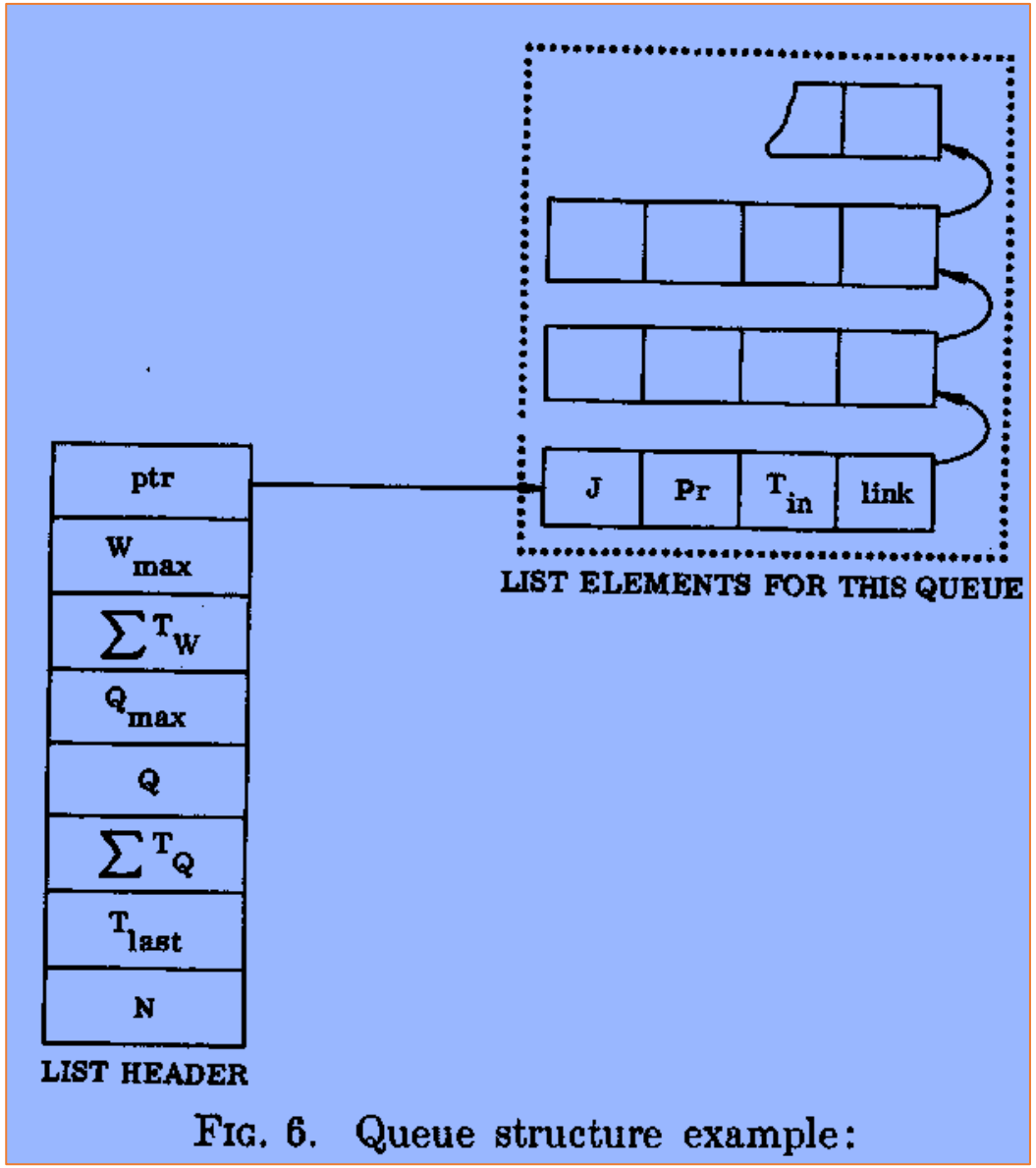


FIG. 6. Queue structure example:

Filas a serem utilizadas

- Cada **estado** representa um **recurso**, com uma **fila de espera** associada, para indicar processos que aguardam autorização para utilizar tal recurso.
- Esses processos devem permanecer em uma fila, **ordenada pelo instante de chegada**, e também por **prioridade** (processos com mesma prioridade ficam ordenados pelo seu instante de chegada).
- **Para cada processo**, um descritor memoriza os **parâmetros correspondentes**, calculados ou coletados durante a simulação.

Motor de eventos da...

... lógica do simulador

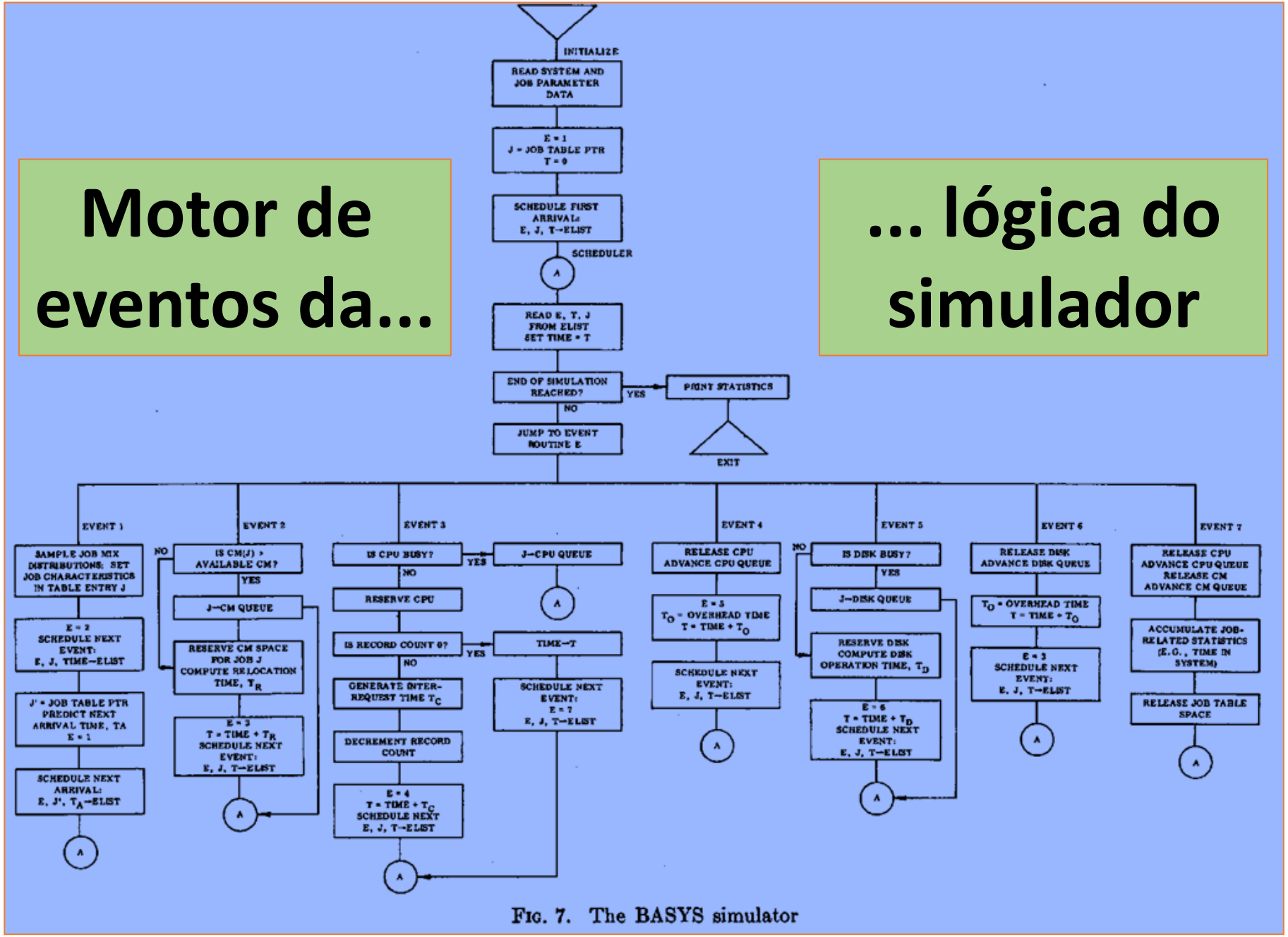


FIG. 7. The BASYS simulator

Lógica

- O diagrama de blocos é auto-explicativo.
- Basta identificar os tratamentos que estejam discrepantes das decisões que foram adotadas para este projeto e modificá-las para que fiquem adequadas às mesmas.
- Novos tratamentos de eventos podem ser incorporados conforme a necessidade. Em seu projeto, incorpore ao menos três ou quatro novos eventos que não constam no artigo e seu tratamento.
- Para facilitar a operação do simulador, novos eventos artificiais podem ser acrescentados
  (por exemplo, para ligar/desligar listagem de logs)

Exemplo de funcionamento da alocação de processador, com política de prioridade

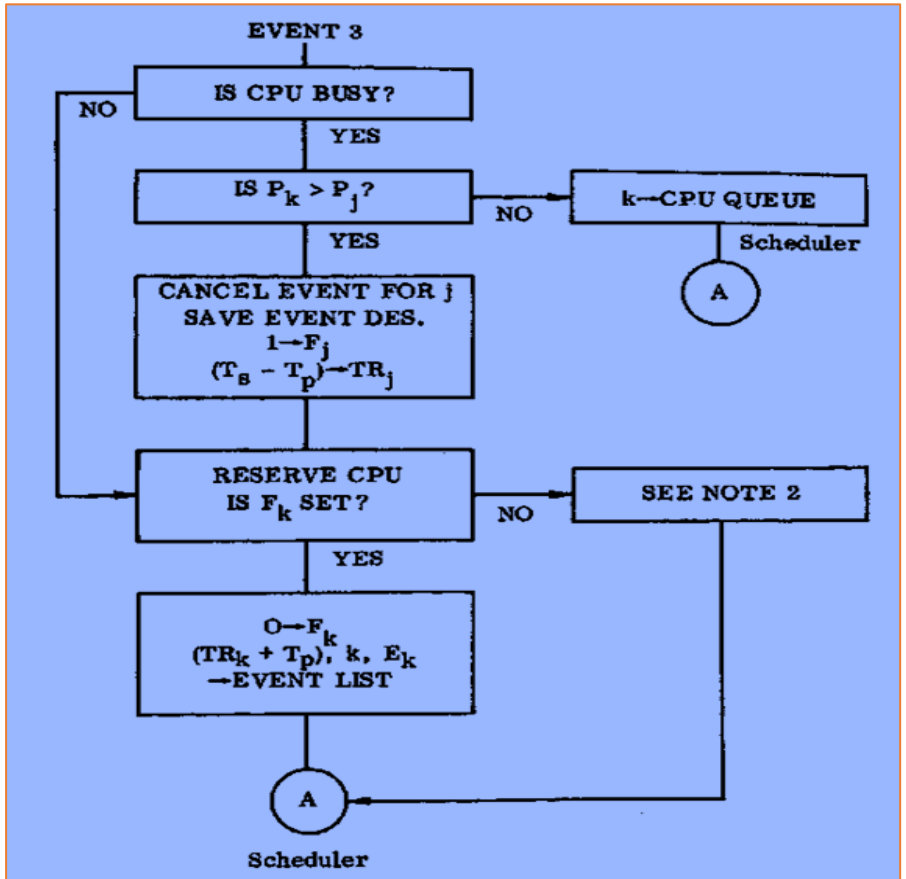


FIG. 8. Example of CPU assignment with priority interrupt

Note 1.
j pointer for job in execution
k pointer for job preempting CPU
P_i priority of job i
T_s next event time for job j
T_p current time
F_i interrupt flag for job i

Note 2. The processing performed here is essentially that discussed in the event 3 routine description.

A mesma figura, ampliada

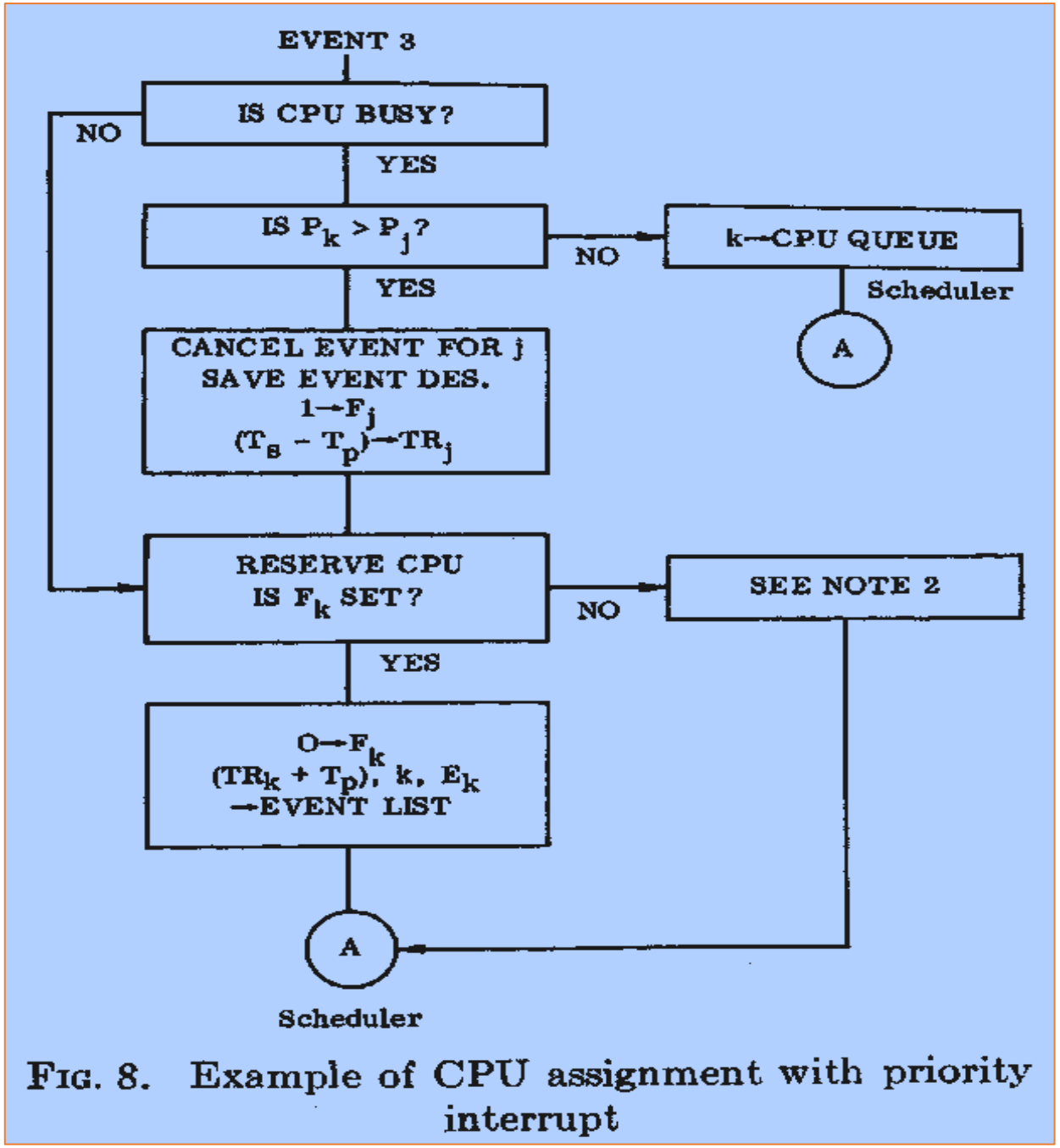


FIG. 8. Example of CPU assignment with priority interrupt

Prioridades

- A forma apresentada na figura anterior é uma das possíveis opções para implementação da política de prioridade, mas naturalmente não é a única.
- Caso se deseje adotar outra forma de implementação, basta substituir a lógica proposta.

FIM da Primeira Parte

SEGUNDA PARTE – o projeto

- Detalham-se a seguir os objetivos do projeto.
- O simulador dirigido por eventos do artigo é do tipo estocástico, com granularidade no nível de sistema.
- É portanto um simulador no qual o modelo adotado para representar os fenômenos simulados não leva em consideração seus detalhes individuais, mas apenas o seu efeito coletivo.

- Não há preocupação da modelagem quanto à relação entre um programa e as operações por ele desencadeadas.
- Por exemplo, a modelagem adotada não considera os acessos individuais do programa ao disco, pois o foco preferencial dessa simulação é observar a carga coletivamente imposta ao sistema pelo conjunto dos programas ativos.
- Além deste, há diversos outros casos similares, descritos no artigo.

- Pretende-se agora refinar a modelagem, pela incorporação de novos detalhes, para que a simulação permita observar, além das estatísticas gerais, em nível de sistema, também algumas métricas relativas aos jobs individuais.

- Os requisitos para o gerenciamento de recursos estão sugeridos a seguir, e estão colocados a título de sugestões gerais.
- Se for conveniente, complete ou altere levemente as especificações propostas.
- Sempre que fizer alterações, descreva-as no relatório, e justifique-as, explicando as razões e as vantagens da opção escolhida em relação à originalmente apresentada.

Gerenciador de memória: memória virtual **segmentada**, sem paginação

- **Cada recurso** gerenciado pelo sistema deve ter **sua correspondente fila de espera**:
  - ***Memória***
  - ***Multiprogramação*** (com **limite pré-fixado de programas** simultaneamente ativos no sistema)
  - ***CPU*** (lista de **processos prontos** para serem executados)
  - ***Entrada e Saída***: uma fila para cada **dispositivo** e **serviço**
  - ***Disco***: Uma fila única para o **disco físico**
  - ***Arquivos***: Diversas filas, cada uma para controlar o acesso a um **arquivo** específico **em uso**

Descrição do job

- Instante de **chegada**,
- **Estrutura** do job (**árvore dos segmentos** que o constituem, e tamanho de cada segmento)
- Número **total** previsto de **entradas**, **saídas** e **acessos a arquivos** para cada segmento
- Identificação dos **acessos a serem feitos** pelo job nos arquivos
- **Tempo máximo previsto de uso da CPU**

Gerenciador de **processos**

- Multiprogramação por ***time-slice***
- **Grau limitado** (ajustável pelo sistema) **de multiprogramação**;
- **Fila** cíclica de **processos prontos** para a execução na CPU (***ready list***);
- **Fila de CPU** em regime cíclico **(*round robin*)**

Gerenciador de **informações**

- **Sistema simples de arquivos** linear único, visível por todos os programas no sistema.
- Um esquema de **proteção de acesso** aos arquivos:
  - Identificação do programa **proprietário do arquivo**
  - **Nome** e **tamanho** do arquivo
  - Arquivos **públicos** de acesso geral, e **particulares**, acessíveis apenas ao proprietário

Premissas

- Durante a execução do programa, é **igualmente provável** que um segmento referencie qualquer dos segmentos de que depende, ou seja, os segmentos abaixo dele na árvore de dependências.

- O **intervalo entre dois pedidos sucessivos de entrada/saída** por um segmento pode ser inicialmente **constante**, e estimado a partir do número de entradas/saídas previstas para cada segmento (representando um tempo médio de uso da CPU pelo segmento entre duas operações de entrada/saída)

Instantes dos pedidos de entrada/saída

- Depois de testado com intervalos constantes, quando da chegada do job ao sistema, o tratamento desse evento passa a estabelecer, **aleatoriamente, com distribuição uniforme**, os instantes relativos dos pedidos de entrada/saída ao sistema (isso é mais realista, e nesse caso tais momentos devem ser registrados em listas associadas aos programas, para serem usados pelo simulador).

Instantes de referência aos segmentos

- Os instantes de ocorrência de **referências aos segmentos de memória** podem ser estimados determinando-se, no momento do recebimento da CPU, se no *time-slice* corrente haverá ou não referência a algum outro segmento.
- É possível tratar as referências à memória de forma análoga à das entradas/saídas.

Ordem de execução dos segmentos

- Se estiver sendo executado um segmento S_i que tem como segmentos possivelmente referenciados S_k ... S_m , ao receber a CPU, o processamento possível será:
  - **Processamento interno em S_i apenas, durante todo o time-slice**. Neste caso, apenas controlar o transcurso do tempo de processamento do job.
  - **Referência a algum dos S_k ... S_m antes do término do time-slice.** Neste caso, contabilizar a parcela já transcorrida do time-slice, e iniciar a execução do segmento referenciado (caso esteja presente na memória, o programa não perde a CPU, mas prossegue no segmento referenciado, caso contrário, inclui-se, no início da lista de eventos, no momento corrente de simulação, um evento de referência a um segmento que não está presente na memória – correspondente a uma **interrupção de falta de segmento**)

Modelagem da memória segmentada

- A **modelagem da memória segmentada** pode ser feita nos mesmos moldes da memória particionada, com partições de tamanho arbitrário (as partições a serem utilizadas para os segmentos dos programas devem ter suas dimensões estabelecidas no momento da chegada do job, extraídas da descrição de sua estrutura).

Garbage Collection

- **Não é obrigatório compactar a memória**.
- Em caso de **fragmentação**, bloquear o aumento do número de processos ativos até que seja liberada uma quantidade suficiente de memória contígua.
- No entanto, se isso for considerado conveniente, há liberdade para incluir uma rotina de ***Garbage Collection***.

Sistema de Arquivos

- A modelagem do **sistema de arquivos** poderá ser feita aplicando-se à área em disco os mesmos algoritmos aplicados às partições de memória (uma partição de disco para cada arquivo, e em caso de fragmentação do disco, este é considerado saturado, a menos que seja adicionada **opcionalmente** uma rotina de compactação).

No simulador final deverão estar presentes, portanto, ao menos os seguintes elementos:

- **Tabela de jobs**, uma para todo o sistema.
  As linhas desta tabela contêm informações sobre os diversos programas em atividade:
  - quanto **tempo de processamento** falta para o programa terminar
  - lista dos pontos no processamento em que há **requisições de entrada/saída**
  - ponteiro para a **árvore de segmentos** que descreve a estrutura do programa, bem como as dimensões dos seus segmentos
  - conjunto de **arquivos a serem referenciados** pelo programa, e suas dimensões no disco.

- **Tabelas de partições livres da memória** física e de **partições livres do disco**, com posição e comprimento.
- **Tabela de partições ocupadas da memória** (uma para cada programa), correspondente à *segment map table* do programa – tabela cujo conteúdo varia durante o processamento, em função de quais segmentos estão presentes na memória física

- **Tabela de partições ocupadas no disco** (única), correspondente ao conjunto de arquivos que formam o sistema de arquivos implementado. Cada linha desta tabela indica o **nome**, o **possuidor**, o **tamanho** e as **características do arquivo** a que se refere.

- **Eventos independentes: chegada dos jobs** (identificação, instante de chegada, recursos utilizados: tempo de CPU, número de operações de entrada/saída, quantidade de memória ocupada pelos diversos segmentos).

FIM