NÚCLEO DE UM SISTEMA OPERACIONAL MULTIPROGRAMADO (CARCARA')

2

Jose' Lavaquial Breitinger

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M. Sc.)

Aprovada por:

Sueli Mendes dos Santos
(Presidente)

Michael Stanton

Paulo M. Bianchi França

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 1983

BREITINGER , JOSE' LAVAQUIAL

Núcleo de um Sistema Operacional Multiprogramado (Rio de Janeiro) 1983

IX, 95p. 29,7cm (COPPE-UFRJ, M.Sc , Engenharia de Sistemas , 1983)

Tese - Univ. Fed. Rio de Janeiro, Fac. de Engenharia.

1. Computacao I. COPPE/UFRJ II. Titulo (série)

'A CARMEN LUCIA, ANA ELISA e ANDRE'

Agradeço a todos que tornaram possível este trabalho, em especial à Carmen Lúcia R. Breitinger, Suely Mendes dos Santos, Ricardo Dias Campos e Sandra Maria S. Carvalho.

SINOPSE

E' apresentado um Sistema Operacional Multiprogramado para microcomputadores, desenvolvido neste trabalho.

Em linhas gerais, os recursos apresentados pelo sistema são:

- Multiprogramação, ate' 62 processos podem estar executando concorrentemente.
- Memória Virtual, cada processo pode ter ate' 1 Mb de espaco virtual.
- E/S independente de periférico, suportadas pelo NÚCLEO.
- Mecanismo de sincronismo e troca de mensagens entre processos.
- Suporte a redes, embutido no NÚCLEO.

Acompanham o trabalho, no capitulo Conclusões, medidas de performance feitas no protótipo do sistema e sua comparação com outros sistemas . Também aí' faz-se uma descrição do histórico e experiências/dificuldades encontradas em cada uma das fases do desenvolvimento do trabalho.

Um novo sistema de preempção e escalonamento de processos via uma estrutura de pilhas e' apresentado. A troca de mensagens e sincronismo entre processos foram implementadas a partir de um novo conceito de primitivas com esta finalidade.

A gerência de memória apresenta características únicas, pois permite programas com espaços virtuais independentes ate' 1 Mbyte, em microprocessadores com endereçamento somente ate' 64K bytes.

ABSTRACT

This work presents a Multiprogrammed Operating System, used in microprocessors.

In summary, it provides the following features:

- Multiprogramming, with a maximum of 62 concurrent processes.
- Virtual Memory: each process is allowed up to 1 Mb of independent virtual space.
- Peripheral independent I/O, supported by the Nucleus.
- Syncronism and message exchange are also available.
- Network support is built-in in the Nucleus.

Performance measurements, obtained in the system's prototype, are compared with other systems. This work also describes the difficulties and experiences gained in the development of the system. These conclusions are presented in the chapter 'Conclusao'.

A new process scheduling method, using a stack structure, is presented. The message exchange and process synchronization were implemented trough new primitives.

Memory management has unique features, as it permits 1 Mbytes programs in microprocessors with 64K bytes of address space.

Í N D I C E

CAPITULO I

INTRODUÇÃO

Este trabalho e' a descrição geral da implementação de um sistema operacional multiprogramado. O NÚCLEO do sistema, objetivo da tese, e' apresentado em detalhe.

O sistema foi definido em paralelo 'à elaboração de uma arquitetura em HARDWARE para abrigá-lo [22]. Mesmo assim, o sistema e' razoavelmete genérico, o que permitiu implementar e testar aproximadamente 90% de seu código em outra máquina [14] [17].

O NÚCLEO e' escrito em assembler do microprocessador MC6809 [11] [12] e na versão atual ocupa aproximadamente 4K de código.

Devido ao cunho prático do trabalho, o sistema e' configurável 'às opções da máquina onde ira' residir. Estas opções consistem em dimensionar os tamanhos das tabelas, especificar os gerentes de perifericos e processos carregados automaticamente ao ligar a máquina. Na geração do sistema, o que é' feito via o programa CONFIGURADOR [20], são definidas estas opções.

I.1 TRABALHOS ASSOCIADOS

1. LINGUAGEM DE COMANDOS (SIRIUS) - é' apresentada na ref. [3]. Faz parte de seu escopo a interface do usuário com o sistema, a implementação dos comandos disponíveis ao usuário e o gerente de vídeo e teclado.

2. SISTEMA DE REDE - é' apresentado na ref. [1]. Implementa o módulo de troca de mensagens entre máquinas geográficamente próximas (parte LOCAL) ou distantes (parte REMOTA).

3. GERÊNCIA ARQUIVOS EM DISCO - é' apresentado na ref. [6]. Suporta arquivos com registros de tamanho fixo e variado, acesso sequencial e direto, pré leitura automática, segmentação física de arquivos.

4. COMPILADOR PASCAL - é' apresentado na ref. [7]. Implementação com ênfase na gerência de memória, de modo a permitir programas com tamanho igual ao máximo que o sistema operacional pode suportar.

I.2 ARQUITETURA DA MÁQUINA

É' uma máquina modular e razoavelmente configurável. Sua parte básica é' composta pelo processador MC6809, RAM (até' 512K), PROM (até' 16K), sistema de paginação e relocação de memória, vídeo texto e gráfico, teclado e duas linhas de comunicacao RS232-C [22].

O VÍDEO e' mapeado na memoria RAM principal. Cada linha na tela pode ser mapeada em qualquer endereço desta memória, independente das demais, via uma tabela de mapeamento.

Existem 4 modos de exibição:

1. TEXTO 80Hx30V caracteres, Preto e Branco ou ate' 256 cores por carácter.

2. GRÁFICO 640Hx240V pontos, Preto e Branco ou ate' 256 cores por ponto.

3. TEXTO 40Hx30V caracteres, Preto e Branco ou ate' 256 cores por carácter.

4. GRÁFICO 320Hx240V pontos, Preto e Branco ou ate' 256 cores por ponto.

As cores também se traduzem em níveis de cinza num monitor Preto e Branco. O tamanho das linhas de texto e gráfico exibíveis na tela é' controlado pelo software básico, (default = 128 p/ texto e 1024 p/ gráfico) que faz a tela correr como se fosse uma janela nas 4 direções.

A expansão é' feita via conectores externos genéricos, que permitem até' 6 placas serem ligadas. Estas placas normalmente são as interfaces com os periféricos, e atualmente existem as seguintes definidas:

1. DISKETTE - placa com processador próprio (MC6809) que controla até' 4 unidades de qualquer tipo. Aceita comandos de alto nível como abertura de arquivos, setores numerados logicamente (de 1 a n, a conversão p/ trilha/setor/cabeça é' feita internamente) . Também é' programável para executar tarefas específicas (busca em banco de dados, etc.). Torna transparente ao sistema o tipo de disco usado.

2. WINCHESTER - idêntico 'à interface diskette só' diferindo na parte de ligação física 'à unidade. Possui as mesmas características, inclusive sendo transparente ao sistema,se é' whinchester ou diskette

3. REDES - placa com processador próprio (MC6809), que controla rede local. Todo o protocolo, manutenção, recepção/transmissão de dados é' feito por seu intermédio. Somente necessita a intervenção do sistema, quando uma mensagem é' corretamente recebida para esta máquina. E' programavel.

Outras expansões são: memória RAM adicional, (mais 3Mbyte), fazendo um total de 3.5Mbytes e CO-PROCESSADOR. Este segundo processador tem acesso intercalado 'à memória adicional, não afetando a velocidade do processador principal nem a sua própria. O processador e' o MC68000.

CAPITULO II

O SISTEMA OPERACIONAL

O sistema operacional (S.O.) e' o conjunto de recursos providos para o gerenciamento, padronização e criação de recursos basicos oferecidos com a maquina. Sao vários os módulos que o compõe e como exemplo podemos citar o NÚCLEO, a LINGUAGEM de COMANDOS (SIRIUS), o sistema gerenciador de REDES, os controladores de PERIFÉRICOS, o CARREGADOR de PROGRAMAS, o EDITOR de TEXTOS, etc.

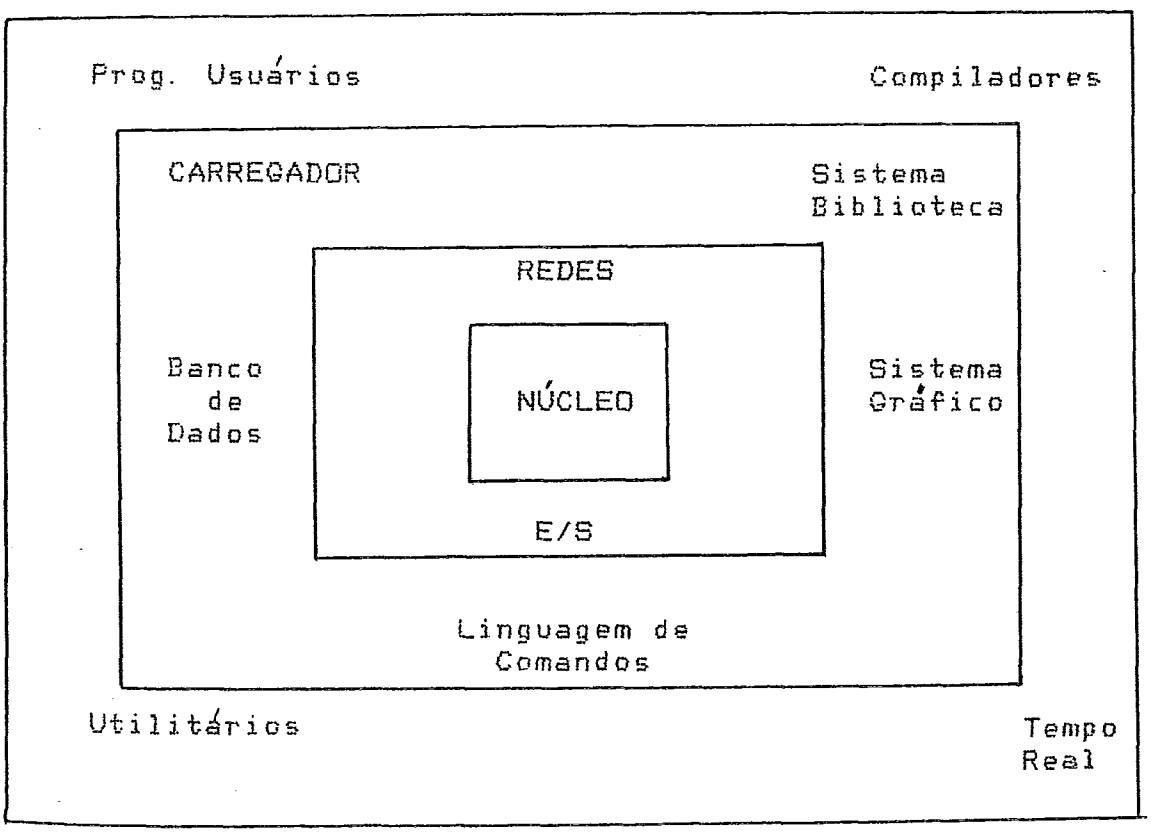

Fig. 2.1 - CAMADAS DO S.O.

Neste trabalho, nos capítulos que se seguem, serão apresentados os conceitos básicos do sistema operacional, isto e', como são implementados o sistema de E/S, de multiprogramação, de gerência de memória, etc. Em detalhe será' visto o NÚCLEO do sistema operacional, que e' a parte mais interna do software.

II.1 CARACTERISTICAS DO S.O.

II.1.1 MULTIPROGRAMAÇÃO

Cada processo em execução tem um contexto independente dos demais processos, executando naquele instante. Deste modo, não ha' interferência mútua em suas execuções.

O funcionamento geral do sistema está' fortemente baseado em multiprogramação, pois além dos programas dos usuários, uma série de tarefas do Sistema Operacional são implementadas como processos. Por este motivo, o NÚCLEO dá' suporte a um índice de multiprogramação razoável, indo a um máximo de 62 processos.

A distribuição do recurso UCP pelos vários processos é' feito pelo módulo ESCALONADOR, que usa uma técnica de PILHA. Esta técnica de PREEMPÇÃO e ESCALONAMENTO e' uma novidade apresentada neste trabalho, estando descrita em detalhes nos apêndices.

II.1.2 E/S PADRONIZADAS

Todas as E/S são realizadas por chamadas a um conjunto único de operações do NÚCLEO, que independem do periférico ao qual são dirigidas. A especificação de um arquivo segue uma sintaxe também padronizada e que independe do periférico. No capítulo sobre E/S e' detalhada esta sintaxe.

II.1.3 SISTEMA DE MEMÓRIA

Através de um sistema de PAGINAÇÃO, os processos têm espaços virtuais independentes. Porém, e' possível dois ou mais processos compartilharem áreas de memória. A facilidade de mapeamento de memória entre processos também permite a implementação de um sistema eficiente de troca de mensagens, dados e sincronização mútua.

II.1.4 RELÓGIO TEMPO REAL

Esta facilidade permite que processos se sincronizem com base em eventos que aconteçam a uma determinada hora ou após um período de tempo determinado. Também está' implementado um relógio com a hora e a data do dia.

II.1.5 SISTEMA DE REDES

O suporte a um sistema de redes está' embutido no NÚCLEO. Permite que E/S remotas sejam feitas de modo totalmente transparente aos processos. Além disto, provê recursos para que a maioria das funções do próprio Sistema Operacional possam ser executadas em outras máquinas da rede.

II.1.6 O NÚCLEO

E' formado por um conjunto de rotinas, que chamaremos OPERAÇÕES DO NÚCLEO. A maioria delas é' invocável diretamente pelos processos, via chamadas a subrotinas.

As operações do NÚCLEO residem em memória RAM. Esta região e' protegida contra escrita por HARDWARE. Os ultimos 2K dos 64K do espaco virtual de cada processo estão permanentemente mapeados sobre o NUCLEO, que e' reentrante, sendo portanto somente uma cópia usada por todos os processos. No apêndice D estão detalhadas as operações do NÚCLEO.

O acesso a regiões compartilhadas do sistema e' feita via exclusão mútua, implementada em dois níveis:

1. áreas compartilhadas somente por processos - O processo que pede o acesso 'a area, recebe uma prioridade maior que os demais processos em execução na máquina, garantindo assim a exclusão.

2. áreas compartilhadas por rotinas de interrupção - a rotina deve fazer a exclusão via inibição de interrupção, o que inibe outras rotinas de interrupção de serem ativadas. Além disto, como estas rotinas têm prioridade maior que os processos, áreas mistas (compartilhadas por processos e rotinas) devem usar esta modalidade de exclusão.

CAPITULO III

PROCESSO

E' a unidade indivisível de processamento. Um processo contém todo um contexto para a execução de um programa (Fig. a seguir). Assim além do codigo e dados (Programa), fazem parte de um processo informações que permitem ao S.O. :

1. Gerenciar a execução do processo
2. Controlar o acesso à memoria
3. Controlar o acesso a arquivos
4. Identificar o usuário "dono" do processo e consequentemente verificar seus privilégios.
5. Fazer estatísticas sobre o processo (tempo de execução, número de E/S realizadas, tamanho da pilha, etc.)

Para cada programa em execução na máquina, e' criado um processo. Ao término do programa, o processo deixa de existir; portanto, ha' uma identidade de duração entre processo e programa. Muitas vezes no texto as duas palavras são usadas com o mesmo sentido.

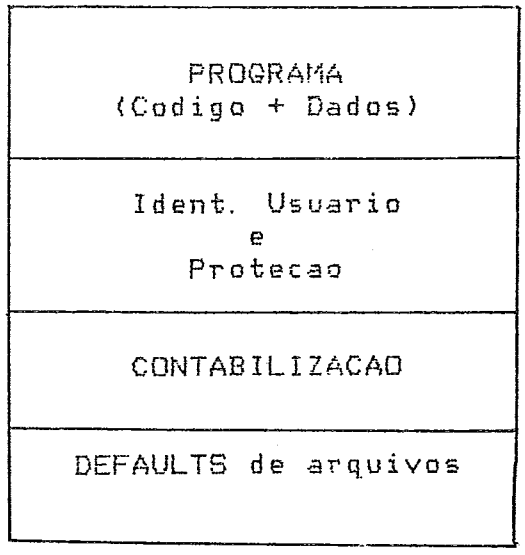

Fig. 3.1 - COMPONENTES de UM PROCESSO

Com exceção de código e dados, os demais elementos formam o "DESCRITOR de PROCESSO". Tais descritores encontram-se na TABELA de DESCRITORES de PROCESSOS, na região de memória reservada ao sistema operacional. A consulta à esta tabela pode ser feita diretamente pela OPERAÇÃO "ESTADO", ou implicitamente por algumas operações do NÚCLEO. Os processos não têm acesso direto à mesma.

III.1 IDENTIFICAÇÃO DE PROCESSO

Cada processo tem uma IDENTIFICAÇÃO do PROCESSO (IDPROC). E' um número único em uma rede de computadores, sendo composto por 2 Bytes. O de mais alta ordem contém o número da máquina de origem do processo. O segundo byte dá' um número único dentro da máquina, onde o processo esta' executando, e e' par. Para referenciar um processo, na maioria das vezes, e' necessário dar sua IDPROC. Em algumas operações (TERMINE, ESTADO, ASSOCIE) e' possível fazê-lo via o nome do processo.

O nome de um processo é' obtido a partir do campo "nome" da especificação do arquivo de onde foi carregado o programa. Isto implica que a referência a um processo por seu nome pode levar a uma ambiguidade, pois vários processos podem ter o mesmo nome. E' de responsabilidade do usuário evitar processos com mesmo nome ou então tera' de conviver com eles. Como exemplo, para cancelar a execução de um processo que seja de outro usuário (operação TERMINE), e' necessário que seja feito via sua IDPROC

e não pelo nome do processo.

III.2 CARACTERÍSTICAS DOS PROCESSOS

Uma série de recursos são oferecidos aos processos, de modo que os mesmos possam usufruir ao máximo os recursos que o sistema oferece. Naturalmente isto leva à particularização e complexidade maiores na confecção dos programas. Como alguns programas não necessitam destas sofisticações e visando simplificar a compatibilidade com código gerado por outros sistemas ou compiladores externos, os processos foram divididos em duas classes: processos SIMPLES e processos COMPLETOS.

III.2.1 PROCESSOS SIMPLES

E' dada a seguir uma descrição das possibilidades e características que um processo SIMPLES pode ter no sistema.

1. 64K de endereçamento, sendo os últimos 2K mapeados sobre a área do NÚCLEO; deste modo, 62K são disponíveis para o código e dados do programa. A Fig 3.2 mostra o mapa de memória dum processo simples.

2. A pilha "S" cresce a partir do endereço 62K em direção ao endereço 0. A área alocada para a mesma depende da extensão do programa. O gráfico na Fig.3.3 relaciona tamanho de programa e tamanho da pilha "S". O sistema operacional (via o carregador) e' responsavel por dimensionar e alocar memória física para esta área no momento da carga do programa.

3. RECURSIVIDADE: As subrotinas ou "procedures" podem se auto invocar, indefinidamente (ou ate' terminar a PILHA).

4. Acesso aos recursos do S.O. : a maioria dos recursos, sendo excluídos os de gerência de memória.

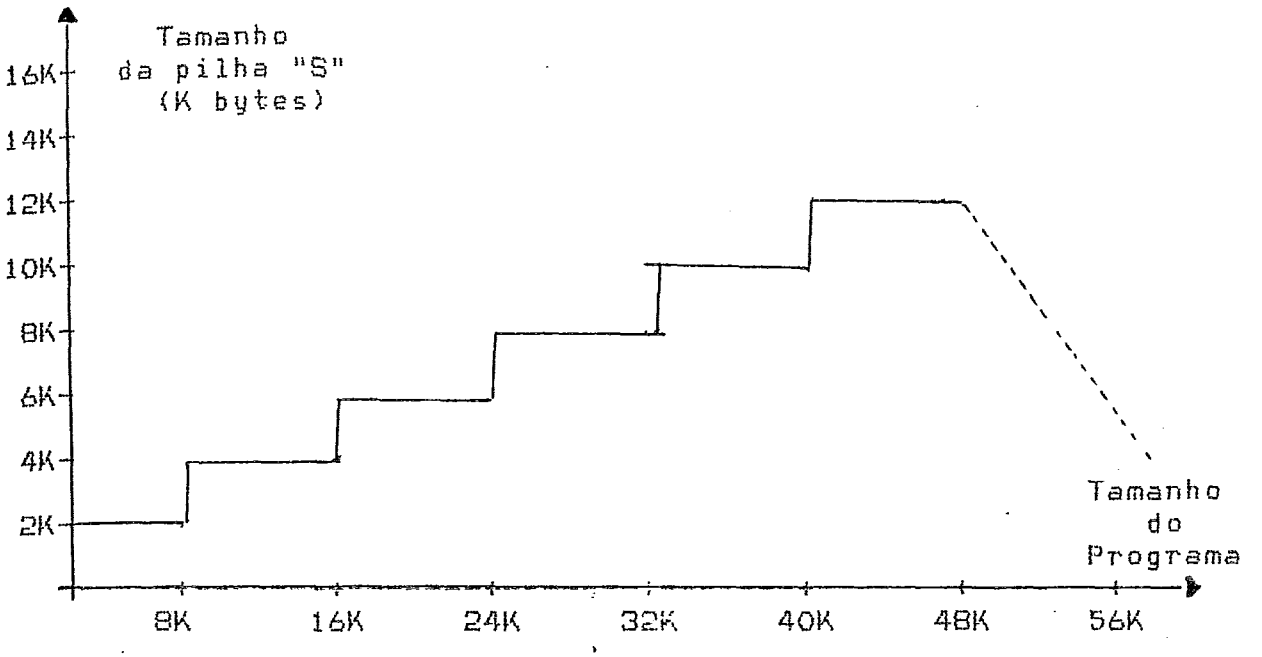

Fig. 3.3 - ALOCAÇÃO AUTOMÁTICA DA PILHA "S"

III.2.2 PROCESSOS COMPLETOS

1. Paginação do processo: Na versão atual do NÚCLEO, um processo pode ser composto de até 512 páginas de 2K cada uma, fazendo um total de 1 Mbyte de memória virtual.

2. Alocação Dinâmica de Memória : é possível criar-se páginas durante a execução do processo. Também existem recursos para a liberação de páginas dinamicamente.

3. O tamanho máximo da PILHA "S" pode ser determinado pelo programador no momento da confecção do programa. Caso isto não seja feito, o mesmo procedimento para programas simples é utilizado no dimensionamento da PILHA (fig. 3.3) Outra possibilidade é o dimensionamento DINÂMICO, em tempo de execução, quando pode ser aumentada ou diminuída.

4. Recursividade - O código pode ser recursivo, sendo o nível limitado somente pelo tamanho da PILHA.

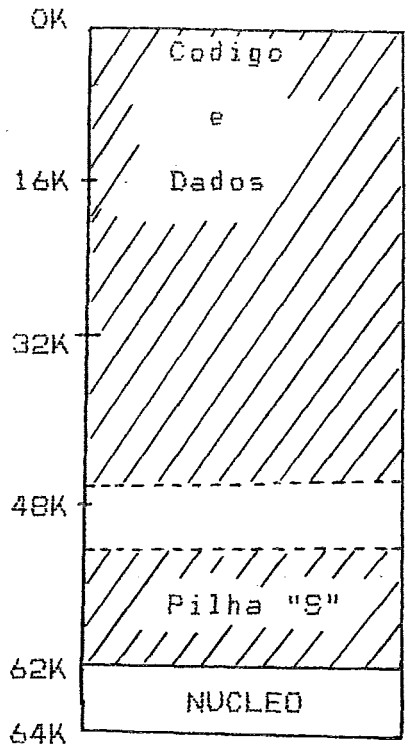

Fig. 3.2 - USO DE MEMÓRIA POR UM PROCESSO SIMPLES

O arquivo que contém o código executável de um programa simples, tem uma estrutura interna bastante simplificada. com isto, a carga deste tipo de programa a partir de um periférico qualquer (exemplo, linha RS232) não apresenta problemas, simplificando sobre-maneira a ja' citada compatibilidade. No capítulo CARREGADOR e' apresentado o formato deste tipo de arquivo

5. Todos os recursos do S.O. são acessíveis, com especial atenção a gerência de memória. Estes recursos englobam alocação/liberação dinâmica de memória, acesso a áreas de memória de outros processos, acesso compartilhado de memória, etc. Maiores detalhes no capítulo de GERÊNCIA de MEMÓRIA.

III.3 REENTRÂNCIA/COMPARTILHAMENTO DE MEMÓRIA

Um processo COMPLETO pode ser PARCIALMENTE ou INTEIRAMENTE reentrante. Cada página pode ou nao ser reentrante, independente das demais páginas do processo.

Processos PARCIALMENTE reentrantes são aqueles que têm algumas de suas páginas compartilhadas. Este e' o caso típico de compartilhamento de rotinas de bibliotecas (ex. rotinas matemáticas, ordenação, etc.). Outro caso e' o de compartilhamento de áreas de dados por 2 ou mais processos. Ainda outro exemplo são os processos que so' compartilham áreas impuras de dados.

Processos INTEIRAMENTE reentrantes têm todas páginas reentrantes, inclusive áreas de dados. Isto implica em poder haver várias execuções simultâneas deste programa e somente uma cópia estar na memória.

O CARREGADOR de PROGRAMAS e' o responsável por realizar o compartilhamento de memória tanto parcial quanto total. Maiores detalhes podem ser encontrados no capítulo CARREGADOR e no tópico multiexecução, a seguir.

III.4 REEXECUÇÃO

Quando um programa COMPLETO e' feito, e' possível dar-lhe a característica REEXECUTÁVEL. Isto indica que o processo, após terminar sua execução nao e' destruído, ficando o maior tempo possível na memória. Se um novo pedido de execução do programa e' feito e o mesmo ainda se encontra na memória, ele e' reativado e reexecutado.

O tempo que um processo reexecutável permanece na memória obedece a uma ordem LRU (Last Recently Used). Deste modo, programas ou utilitários frequentemente usados normalmente não precisam ser recarregados a cada execução.

Cabe ressaltar que o programa e' o responsável por manter-se em um estado consistente entre uma execução e outra.

III.5 MULTIEXECUÇÃO

Se um programa já' está executando num processo e uma nova execução do mesmo é' pedida, um novo processo e' criado. So' que as páginas "nao modificáveis" são compartilhadas, isto e', o novo processo utiliza aquelas do processo que ja' esta' executando. Eventualmente, se todas as páginas forem declaradas "nao modificáveis", o novo processo não gasta memória alguma. E' de responsabilidade dos processos gerenciar a divisão das áreas compartilhadas sujeitas a alteração, caso existam. Somente as páginas modificáveis, caso existam, devem ser criadas ou trazidas do arquivo onde se encontra o programa. A Fig 3.4 mostra um exemplo de 2 execuções silmutâneas.

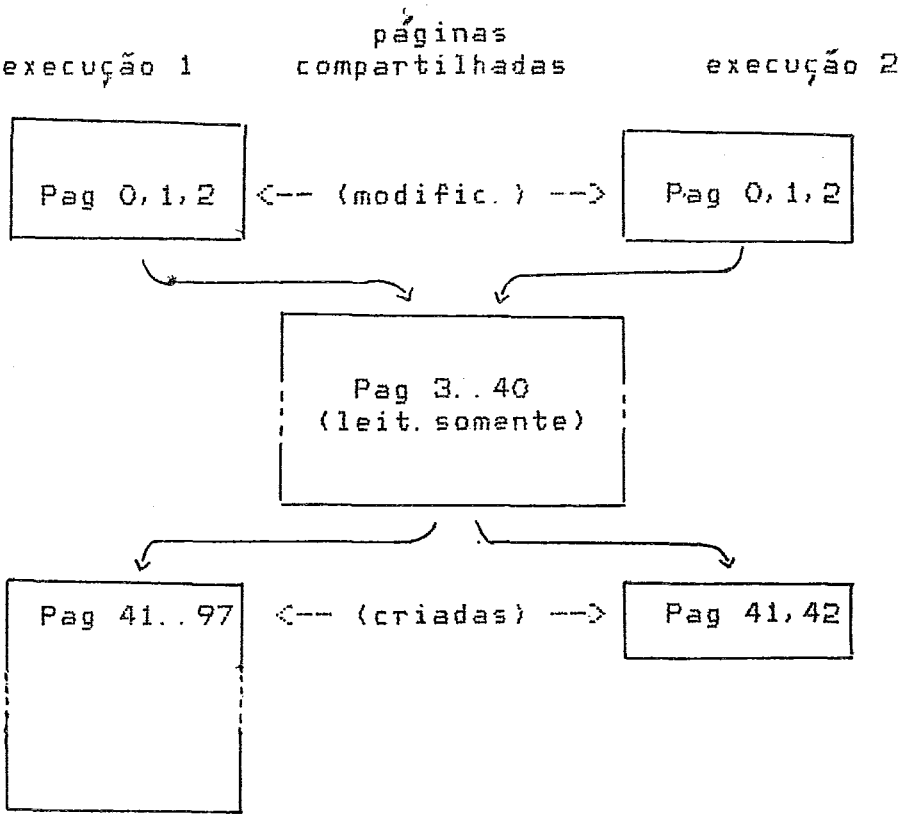

Fig.3.4 - Multiexecução de um processo

III.6 PROCESSOS PERMANENTES

Alguns processos não podem ser terminados mesmo que o usuário ou um outro processo na máquina emita uma ordem para tal. Este e' o caso de alguns processos de sistema, que se cancelados, tirariam a integridade do sistema ou tornariam impossível a continuação da operação da máquina.

E' uma opção na criação do programa, a especificação de permanente ou não. Deste modo, programas do usuário (controle de processos, máquinas com software dedicado, data-entrys, etc.) também podem ser permanentes. Alguns comentários sobre este assunto, podem ser encontrados no capítulo TERMINADOR.

CAPITULO IV

COMUNICAÇÃO ENTRE PROCESSOS

Num sistema como este, onde toda a estrutura esta' montada sobre processos, a troca de mensagens entre eles assume papel fundamental. A filosofia do sistema de troca de mensagens foi criada especificamente para este sistema, embora esteja baseada nos princípios atuais de trocas de mensagens e monitores (ref. [23], [24]).

O NÚCLEO do sistema provê uma série de recursos para a comunicação eficiente e sincronismo entre processos, via o conceito de FILAS de SERVIÇO. A nível de exemplo, de modo que fique clara a importância das filas no sistema, as operações de E/S são todas feitas por seu intermédio.

IV.1 FILAS DE SERVIÇO

Em um determinado sistema existe um número máximo de filas. Este número e' determinado durante a geração do sistema. Conforme mostrado na fig.4.1, uma fila de um lado tem os PROCESSOS REQUISITANTES e de outro um PROCESSO SERVIDOR.

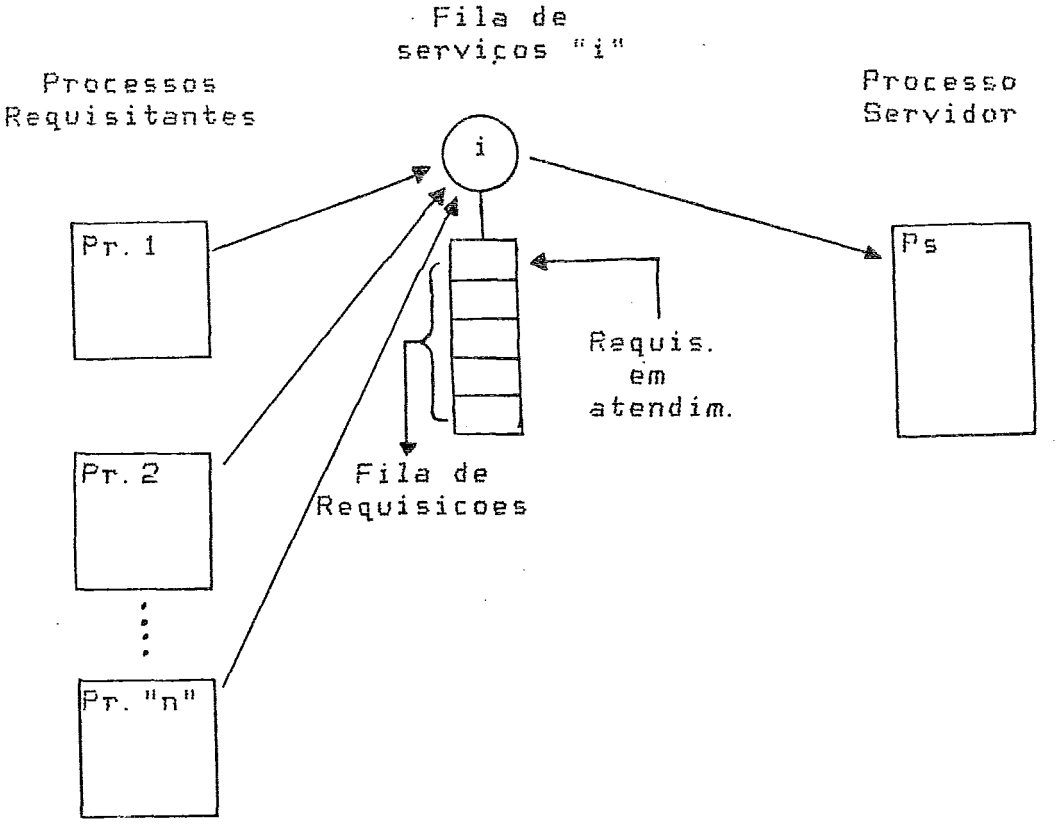

Figura 4.1 - ESQUEMA de FILAS

Uma fila de serviço pode receber requisições de vários processos, mas tem SOMENTE 1 processo servidor. As filas são numeradas de 1 a n, onde n e' um máximo determinado em tempo de geração de sistema, como ja' dito.

Quando um processo quer se comunicar ou pedir a outro a execução de determinado serviço, deve solicitá-lo ao NÚCLEO. Este, deposita a requisição no final da fila de requisições pendentes do serviço especificado. Por outro lado, o processo que vai atender as requisições de uma fila, as retira uma a uma, via chamadas ao NÚCLEO. Portanto, a ordem de atendimento e' FIFO.

O controle de sincronismo entre os processos que fazem as requisições e os que as atendem, e' feita automáticamente pelo NÚCLEO. Deste modo, o processo requisitante pode esperar ate' que o atendimento a sua requisição termine, quando ele voltara'

a executar automáticamente. O processo servidor também entra em espera quando ele faz uma retirada e não há nehuma requisição na fila .

As filas funcionam totalmente independentes umas das outras, cada uma atendendo a um propósito distinto. Existem filas PARTICULARES e PÚBLICAS. Conforme exemplificado na fig. 4.2, as particulares são usadas entre processos que se conhecem. As públicas normalmente atendem requisições gerais. Um exemplo de fila PÚBLICA e' a fila do carregador de programas, ao qual todos os processos tem acesso.

A criação/remoção de filas e' dinâmica, devendo ser criadas pelos processos que as atenderão. Quando seu uso não for mais necessário, o processo deve removê-la; assim, mais tarde, algum outro processo pode usar este espaço para a criação de nova fila.

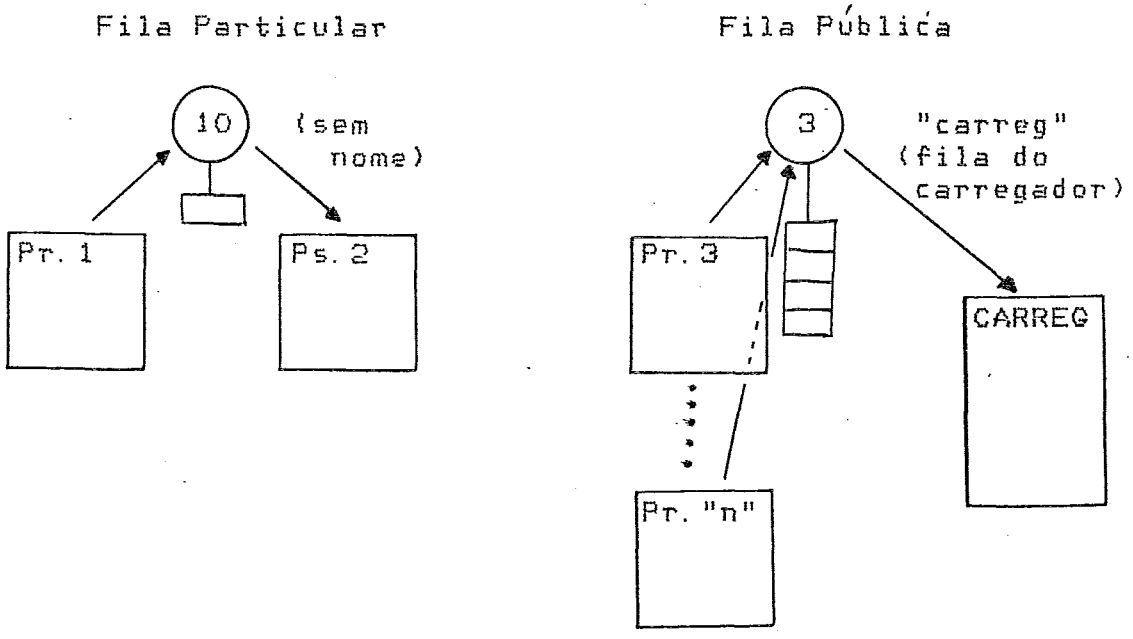

Fig. 4.2 - Exemplo de filas particular e pública

IV.2 ACESSO A FILAS

A utilização de filas de serviço no sistema e' dinâmica. Por este motivo, um processo servidor deve CRIAR uma fila ao iniciar seus serviços e TERMINA-LA quando não for mais necessária. Para tanto existe a operacao do NUCLEO CRIE-SERVICO, que deve ser invocada pelo processo que ira' servir a fila.

E' possivel dar um ou mais nomes simbolicos a uma fila. Estes nomes são dados 'a fila pela operacao NOME-A-SERVICO, e podem ter 6 caracteres no máximo Eles são gerais, i.e', qualquer processo executando no sistema pode acessa-los.

Um processo que queira fazer suas requisições a uma fila e conhece apenas um de seus nomes, deve invocar a operação ABRA-SERVICO do NUCLEO e passar este nome como parametro. Como resposta, e' devolvido o número da fila.

Com este número o processo pode então fazer as requisições à fila. A fig. 4.3, contém um exemplo do uso de ABRA-SERVICO.

IV.3 DEPOSITO/ESPERA

No lado esquerdo da fig.4.1 estão representados PROCESSOS REQUISITANTES depositando pedidos em uma fila "i". Estes pedidos são feitos ao NUCLEO via a operação DEPOSITO. Sua função e' depositar a requisição no final da fila, que consiste dos parametros descritos a seguir.

IV.3.1 PARAMETROS PARA UM DEPOSITO

Os serviços executados pelas filas são os mais variados, desde especificos entre processos particulares ate' carga de programas, E/S, etc. Portanto, a forma da mensagem passada entre processos varia enormemente tanto de forma quanto tamanho, para cada caso.

Um dos parametros a passar para DEPOSITO e' "VALOR", que e' formado por 2 BYTES. Normalmente e' o ENDERECO de uma estrutura de dados no processo requisitante; naturalmente o processo servidor deve conhecer o formato de tal estrutura. Quando a mensagem e' curta (ate' 2 BYTES), ela propria pode ser passada como o "VALOR" .

Alem do VALOR, deve-se especificar o NÚMERO da fila onde sera' depositada a requisição. Este número deve ser dado em um BYTE, onde o bit 7="1" indica uma REQUISIÇÃO COM ESPERA.

IV.3.2 REQUISIÇÕES COM ESPERA

Implica que o processo que faz a requisição deve parar de executar, ficando em espera até que a requisição seja atendida. Quando terminar o atendimento, o processo continua a executar na instrução seguinte à chamada ao NÚCLEO. São 2 os modos de entrar em espera: um no momento do DEPÓSITO, fazendo-se o bit 7=1 no segundo parâmetro. O outro é via a operação ESPERA, que ao ser invocada, suspende o processo, caso haja requisição pendente ou sendo atendida na fila. Como parâmetro, ESPERA deve receber o número da fila. Caso não haja requisições pendentes, o processo continua a executar normalmente.

```
(* este exemplo é' feito numa linguagem
hipotética, visando mostrar o uso das diretivas de
manipulação de filas
*)

(*      definição das variáveis  *)

fila1 : BYTE; (* variáveis que receberão os nros.
*)
fila2 : BYTE; (* da filas a partir de seu nome
*)

nome1 : CADEIA "GARREG"; (* nome da fila do
carregador *)
estrut : RECORD (* estrut. hipot. do carregador *)
           nomeprog : CADEIA "PROG1.EXE"
           operação: BYTE;
         END;
mensag : CADEIA (50); (* estrut. p/ msg particular
*)



(* início do programa *)

  ABRA-SERVIÇO(nome1,fila1);  (* deter. nro. da
  fila *)
  ABRA-SERVIÇO("joão",fila2); (* a  partir do nome
  *)
    -
    -
    -
  DEPÓSITO ( fila1, estrut);      (* sem espera *)
  DEPÓSITO ( fila2+$80, mensag); (* c/ espera  *)
    -
    -
    -
  ESPERA ( fila1); (*espera término atendim. fila1
  *)
    -
    -
    -
FIM.
```

Fig. 4.3 - EXEMPLO DE PROCESSO REQUISITANTE

IV.4 RETIRA/FINALIZA

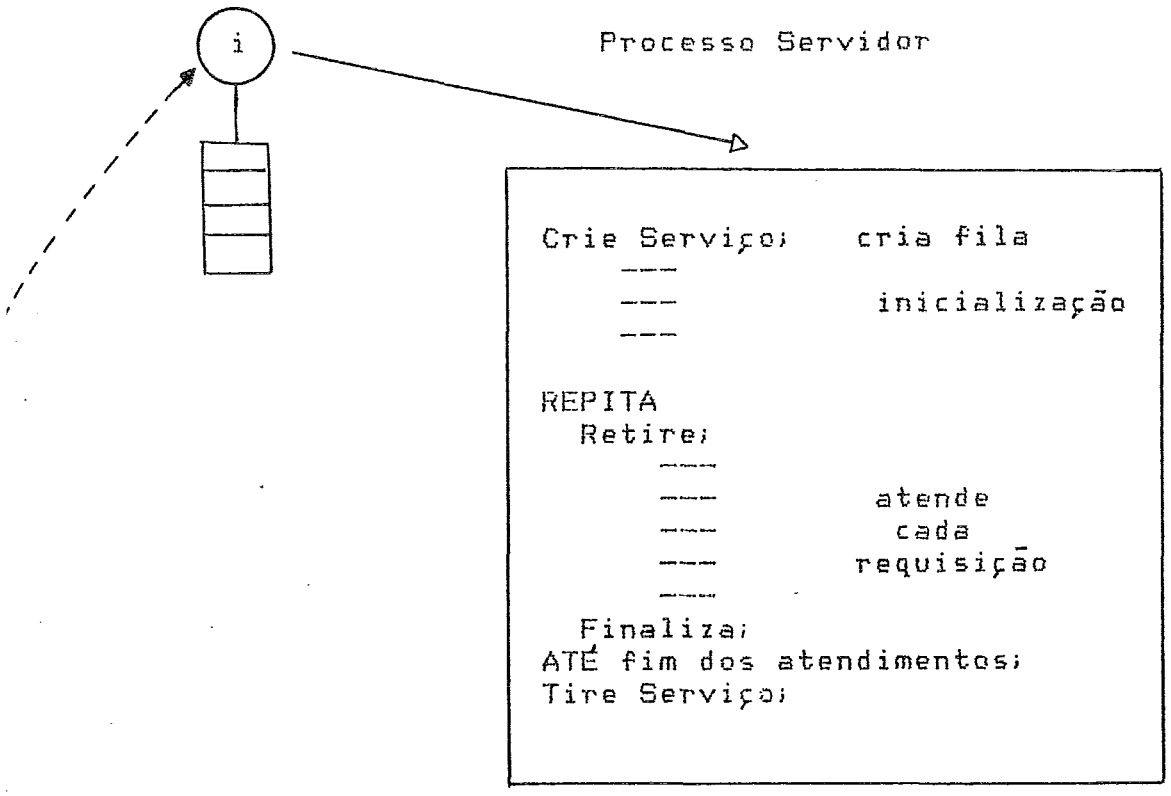

Fig.4.4 - UM SERVIDOR TÍPICO

O par de operações RETIRA/FINALIZA e' usado pelos processos servidores que ATENDEM as requisições de uma fila. As figuras 4.4 e 4.5 complementam o texto a seguir. A operação RETIRADA, devolve uma cópia da 1a. requisição que está' no topo da fila especificada. Ao terminar o atendimento de uma requisição, a operação FINALIZA deve ser invocada. Ela verifica se o processo que fez o DEPOSITO está' em espera, neste caso reativando-o.

Se não ha' nenhuma requisição na fila no momento que uma RETIRADA e' feita, o processo que a fez entra em ESPERA, ate' que uma requisição seja depositada naquela fila por um outro processo.

IV.4.1 PARÂMETROS DE UMA RETIRADA

Deve-se especificar no primeiro parâmetro o número da fila a acessar. No segundo, deve-se dar o endereço de uma palavra dupla (4 bytes), que recebera' a requisição do topo da fila. Nos 2 primeiros BYTES e' devolvida a IDPROC do processo que fez o depósito. Nos 2 BYTES seguintes, vem o "VALOR" que foi depositado.

IV.5 PROCESSOS SERVIDORES

A figura 4.4 da' uma visão geral de um processo servidor. A fig.4.5 apresenta, em uma linguagem hipotética, as chamadas as operações do NÚCLEO, feitas por um processo servidor. Normalmente um processo servidor tem uma característica CIRCULAR, so' terminando, quando não mais for necessário o atendimento 'a fila.

```
(* Exemplo simplificado de um carregador de
programas em uma linguagem hipotética. Notar que
não houve rigor na passagem de parâmetros p/ as
operações, para não dificultar o entendimento
lógico.

     O apêndice D.4 deve ser usado no
acompanhamento. O nome do arquivo a carregar e'
passado como parâmetro. *)


VAR
   nro_fila : BYTE;
   parâmetro : RECORD
                 nome,
                 IDproc : WORD;
               END;

início

  ---> inicializa fila

  Crie_serviço ( nro_fila);
  Nome_serviço ( 'CARREG', nro_fila);

  ---> loop pega cada requisição
  REPITA
    RETIRADA   (nro_fila, parâmetro); ---> pega req.
    crie_processo;
    abre-arquivo ( parâmetro.nome );
```

```
        ---> le o programa para memória
        REPITA
           leia_registro;
        ATÉ fim de arquivo;

        FINALIZA   (nro_fila); ---> termina atendim.

      PARASEMPRE;  ---> carregador sempre ativo

    fim.
```

Fig. 4.5 - *exemplo de um processo servidor*

IV.5.1 LIBERAÇÃO DE RECURSOS ALOCADOS

Em servidores que alocam recursos para as requisições que estão servindo (p. exemplo buffers), logo após a operação RETIRADA, deve ser feito um teste para verificar se o "VALOR" e' zero. E' que quando o NÚCLEO esta' terminando um processo, ele deposita uma requisição com "VALOR"=0 nas filas que foram usadas pelo processo. A IDPROC e' a do processo que esta' sendo terminado.

Portanto, quando o processo servidor receber uma requisição com "VALOR" 0, deve assumir que o processo que a fez esta' sendo terminado. Deste modo, todos os recursos alocados ao mesmo podem ser liberados. Alguns detalhes a mais sobre este assunto podem ser encontrados no capítulo TERMINADOR.

IV.5.2 RECEBENDO PARÂMETROS

Os parâmetros que informam ao processo servidor a tarefa a executar, normalmente estão na área de memória do processo que fez o pedido. Existe a operação MAPVIR do gerente de memória, que pode ser usada para ter-se acesso a estas áreas.

No caso de um processo servidor, MAPVIR mapeia uma parte de seu endereço virtual sobre uma área do processo que fez o depósito na fila. Normalmente, o "VALOR" retirado da fila da' o endereço no processo requisitante onde estão os parâmetros, e consequentemente onde deve ser feito o mapeamento. Deste modo, o processo servidor tem acesso aos dados que estão no processo requisitante. Maiores detalhes desta operação e mapeamento entre processos podem ser encontrados no capítulo GERENTE de MEMÓRIA.

IV.5.3 USOS DE PROCESSOS SERVIDORES

Uma boa parte das funções do sistema operacional são acessadas via filas de serviço. Este e' o caso de todo o sistema de E/S, onde cada gerente de periférico e' um processo servidor com sua respectiva fila. Portanto, pedidos de E/S são feitos em última instância por depósitos em filas. Maiores detalhes são apresentados no apêndice sobre implementação de GERENTES de E/S.

O CARREGADOR de programas também e' implementado segundo esta política. Os pedidos de carga e execução de programas devem ser depositados na fila "CARREG", que e' atendida pelo processo servidor CARREG. No capítulo CARREGADOR encontram-se maiores detalhes.

Ainda como exemplo temos o sistema de REDES [1], a linguagem de COMANDOS [3], que são todos implementados segundo este método. Como ja' citado, servidores para uso particular (contrôle de processos, execução paralela de tarefas, etc) também são aplicações possíveis de implementação.

CAPITULO V

E/S

Neste capítulo, e' dada uma visão dos conceitos e recursos oferecidas pelo NÚCLEO ao sistema de E/S. No apêndice B são apresentados detalhes da implemetação de gerentes de periféricos.

V.1 CARACTERÍSTICAS

As E/S são INDEPENDENTES de periférico. Além disto, elas são PADRONIZADAS. Estas duas características são implementadas no NÚCLEO do sistema, sendo portanto acessíveis a qualquer nível, conforme pode ser visto na fig. 2.1.

A INDEPENDÊNCIA de E/S permite que um programa, tanto em assembler quanto em linguagem de alto nível possa ser confeccionado sem fazer referência a periférico específico; basta no momento da execução, informar qual o periférico/arquivo efetivamente usado.

A PADRONIZAÇÃO das E/S no nível mais interno do NÚCLEO, leva a um mesmo tratamento e linguagem por parte dos demais módulos, quer sejam do sistema ou do usuário. Esta padronização compreende uma sintaxe, para a especificação dos nomes de ARQUIVOS e o conceito de NOMES LÓGICOS.

V.2 NOMES DE ARQUIVOS

E' necessário neste ponto introduzir alguns conceitos. O primeiro deles e' o de VOLUME. Ao se especificar um arquivo, e' necessário dizer o NOME do VOLUME, isto e', meio físico onde o mesmo se encontra. No caso de discos e fitas, VOLUME e' o nome de um determinado disco ou fita que esta' montado em uma das unidades naquele instante. No caso de impressoras, teclados, etc., VOLUME e' o nome do periférico. Portanto, com este conceito unificado, não e' necessário saber em que unidade esta' montado um disco. Através de seu nome o sistema se encarrega de descobrí-lo.

O segundo conceito e' o de VOLUMES ESTRUTURADOS e NÃO ESTRUTURADOS. O primeiro caso e' de meios que suportam mais de um arquivo, como por exemplo discos, fitas, etc. Nêles, é' necessária além da especificação do volume, a especificação do arquivo dentro do volume. No segundo caso, nos não-estruturados, enquadram-se os periféricos que não têm subdivisões em arquivos, como por exemplo impressoras, teclados, vídeos, etc. Neste caso, basta especificar o VOLUME.

O terceiro e último conceito e' o de DIRETÓRIO. Conforme pode ser visto abaixo, isto permite criar vários grupos de arquivos correlatos em um mesmo volume.

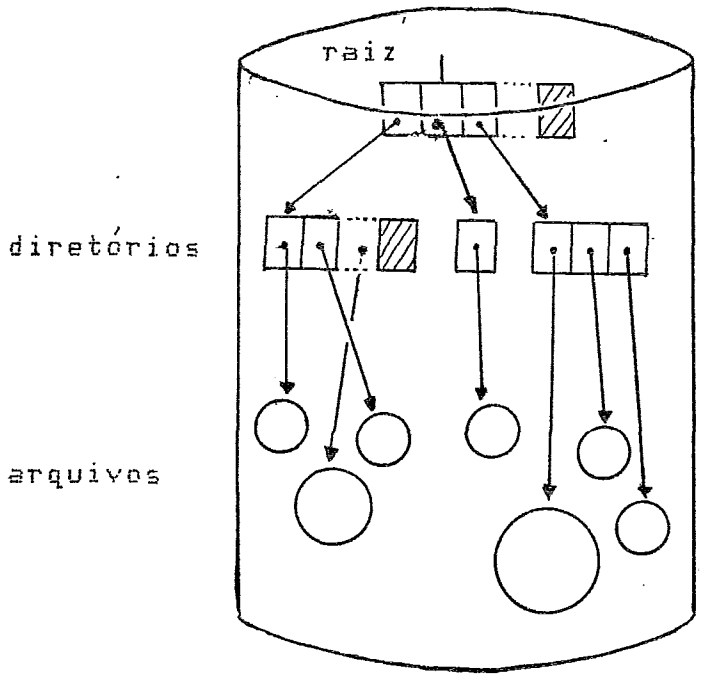

Fig. 5.1 - VOLUME ESTRUTURADO

Por NOME de ARQUIVO subtende-se uma definição completa que permita especificar sem ambiguidades um arquivo, quer em volumes estruturados ou não. Segue-se a sintaxe genérica, que e' usada em todos os níveis do sistema [13]. Notar que a pontuação e' OBRIGATÓRIA.

NO: VOLUME: DIRETÓRIO: NOME: EXTENSÃO

onde

NO: - um número entre 0 e 255, que informa em que

máquina, num conjunto interligado pelo sistema de rede, se encontra o arquivo.

VOLUME:- nome do volume montado em uma unidade (disco, fita, etc.) ou nome da unidade (impressora, teclado, tela). Pode ter de 1 a 6 caracteres quaisquer, com exceção de ":", ".", ","

DIRETÓRIO: - nome do diretório onde se encontra o arquivo. E' obrigatório somente em VOLUMES estruturados. Formado de 1 a 6 caracteres quaisquer, com exceção de ":", ".", ",".

NOME: - nome do arquivo. Usado obrigatoriamente em volumes estruturados, pode ser composto por até' 6 caracteres quaisquer, com exceção de ":", ".", ",".

EXTENSÃO: - 3 caracteres, indica o tipo do arquivo. Segue uma lista de nomes pre'-definidos, embora o usuário possa criar qualquer outro.

DAT - arquivos de dados
BAS - arquivos com programas BASIC
PAS - arquivos com programas PASCAL
FOR - arquivos com programas FORTRAN
COB - arquivos com programas COBOL
PRO - arquivos de comandos SIRIUS
EXE - arquivos com programas na forma executável
TXT - arquivos com textos genéricos
BIB - arquivo de bibliotecas

Somente os periféricos estruturados necessitam dos campos DIRETÓRIO, NOME, EXTENSÃO ; para os não estruturados, basta especificar até' o campo VOLUME.

V.3 DEFAULTS

Como visto, a sintaxe acima e' algo longo para digitar-se a todo instante que a referência a um arquivo fôr feita. Assim, e' possível especificar-se a priori um valor para os campos NO, VOLUME e DIRETÓRIO. Deste modo, numa referência a um arquivo onde um ou mais campos estiverem faltando, automaticamente são preenchidos com os DEFAULTS. Os DEFAULTS podem ser

especificados no momento em que o usuário entra na máquina, sendo válidos até o término da sessão ou até um comando "DEFAULT" ser dado na SIRIUS.

NOME	ASSUME POR DEFAULT
117:PROJ:FONTES.CALC.PAS	- - -
PROJ:CALC.EXE	no, diretório
COEFIC.DAT	no, volume, diretório
TELA:	no
23:IMPR:	- - -

FIG 5.2 - EXEMPLO NOME DE ARQUIVOS

V.4 NOMES LÓGICOS

Um programa pode ser confeccionado, fazendo-se a referência a arquivos via NOMES LÓGICOS. Mais tarde, antes de executar o programa, associa-se este nome lógico a um nome de arquivo, via o comando ASSOCIE da SIRIUS ou via a operação ASSOCIE do NÚCLEO. A partir daí, todas as referências ao nome lógico são dirigidas ao arquivo a ele associado. Outro caso típico é o uso dos nomes lógicos padrões do sistema, como por exemplo SISENT, SISSAI, LISTA, BIB, que por default estão associados, a nível de sistema, ao teclado, vídeo, impressora e biblioteca respectivamente.

Um nome lógico é composto por até 6 caracteres quaisquer, excluindo ":", ".", ",". Onde fôr valido especificar-se um nome de arquivo, também é válida a especificação de um nome lógico. A conversão do nome lógico para o nome de arquivo equivalente é feita pelas rotinas do NÚCLEO que manipulam com arquivos, que serão vistas adiante. A distinção entre os dois casos é feito pela pontuação. No nome de arquivo aparecem os símbolos ":" e/ou ".". No nome lógico, eles não aparecem.

NOME LÓGICO	NOME de ARQUIVO
ENTRA	X.DAT
ORDENA	2:TESTE:PROGS.PGM1.DAT
SISSAI	TELA:
IMPRES	25:IMPR:

Fig 5.3:- ASSOCIAÇÃO DE NOMES LÓGICOS A ARQUIVOS.

Um nome lógico pode estar associado a um nome de arquivo ao nível do processo ou do sistema. A nível de processo significa que somente tem acesso ao nome lógico o processo ao qual ele foi associado. Assim, os nomes lógicos de um processo não interferem com os de outro. A nível de sistema significa que o nome lógico e' acessível a todos os processos. Uma associação a nível de processo tem uma prioridade maior que a nível de sistema. Portanto, um nome lógico pode estar associado nos 2 níveis, e somente se o não estiver a nível de processo e' que o nível sistema e' verificado. No tópico OPERAÇÕES AUXILIARES deste capítulo, encontram-se maiores detalhes sobre associações.

A SIRIUS [3] se utiliza extensivamente desta facilidade na maioria de seus comandos.

V.5 BCES

Ao se requisitar uma operação de E/S, e' necessário passar uma série de informações para o gerente do periférico, tais como código da operação, tamanho e endereço dos dados, etc. Para tanto existe uma estrutura de dados padronizada, o Bloco de Controle de Entrada ou Saída (BCES).

A maioria dos trabalhos desenvolvidos em torno deste NÚCLEO se utilizam ou fazem referência a esta estrutura.

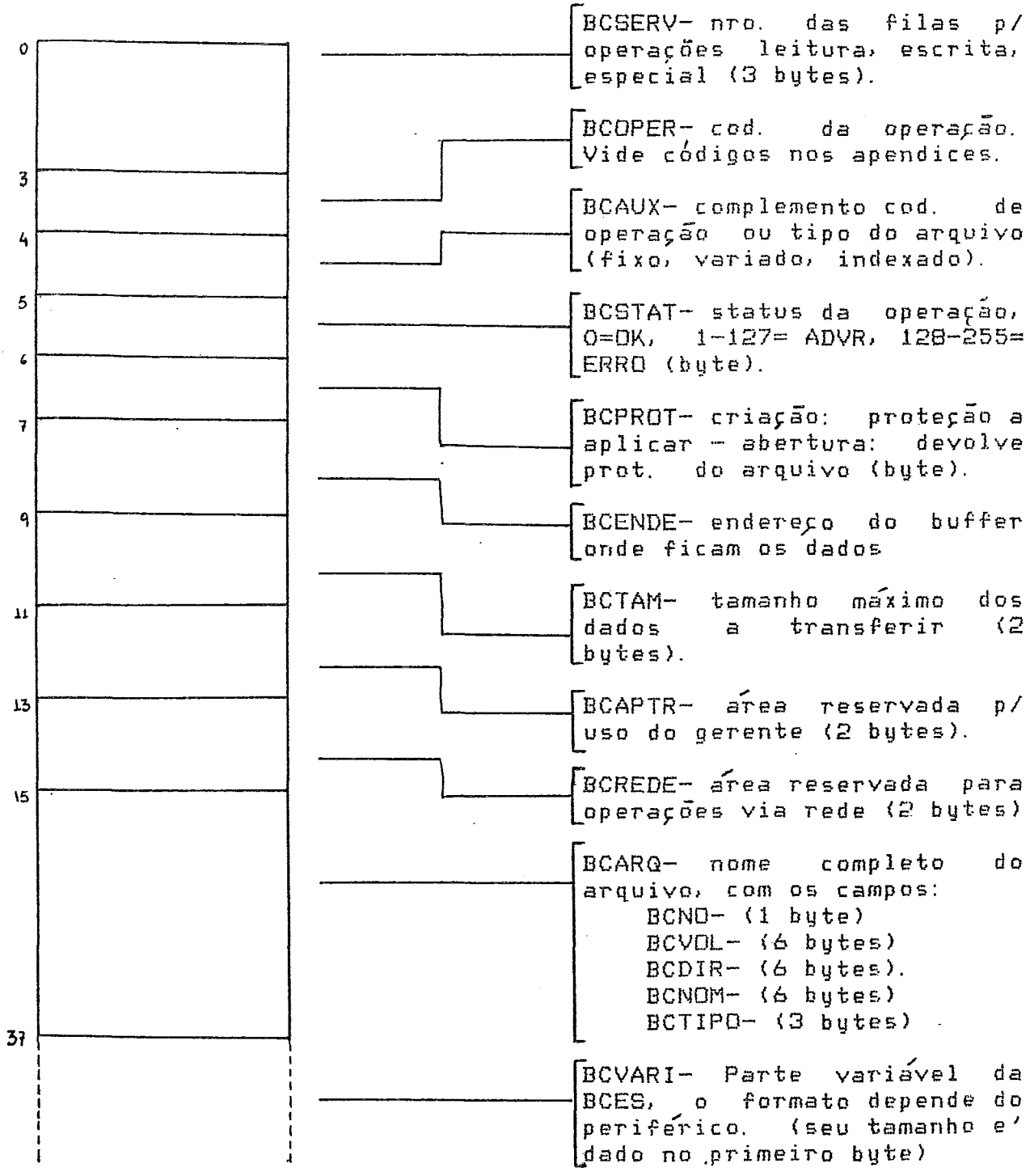

Fig 5.4 - BCES

A parte FIXA desta estrutura e' comum a todos os tipos de periféricos. A parte VARIÁVEL, e' reservada e específica para os periféricos que necessitem informações adicionais e particulares para exercerem as suas funções. Um exemplo e' o