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 maquinas geograficamente 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.

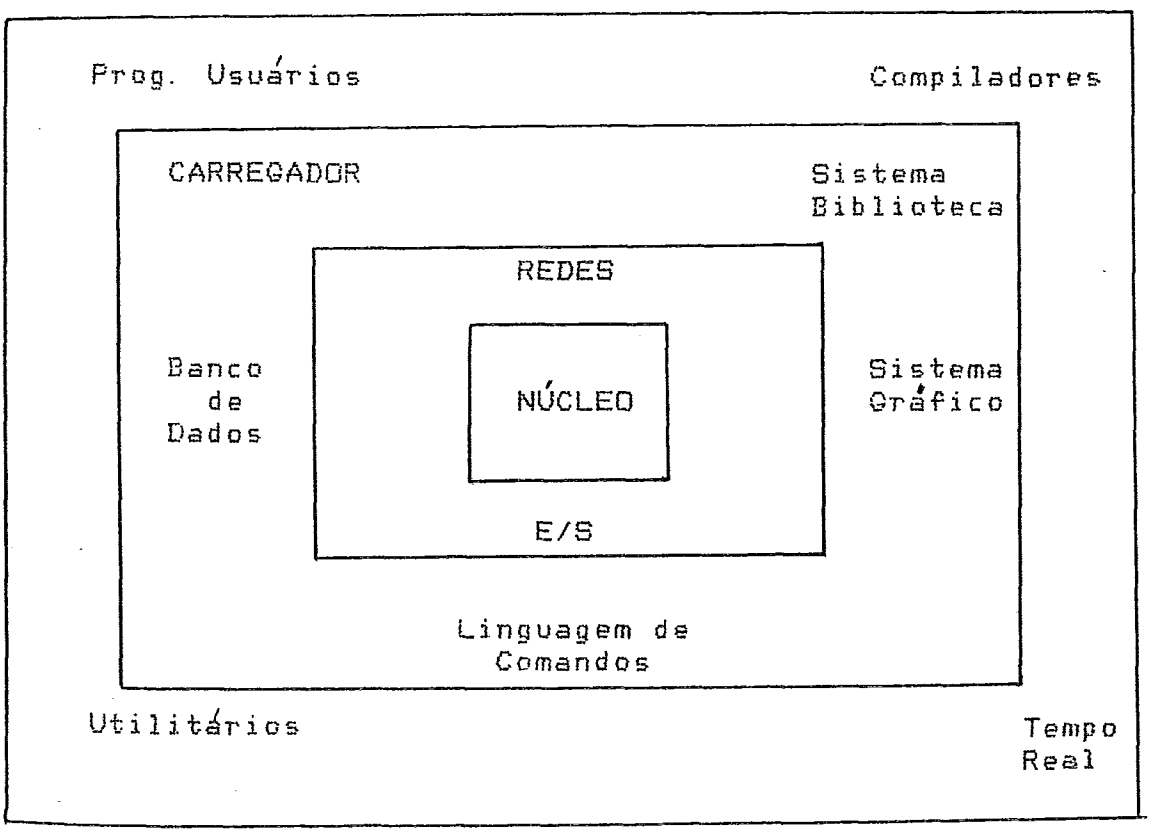


Fig. 2.1 - CAMADAS DO S.O.

Neste trabalho, nos capítulos que se seguem, serão apresentados os conceitos básicos do sistema operacional, isto é, 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 é 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 há 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 é 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 é detalhada esta sintaxe.

II.1.3 SISTEMA DE MEMÓRIA

Através de um sistema de PAGINAÇÃO, os processos têm espaços virtuais independentes. Porém, é 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.

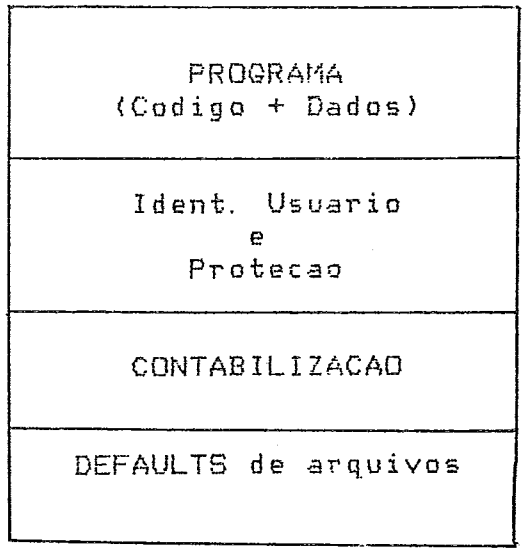


Fig. 3.1 - COMPONENTES de UM PROCESSO

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.

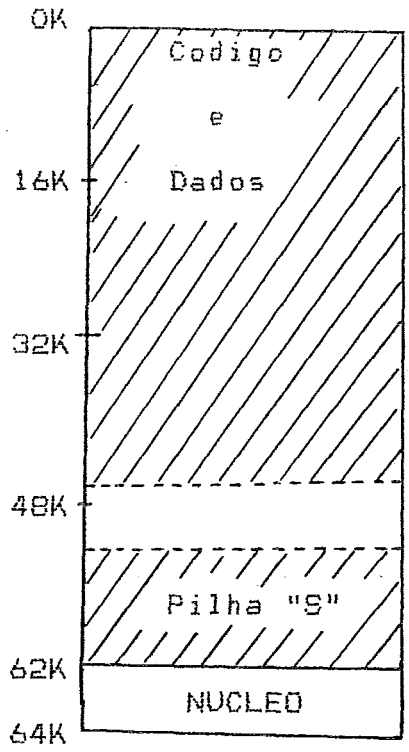


Fig. 3.2 - USO DE MEMÓRIA POR UM PROCESSO SIMPLES

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

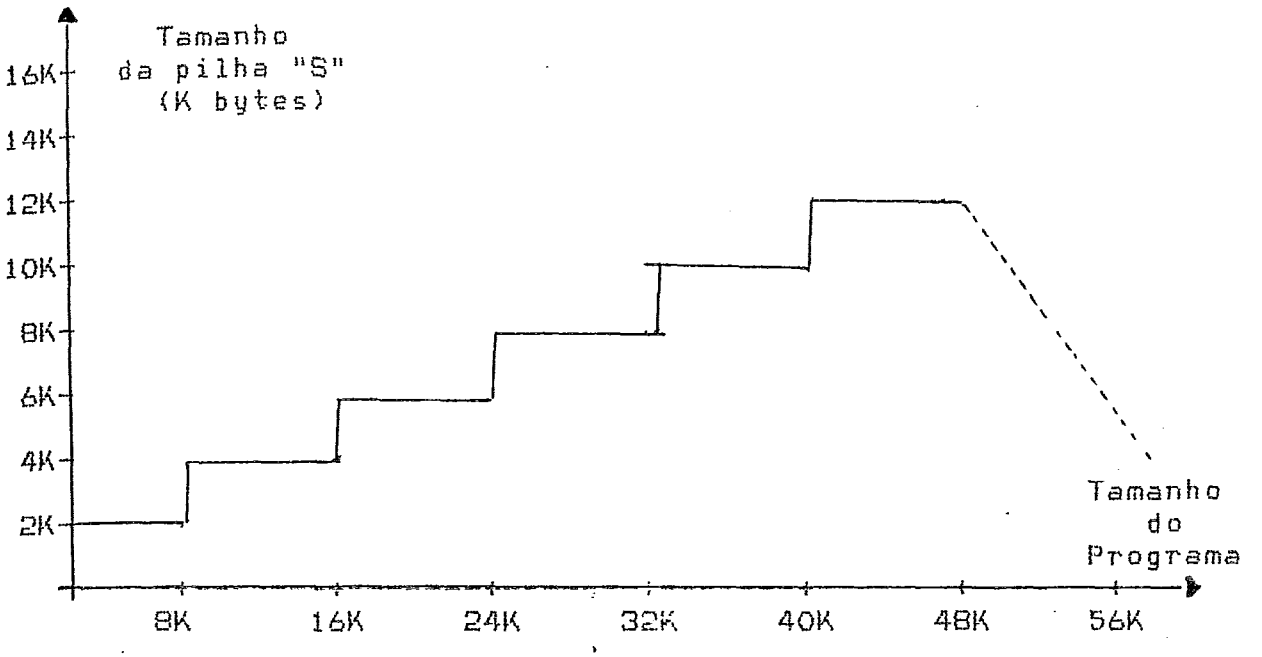


Fig. 3.3 - ALOCAÇÃO AUTOMÁTICA DA PILHA "S"

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.

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.

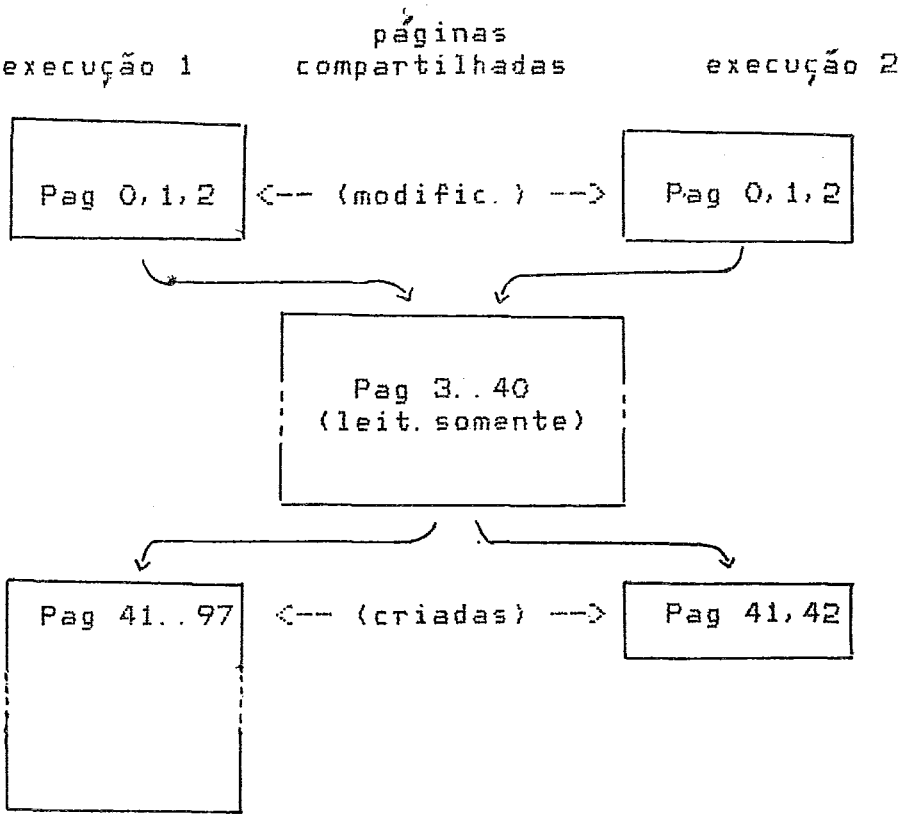


Fig.3.4 - Multiexecução de um processo

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.

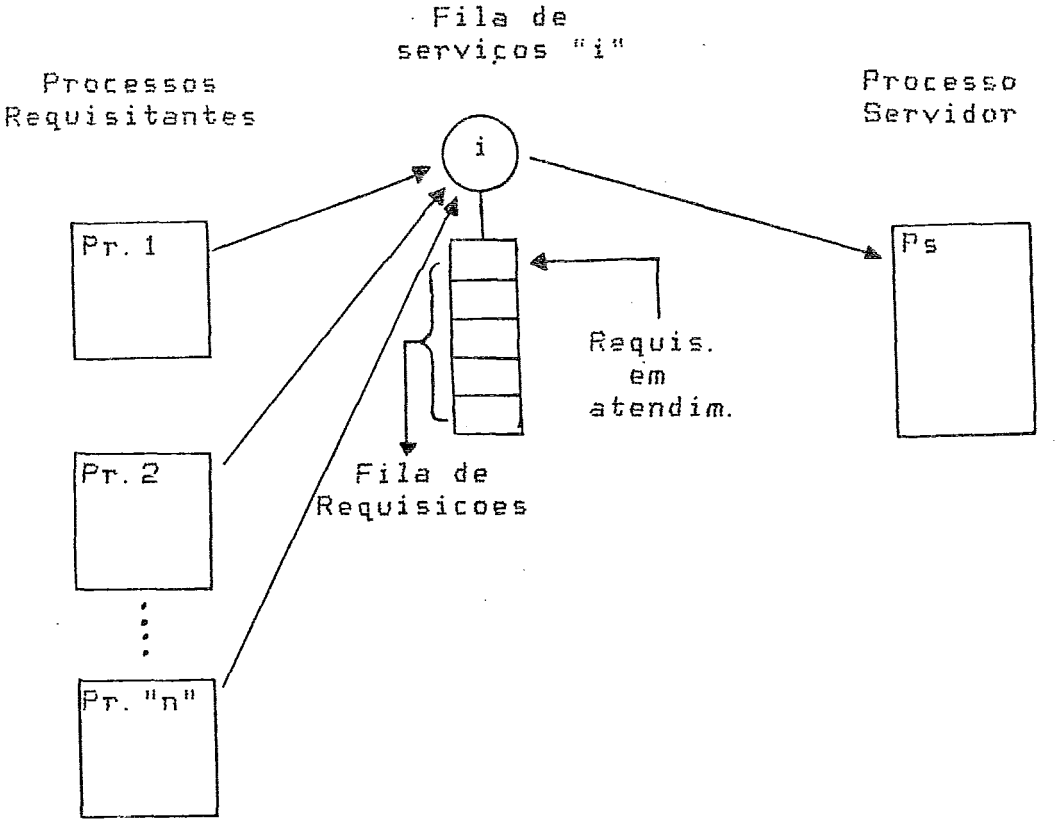


Figura 4.1 - ESQUEMA de FILAS

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.

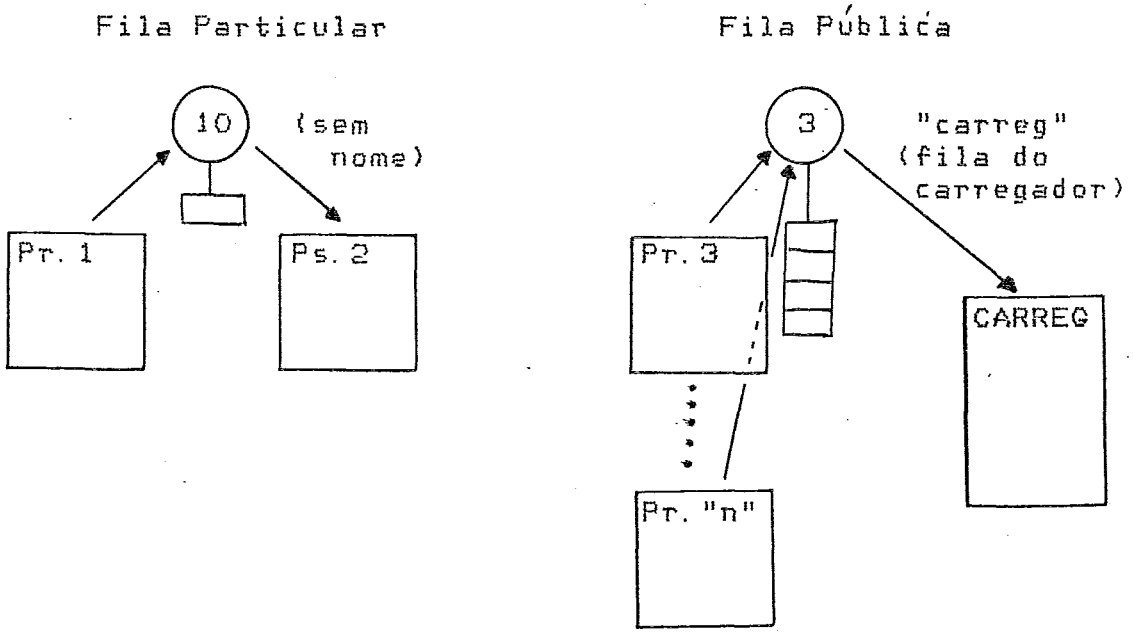


Fig. 4.2 - Exemplo de filas particular e pública

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

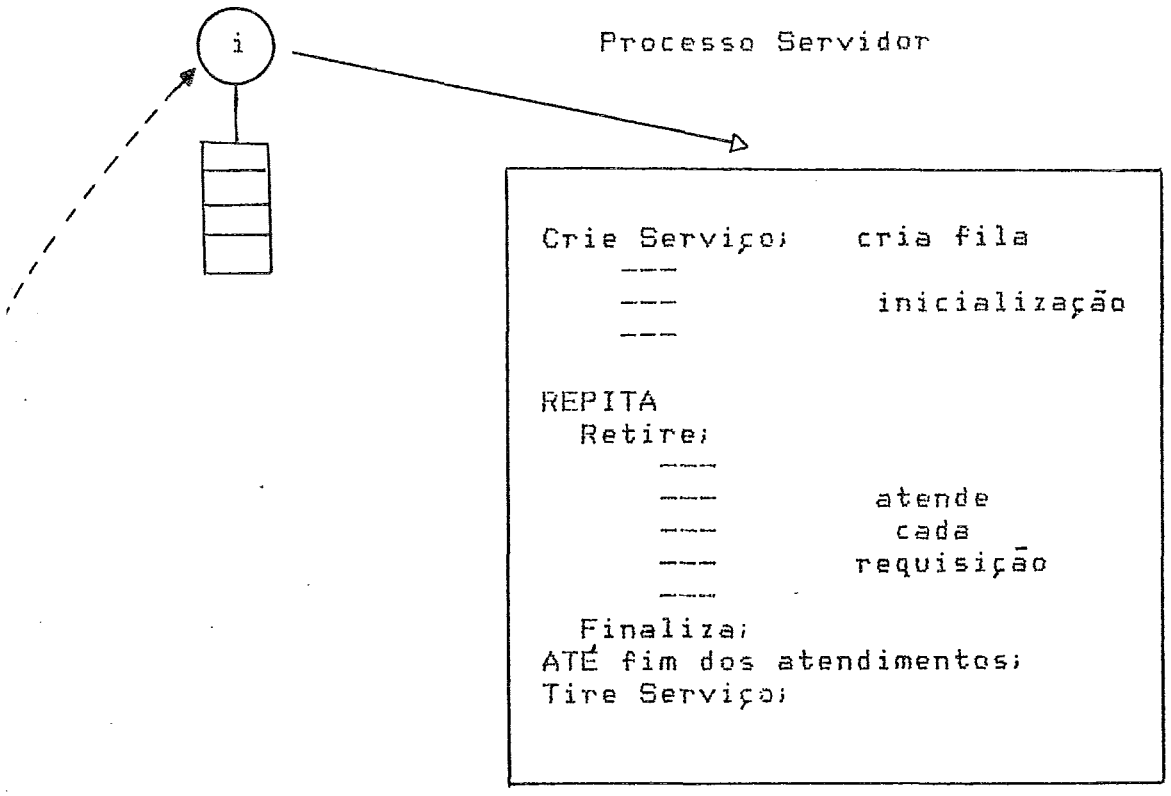


Fig.4.4 - UM SERVIDOR TÍPICO

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.

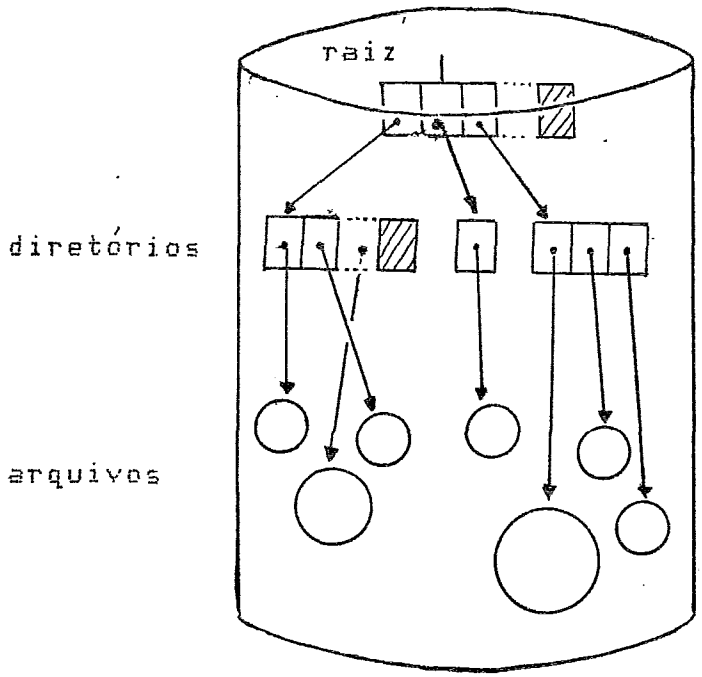


Fig. 5.1 - VOLUME ESTRUTURADO

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.

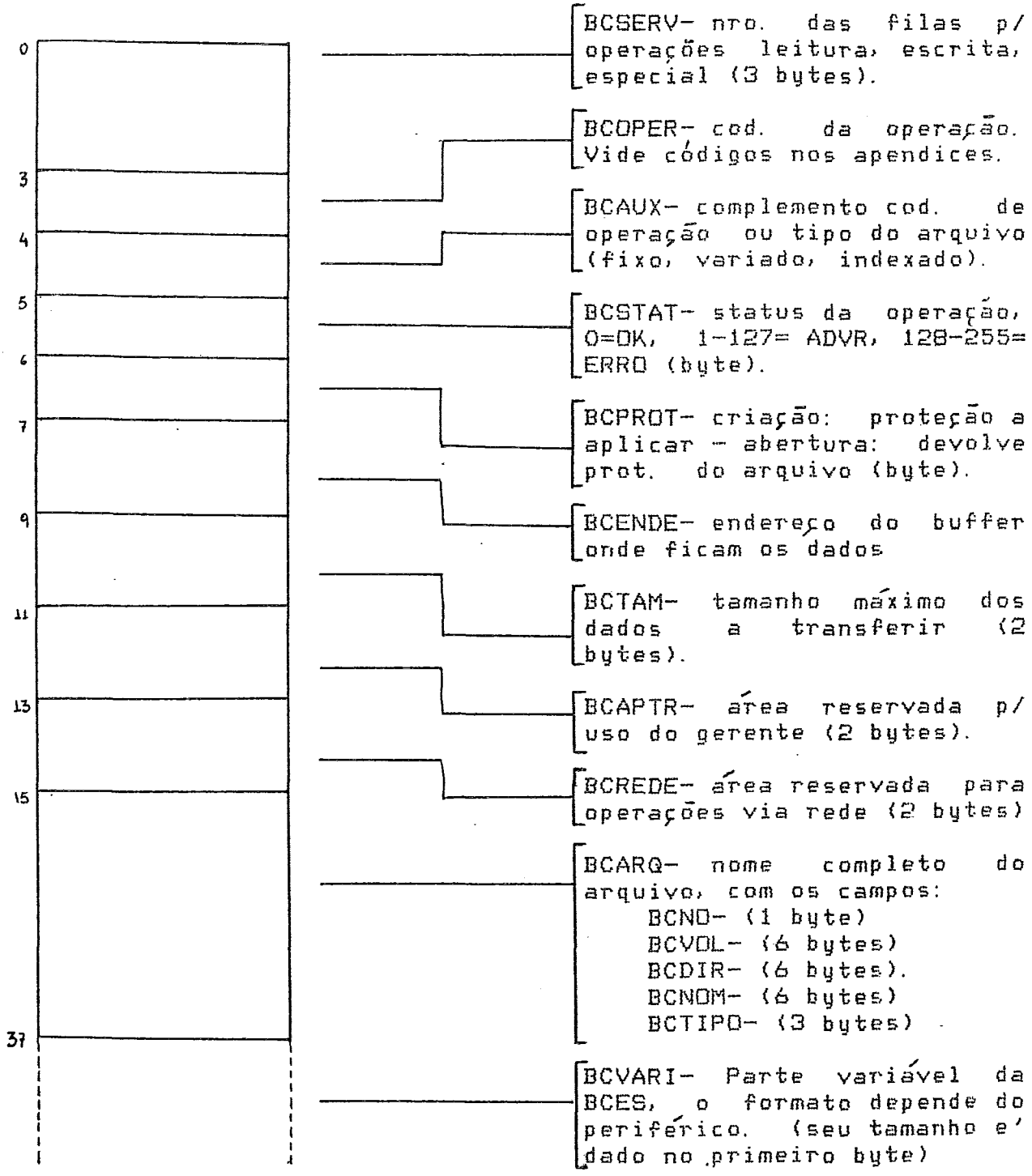


Fig 5.4 - BCES

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

endereço de um registro em um arquivo de acesso direto.

No apêndice C encontram-se os códigos a atribuir aos campos BCOPER, AUX, PROT, etc.

Cada arquivo em manipulação tem uma BCES associada. Portanto, num processo com um número "n" de arquivos em uso simultâneamente, "n" BCES devem existir.

As BCES ficam na região de memória do processo que esta' usando o arquivo. Elas devem ficar entre os endereços 16K - 62K do espaço virtual do processo.

O campo BCOPER, e' formado de 3 bytes. Cada um deles contém o número de uma fila de serviços que atendem as operações de leitura, escrita e especiais, respectivamente. Normalmente estes 3 números são iguais.

V.6 OPERAÇÕES DE E/S

O NÚCLEO oferece uma série de operações para a realização das E/S. As operações ABRA-ARQUIVO, LEIA, ESCREVA, ESPECIAL e FECHE-ARQUIVO são ditas DIRETAS por realizarem operações nos periféricos. As operações ASSOCIE, CONVERTA, DESASSOCIE e SEPARA-ARQUIVO são as AUXILIARES.

O exemplo da fig 5.5 pode ser usado como complemento ao texto que se segue.

V.6.1 ABRA-ARQUIVO

Esta e' a primeira e obrigatória operação para ter-se acesso a um arquivo. Recebe 2 parametros: o endereço de uma BCES e o endereço de uma cadeia de caracteres contendo o nome do arquivo (pode ser um nome lógico).

Antes de invocar ABRA-ARQUIVO, toda a BCES deve ser zerada e o campo BCOPER deve ser preenchido com o código de abertura apropriado. Caso seja criação de arquivo, os campos BCAUX (tipo do registro) , BCPROT devem também ser preenchidos. Estes valores estão descritos apêndice C. Arquivos com registro de tamanho fixo devem ter BCTAMA também preenchido, quando forem criados.

Caso no parâmetro 2 seja fornecido um nome lógico, e' feita a conversão para o nome de arquivo correspondente. Em seguida, este nome e' analisado, seus campos separados e colocados em BCARQ da BCES. Os campos que estiverem faltando são preenchidos

com os DEFAULTS.

Neste ponto e' feito o roteamento do pedido. Caso o no' indique que a E/S deve se dar em outra máquina, a BCES e' dirigida ao sistema de REDES. Deste modo, e' transparente e genérica a passagem dos pedidos pelo sistema de redes. Se o pedido e' para a própria máquina, a BCES e' entregue ao gerente do periférico em questão (o nome do VOLUME e' usado para determiná-lo). O gerente interpreta o código em BCOPER, abre o arquivo do modo que lhe aprouver e preenche os campos necessários na BCES, completando assim a abertura. O campo BCAUX retorna o tipo do arquivo aberto e BCPROT sua proteção.

Ao requisitar uma abertura, o processo entra OBRIGATORIAMENTE em espera pelo seu término.

```
SE nome do arquivo dado
  ENTÃO separa campos do
        nome dado.
        DEFAULTS nos campos
        que faltaram.

SE BCES.no <> desta  máquina
  ENTAO dirige requis. `a rede
  SENAO deter. nro. da fila
        a partir do nome do
        volume. (abra_servico)
        deposita BCES na fila.
```

Fig. 5.5 - Sequência de uma Abertura

Uma opção especial da abertura e' a analise externa do nome do arquivo. Para tanto, deve-se proceder como no caso anterior, e adicionalmente preencher todos os campos de BCARQ antes da abertura. Além disto, o segundo parâmetro não deve ser dado. Com este recurso, tratamentos especiais de nomes de arquivos podem ser feitos pelo processo. A Fig. acima mostra o desvio da análise interna.

V.6.2 FECHE ARQUIVO

Esta operação e' o complemento da anterior; e' bastante simples e seu único parâmetro e' a BCES. O campo BCOPER deve conter o código de fechamento com suas opções (apendice C).

V.6.3 LEIA, ESCREVA, ESPECIAL

Realizam as operações de E/S propriamente. Nos 3 casos, o único parâmetro e' o endereço da BCES, ja' inicializada por ABRA-ARQUIVO. Antes de invocá-las, o código da operação BCOPER obrigatóriamente deve ser devidamente preenchido. Os demais campos devem ser preenchidos conforme a necessidade da operação, inclusive em algumas delas, campos da parte variável devem ser preenchidos. Um exemplo e' o endereço de um registro em um arquivo de acesso direto. As operações de E/S sequenciais precisam ter os campos BCENDE e BCTAMA preenchidos.

A operação ESPECIAL deve ser usada na requisição de comandos que não transfiram dados ou que tenham características particulares. Nem todos periféricos lhes dão suporte, sendo detalhes encontrados na documentação do gerente de periférico específico [6], [3].

Estas operações podem ser pedidas com ou sem ESPERA pelo término da operação. O bit 7 de BCOPER em "1" indica com espera e em "0" sem.

Num pedido sem espera, a E/S e' disparada, mas o processo que a fez continua executando, paralelamente. A qualquer momento e' possível esperar pelo término da E/S via a operação ESPERA (vide requisições com espera, cap. comunicação entre processos). O parâmetro a passar para esta operação e' um byte retirado da BCES; se a E/S a esperar e' uma leitura, o primeiro byte de BCSERV e' o usado. Se escrita, o segundo byte deve ser passado; se especiais, o terceiro deve ser passado.

```
(* Este exemplo demonstra o uso das operações do núcleo
para E/S numa linguagem hipotética.  E' um programa
le/imprime.  *)

(* Definição das variáveis (BCES, buffers, etc.)*)

BCENTRA: estrutura do tipo BCES;
BCSAI:   estrutura do tipo BCES;
BUFFER1, BUFFER2: vetor 100 bytes;
nomearq: cadeia 30 caracteres;

(*programa*)

bcentra := 0;
bcsai   := 0;
bcentra.bcoper: = abra-leitura-sequencial;
ABRA-ARQUIVO (bcentra, 'leitor');     (*abre leitura
bcsai: = bcentra;                     (*via nome logico
bcsai.bcoper: = cria-sequencial;

SISSAI ('nome do arquivo saida?');    (*pede nome no
SISENT (nomearq);                     (*terminal
ABRA-ARQUIVO (bcsai, nomearq);

(* preenche BCES p/ leitura / escrita *)

bcentra.bctam: = 100;
bcentra.bcend: = endereço (buffer 1);
bcsai.bcend: = endereço (bufer2);
bcent.bccod: = cod-leitura+bit.7=1; (*com espera
bcsai.bccod: = cod.escrita;         (*sem espera

REPITA
   LEIA (bcent);
   bcsai.bctam: = bcent.bctam;
   ESPERA (bcsai.bcserv+1);
   buffer2: = buffer1;
   ESCREVA (bcsai);
ATE' bcent.satatus = fim.arquivo;

FECHE (bcsai);
FECHE (bcent);
FIM.
```

Fig 5.6 - EXEMPLO DE OPERAÇÕES DE E/S

V.6.4 OPERAÇÕES AUXILIARES

ASSOCIE - atribui a um nome lógico uma cadeia de caracteres. Para as E/S, esta cadeia deve ser o nome de um arquivo. A associação pode ser a nível de processo ou a nível de sistema; no primeiro caso deve-se especificar o nome do processo no segundo parâmetro e no segundo caso colocar um "$" neste parâmetro.

CONVERTA - dado um nome lógico, a cadeia a ele associada e' devolvida. Primeiro e' verificado a nível de processo se existe a associação, e em seguida a nível de sistema. Para forçar a busca somente a nível de sistema, um "$" deve ser dado no segundo parâmetro. Abra-arquivo invoca internamente esta operação.

DESASSOCIE - o inverso de associe, podendo optar pela desassociação de somente um nome lógico ou de todos pertecentes a um processo. O parâmetro "nome lógico", se não dado, força a desassociação total do processo.

V.6.5 SISENT E SISSAI

A operação SISENT lê dados do arquivo associado ao nome lógico SISENT. Como único parâmetro necessita do endereço de uma cadeia onde serão depositados os dados.

A operação SISSAI e' análoga, so' que escreve os dados de uma cadeia dada como parâmetro no arquivo associado ao nome lógico SISSAI.

A vantagem de usar estas operações e' a simplicidade. Não são necessárias BCES, abertura de arquivos, etc. Isto tudo e' feito internamente.

Os nomes lógicos podem ser associados a qualquer periférico que aceite acesso sequencial, como por exemplo arquivos em disco, fita, vídeo, teclado, impressora etc.

CAPITULO VI

GERENTE DE MEMÓRIA

Neste capítulo, são apresentados os recursos oferecidos pelo sistema e as características de HARDWARE suportadas.

VI.1 MEMÓRIA FÍSICA

E' destinado a processadores com espaço de endereçamento até' 64K bytes (16 linhas de endereço), expandidos externamente por hardware de memória paginada. As páginas devem ter 2K bytes de tamanho. Suporta proteção por HARDWARE contra escrita, contra leitura ou ambos, independente por página.

São suportados pelo NÚCLEO até 8 Mbytes de memória real do tipo RAM, embora na presente versão da máquina o máximo a que se pode atingir seja 3.5 Mbytes. Também existe uma memória PROM, usada principalmente na fase de carga e inicialização do sistema. E' composta de até 16K, sendo porém o típico 2K.

A memória RAM pode ser dividida em 2 partes: memória de vídeo e memória comum. A memória de vídeo e' onde devem residir dados a exibir na tela; suas demais características são idênticas à memória comum, podendo inclusive receber dados e programas normalmente. Se um pedido de alocação de memória especifica "memória de vídeo", somente nela sera' feita a busca. Se nada e' especificado, a busca pelo espaço se da' primeiro na memória comum e depois na memória de vídeo.

As interfaces com periféricos funcionam baseadas no principio de "E/S mapeados em memória" (memory mapped I/O), isto e', acessa-se-lhes como se fossem memória RAM comum. São suportados até' 6 interfaces, cada uma podendo dispor de até' 2K bytes de região de comunicação. Além disto, outros 2K são usados pelos registros de controle da máquina e pelos periféricos oferecidos na configuração básica (teclado e duas linhas seriais).

Cada processo tem seu espaço de endereçamento exclusivo, isto é, pode-se referir a endereços entre 0 e 64K. Estes endereços são chamados ENDEREÇOS VIRTUAIS do processo. É dividido em 32 janelas de 2K bytes cada (janela 0 a janela 31). Cada uma destas janelas pode estar mapeada em qualquer um dos ENDEREÇOS FÍSICOS disponíveis (RAM, PROM ou E/S). Mais de um processo pode estar usando um mesmo endereço físico simultaneamente. O exemplo a seguir mostra 2 processos, seus espaços virtuais e o mapeamento na memória física.

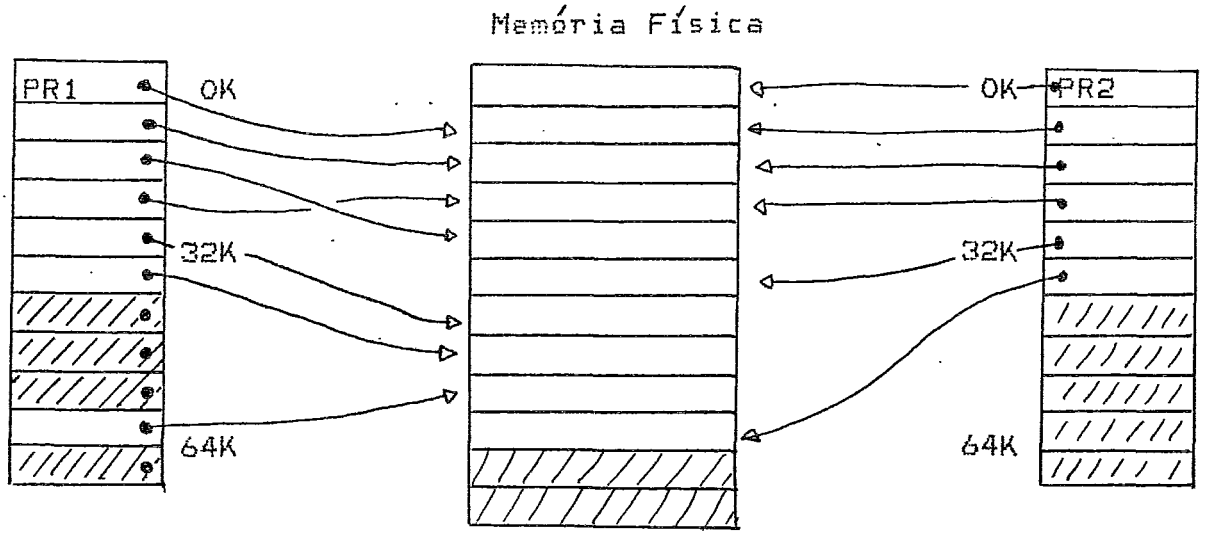


Fig. VI.1 - espaço virtual x endereço físico

O NÚCLEO provê a operação MAPFIS, que permite ao processo mapear uma de suas janelas 0 a 30 sobre qualquer endereço físico. Além dela, existem várias outras operações referentes à mapeamento, e serão descritas adiante ou no apêndice D.

VI.2 PÁGINAS DOS PROCESSOS

De modo a não limitar o tamanho dos processos a 64K, foi criado o conceito de páginas, descrito a seguir e na figura VI.2.

Um processo é composto por PÁGINAS, cada uma com tamanho de 2K bytes. Na atual versão do sistema, o número máximo de páginas por processo é 512, numeradas de 0 a 511. Assim, o tamanho máximo de um programa fica em 1 Mbyte.

O tamanho das páginas coincide com o das janelas, de modo a permitir que o programa possa "ver" (acessar) as suas páginas através destas janelas. Cada página tem a janela (endereço virtual) que lhe corresponde; porém, páginas com código PIC (Posicion Independent Code) podem ter este endereço virtual (número da página) modificados em tempo de execução, conforme descrito no tópico ativação de páginas.

Nem todas as páginas precisam estar na memória durante a execução do programa; podem ficar em memória auxiliar de acesso direto, sendo trazidas na medida que o processo as requisite. No tópico gerência do espaço físico, a seguir, e' detalhada a política de alocação de memória física.

Cada página pode ter os seguinte atributos:

1. número da página default - da' o endereço virtual pelo qual a página sera' acessada pelo programa. Pode ser alterado na ativação da página (tópico a seguir).

2. memória de VÍDEO - memória alocada obrigatoriamente em memória de vídeo

3. memória COMUM - memória alocada preferencialmente em memória comum; caso não haja, e' aloca em memória de vídeo.

4. não modificáveis - são páginas que normalmente só permitem leitura de seu conteúdo. Podem ou não ser protegidas contra escrita por hardware. Em multiexecução, são compartilhas pelas varias cópias dos programas.

5. modificável - permite escrita/leitura de seu conteúdo. Podem ser criadas dinâmicammente.

6. permanentes na memória - não e' sucetível a ser desalojada da memória física.

7. permanentemete ativa - esta' sempre ativa (tópico a seguir).

VI.2.1 CRIAÇÃO DE PÁGINAS

As páginas podem ser criadas na confecção do programa pelo compilador/montador ou dinamicamente durante a execução do programa. Para criá-las dinamicamente, o programa deve invocar a operação DEFPAG, especificando os números de páginas a criar, as janelas sob as quais serão mapeadas e seus atributos. A memória física para estas páginas pode ou não ser alocada automaticamente neste instante. Para acessar estas páginas, deve-se ativá-las primeiramente (tópico a seguir).

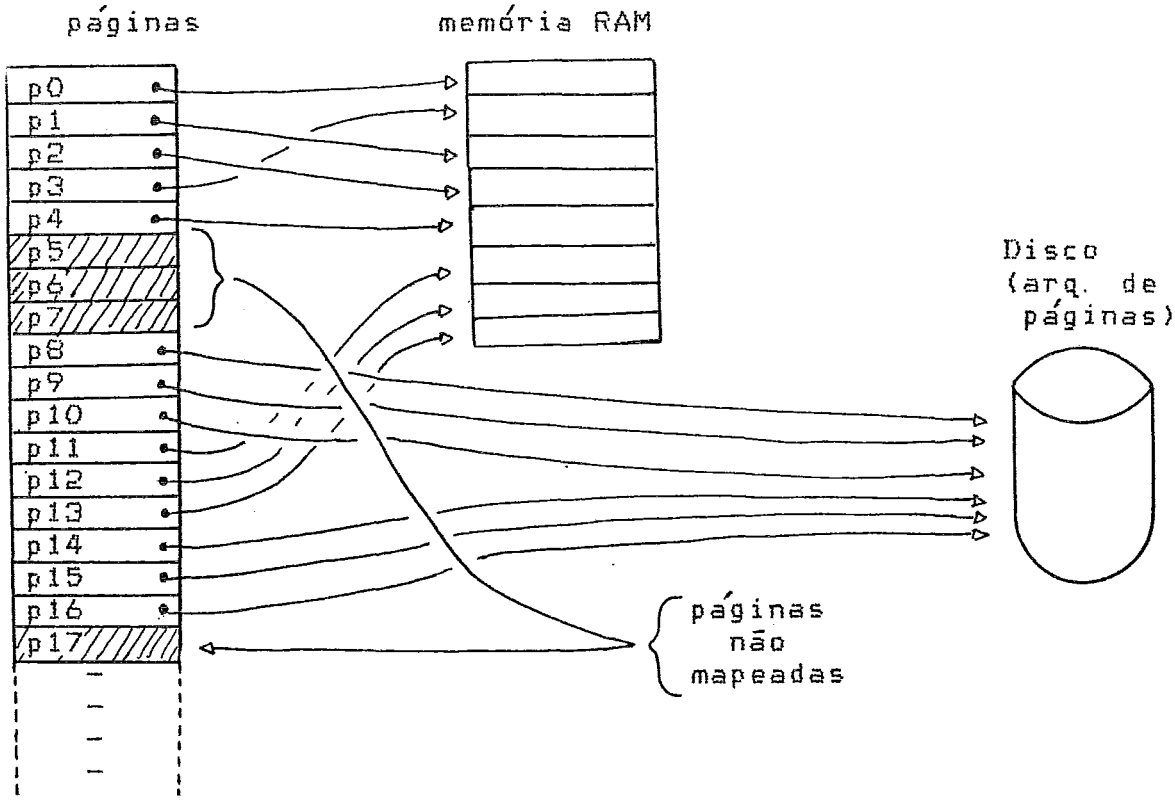


Fig. VI.2 - PÁGINAS DE PROCESSOS

VI.2.2 ATIVAÇÃO DE PÁGINAS

E' facil perceber que num processo maior que 62K, somente algumas de suas páginas podem estar ATIVAS em determinado momento, isto e', as janelas estarão mapeadas sobre os endereços físicos onde se encontram estas páginas ativas. As demais páginas podem estar na memória RAM ou na memória secundária.

Uma página ou conjunto de, podem ser ATIVADAS a qualquer instante pelo processo via chamada 'a operação ATVPAG do NÚCLEO. Se elas estiverem na memória, e' feito o mapeamento das janelas que lhe correspondem ; (fig. VI.3) naturalmente, havendo páginas ativas sob as mesmas janelas, estas são DESATIVADAS. Encontrando-se algumas ou todas as páginas a ativar em memória secundária, são trazidas para a memória principal e so' então o mapeamento e' feito.

Notar que para acessar o conteúdo de uma pagina, a mesma deve estar necessariamente ativa.

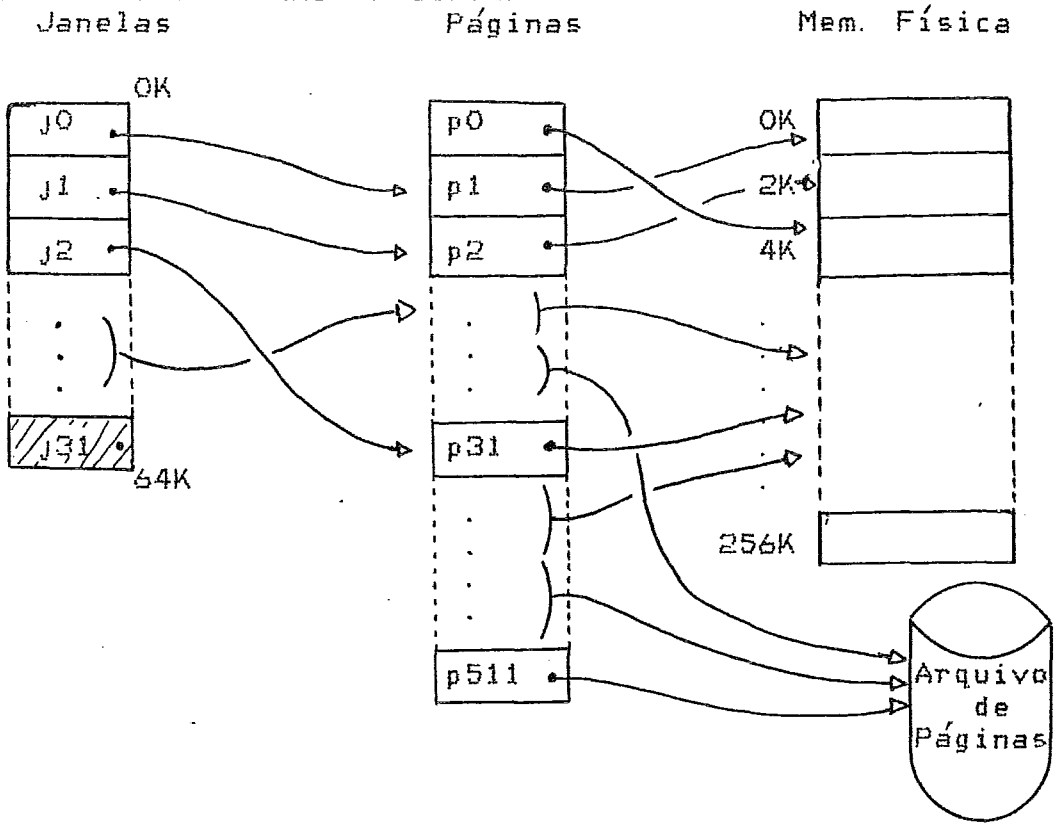


Fig. VI.3 - JANELAS, PÁGINAS E MEM. FÍSICA

Em programas SIMPLES, todas as páginas que o compõem são ativadas no momento da carga do programa. Em programas COMPLETOS, deve-se especificar (na sua confecção) as páginas que serão ativadas no momento da carga. As outras páginas deverão ser ativadas explicitamente pelo programa ao necessitá-las. Normalmente os compiladores geram código que tornam transparente estes detalhes.

VI.2.3 ALOCAÇÃO/DEALOCAÇÃO DE MEMÓRIA

Durante sua execução, um processo pode alocar ou dealocar memória necessária para uma ou mais páginas. Isto pode ser feito pelas operações ALOMEM e DLOMEM do NÚCLEO. Com isto, áreas de dados temporários gastam memória somente, quando necessário. A área liberada por DLOMEM torna-se acessível a outro processo ou ao próprio, quando necessária.

Uma mesma região de memória RAM pode ser alocada ao mesmo tempo por vários processos. Este recurso e' usado principalmente em processos com código ou dados compartilhados. Este tipo de alocação

e' feita por chamadas 'a operação ALOCOM. Somente quando todos os processos que usam uma região multialocada a liberarem, a mesma e' físicamente liberada.

VI.2.4 MAPEAMENTO ENTRE PROCESSOS

Um processo pode mapear temporáriamente janelas (endereços virtuais) sobre outro processo. Podem ser quaisquer janelas entre 8 e 30 (16K a 62K). Para tanto deve ser usada a operação MAPVIR, que recebe como parâmetros:

1. o nro. da janela a mapear - este sera' o endereço pelo qual o processo requisitante fara' o acesso aos dados no outro processo.

2. endereço virtual no processo destino - endereço virtual (0 a 62K) dos dados no processo que se quer acessar.

3. ID do processo - indica o processo sob o qual se fara' o mapeamento.

Cada chamada a MAPVIR mapeia 2 janelas (4K) sobre o processo. O endereço dado no segundo parâmetro e' convertido em relaçao ao endereço dado no primeiro parâmetro. Isto permite o acesso direto aos dados, pelo processo requisitante.

Este mapeamento e' razoavelmente rápido e e' o modo utilizado para a comunicação entre processos; um processo informa ao outro onde estão os dados (normalmente via fila de serviços), que então mapeia suas janelas sobre estes dados.

VI.3 GERÊNCIA DO ESPAÇO FÍSICO

Quando e' necessário memória RAM comum, o NÚCLEO recebe este pedido e segue a sequência de tentativas abaixo, ate' que uma delas permita que o espaço seja conseguido.

1. pega memória livre.

2. tira páginas não ativas e não modificáveis da memória do processo que fez o pedido, ate' conseguir o espaço ou ate' chegar 'a área de trabalho (WORKSET) mínima do processo.

3. tira páginas não ativas mas modificáveis do processo que fez o pedido, ate' conseguir o espaço ou atingir o WORKSET mínimo. Elas são copiadas para o arquivo de páginas mantido pelo sistema.

Caso não tenha conseguido, e' retornado o codigo "não ha' memória disponível" e a alocação não e' feita. Esta' previsto em futura versão a continuação da busca, seguindo os itens abaixo,

1. repete os itens 2 e 3 anteriores, so' que sobre os demais processos em execução.

Nesta versão do sistema, a escolha da página a tirar sera' feita via uma funcão randômica. Esta' previsto o aperfeiçoamento desta técnica em versão futura pelo uso de um esquema LRU.

VI.4 PROCESSOS SIMPLES

São compostos de 2 partes: a do programa e o da PILHA + NÚCLEO. O tamanho da parte 1 e' determinado pelo carregador a medida que vai lendo o arquivo do programa. A parte 2, da PILHA, e' determinado conforme visto no cap. II. E' formado por páginas nao retiráveis da memória e sem proteção contra escrita.

CAPITULO VII

CARREGADOR/TERMINADOR

Os conceitos apresentados no cap. III são extensivamente usados na exposição que se segue.

Neste capítulo serão apresentados os módulos responsáveis por iniciar e terminar a execução de um processo.

VII.1 CARREGADOR

A partir de uma solicitação de um processo, o CARREGADOR faz uma série de verificações para determinar se um processo deve ser realmente criado e se um programa, somente algumas de suas páginas ou nenhuma delas devem ser trazidas para a memória. As decisões são baseadas nas seguintes observações:

1. Se o programa esta' num processo na memória, e' reexecutável e esta' desativado, a única providência e' reativá-lo. Portanto, não ha' criação de processo ou carga de programas.

2. Se o programa ja' tem uma cópia executando num processo na memória, um novo processo e' criado, as páginas não modificáveis são compartilhadas com as da cópia ja' existente, as páginas modificáveis não criadas dinamicamente são trazidas do arquivo onde esta' o programa.

3. Não estando o programa ainda na memória, um processo e' criado e todas as páginas são trazidas do arquivo.

As requisições ao CARREGADOR são feitas via depósitos na fila de serviços de nome "CARREG". Os dados a passar seguem a estrutura apresentada abaixo:

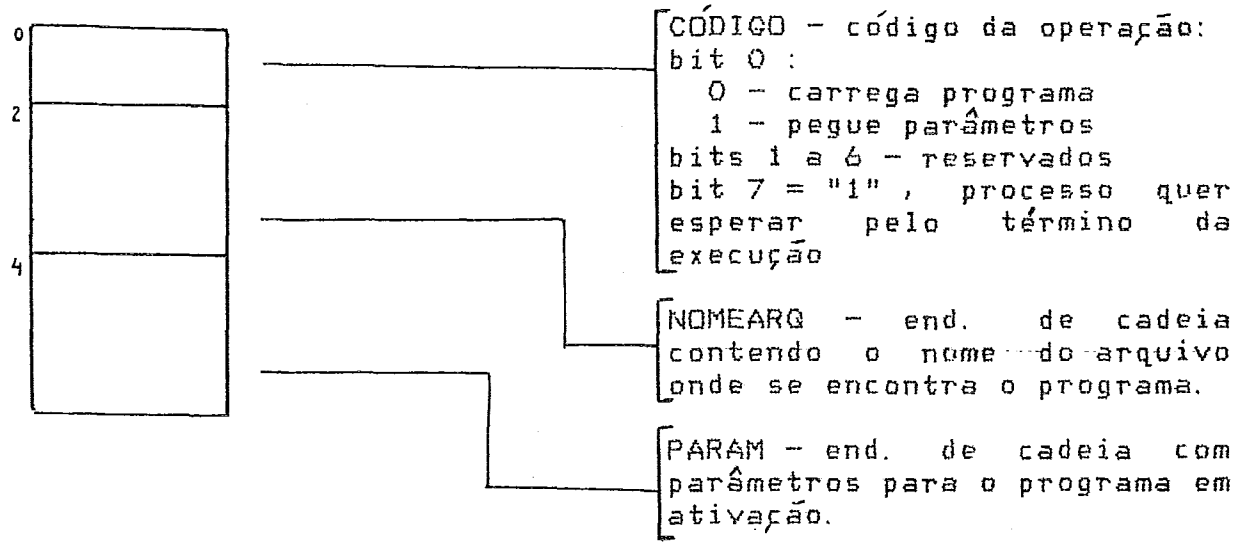


Fig 7.1 - REQUISIÇÃO AO CARREGADOR

Ao fazer a requisição ao CARREGADOR, o processo pode indicar que quer entrar em ESPERA ate' o término da execução do programa que esta' ativando. Ao terminar a execução, automaticamente o processo que estava em espera volta a executar. Com este recurso e' possível o sincronismo automático entre o processo ativador e o ativado. O bit 7 do código deve ser usado p/ esta opção, e o depósito deve ser feito com espera.

Os campos de NOMEARQ que estiverem faltando são preenchidos com os defaults do processo que pediu a carga.

Parâmetros podem ser passados para o processo em ativação. Para tanto o campo PARAM deve ser preenchido com o endereço de uma cadeia que os contenha. Por sua vez, o programa carregado deve fazer uma requisição ao carregador, com o código "pegue parâmetros", e o endereço de uma cadeia que recebera' os parâmetros. Neste caso, não e' necessário preencher o campo NOMEARQ.

VII.1.1 BIBLIOTECAS

O CARREGADOR e' o responsável por manter as bibliotecas ou parte das mesmas na memória, bem como mapear páginas dos programmas que esta' carregando sobre as mesmas.

No espaço virtual do processo CARREGADOR ficam as páginas das bibliotecas que estejam sendo referenciadas por um ou mais processos; portanto, cada processo que faça referência 'as estas partes de bibliotecas têm algumas de suas páginas mapeadas na mesma região física que o CARREGADOR. Isto implica em somente uma cópia de código comumente usado ser compartilhado por todos os processos.

Todas as páginas de bibliotecas são fixas na memória, não sendo suscetíveis à remoção para o arquivo de páginas.

VII.2 ARQUIVOS DE PROGRAMAS

São dois os tipos de arquivos para o armazenamento de programas: para os simples e para os completos. O tipo do arquivo e' indicado pelo primeiro byte do mesmo; se for diferente de zero, e' arquivo simples e sendo zero indica arquivo completo.

VII.2.1 ARQUIVOS SIMPLES

Basicamente são 2 os tipos de arquivos simples: o formato "S1" padrão da MOTOROLA, em ASCII e o formato binário. Ambos podem ser armazenados em qualquer meio que suporte acesso sequencial (disco, fita, linha de comunicação, etc.); Pode-se dar entrada em programas no formato "S1" via teclado, uma vez que e' ASCII. Segue o formato de cada registro "S1".

S1CCAAAAdddd....ddKK

onde:

S1 - 2 caracteres ASCII, "S" e "1", sendo o cabeçalho de cada registro

CC - número hexadecimal de 2 dígitos, da' o número de bytes do registro (também inclui nesta contagem 2 para o endereço e um para o checksum).

AAAA - número hexadecimal de 4 dígitos que da' o endereço de carga do registro.

ddd....dd - e' uma série de pares hexadecimais, cada um representando um byte

KK - número hexadecimal de 2 digitos que da' o checksum. E' calculado somando-se todo o registro, com exceção de "S1". O "vai 1" (carry) e' desprezado na soma.

Entre os registros pode haver quaisquer caracteres ASCII diferentes de "S". Em particular <CR>, <LF> podem ser adicionados para facilidade de leitura.

Exemplo
S107100CFF00EA

O endereço do início da execução deve ser dado no último registro, que deve ter tamanho (CC)=3, isto é, não deve ter dados. Caso este registro não seja dado, é usado o endereço dado no primeiro registro carregado.

A marca de fim-de-arquivo é dado por um registro contendo "S9" ou pela detecção de fim de arquivo.

Fig 7.2 - FORMATO "S1".

O formato binário é semelhante ao formato "S1", só que mais compacto (gasta a metade do espaco de "S1"). A desvantagem é estar em forma binária, não sendo legível/digitável diretamente pelo usuário. Segue seu formato:

. SYNC. TAMANHO. ENDEREÇO. dados. . . . dados. CHKSUM

onde

SYNC - 1 byte contendo o valor $16

TAMANHO - 1 byte, que contém o tamanho do registro

ENDERECO - 2 bytes, contém o endereço de carga do programa.

DADOS - são os bytes a depositar na memória

CHKSUM - byte, é o checksum do registro, com exceção do caracter SYNC.

Entre os registros podem haver quaisquer caracteres diferentes de SYNC e EOT. O endereço de execução deve ser dado no último registro, que deve ter tamanho = 3 (não contém dados). O fim de arquivo é dado pelo codigo EOT ($04) ou pela detecção de fim de arquivo.

Fig 7.3 - FORMATO BINÁRIO

VII.2.2 ARQUIVOS COMPLETOS

Só podem residir em meios de acesso direto, têm registros de tamanho fixo (256 bytes) e são compostos de 2 partes:

1. O descritor de páginas, onde se encontram as informações que permitem ao CARREGADOR carregar cada página. Além disto, aí estão informações gerais sobre o programa, tais como página e endereço início de execução, páginas que devem ser criadas como área de trabalho, lista de bibliotecas e que páginas as referenciam, etc.

2. As páginas propriamente ditas, so' existindo aquelas que têm código ou dados inicializados. As páginas de rascunho (que contêm matrizes, variáveis não inicializadas) são criadas dinamicamente no momento da carga do programa.

A ref. [19] detalha o formato de um arquivo completo.

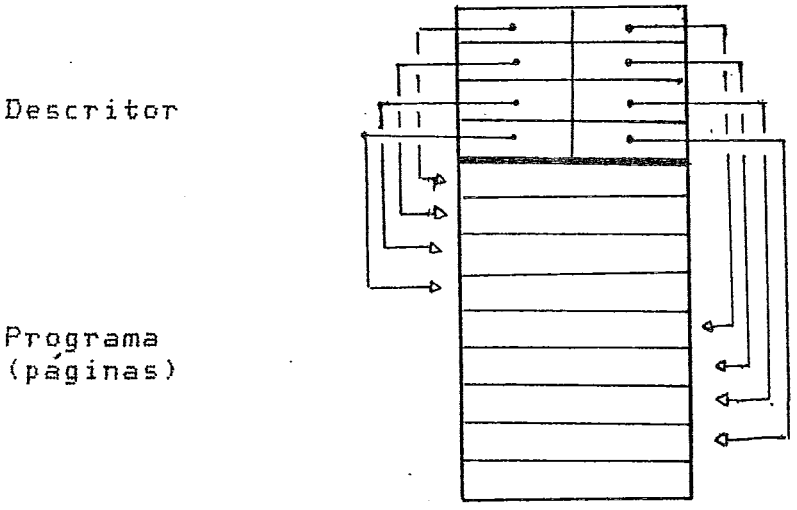


Fig 7.4 - ARQUIVO COMPLETO

VII.3 TERMINADOR

O terminador faz parte do NÚCLEO do sistema. Sua função e' terminar de modo ordenado a execução de um processo, liberando os recursos alocados ao mesmo. Eventualmente a memória ocupada também e' liberada.

A liberação de recursos se da' via aviso a cada servidor que esteja ou que ja' esteve atendendo a requisições do processo que o mesmo esta' em fase de término. Daste modo, os servidores podem liberar áreas, fechar arquivos, etc. relacionados ao processo.

Para tanto, o TERMINADOR coloca uma requisição com valor zero em cada fila de serviços que o processo acessou. Quaisquer requisições de processo pendentes nestas filas, são eliminadas.

A liberação de memória e consequente extinção do processo se da' sempre que o processo não for reexecutável. Nos reexecutáveis, o processo e' desativado, suas páginas modificáveis liberadas, sendo então colocado em uma fila LRU, para eventual posterior execução.

O terminador e' invocado por uma chamada a operação TERMINE do NÚCLEO. Pode terminar o próprio processo (término normal) ou outro qualquer, (cancelamento assíncrono). O comando SIRIUS SUSPENDA chama esta operação.

Nos descritores dos processos, existe um campo que indica se o processo e' permanente ou não. Caso afirmativo, o TERMINADOR verifica se e' um auto término (o processo terminando a si próprio), sendo então a sequência seguida normalmente. Sendo um cancelamento assíncrono, o procedimento e' abortado, não sendo, pois, cancelado. Deste modo, um processo permanente não pode ser afetado por outro.

Cabe também ao TERMINADOR, quando necessário, a ativação do processo que esta' a espera do término desta execução (conforme visto no carregador).

CAPITULO VIII

SISTEMA DE REDES

O sistema de redes [1] e' composto por processos que executam 2 funções básicas: a primeira e' entregar pedidos e receber respostas da rede; a segunda e' atender 'as requisições provenientes da rede. de forma assíncrona e transparente ao resto do sistema que esteja executando na máquina.

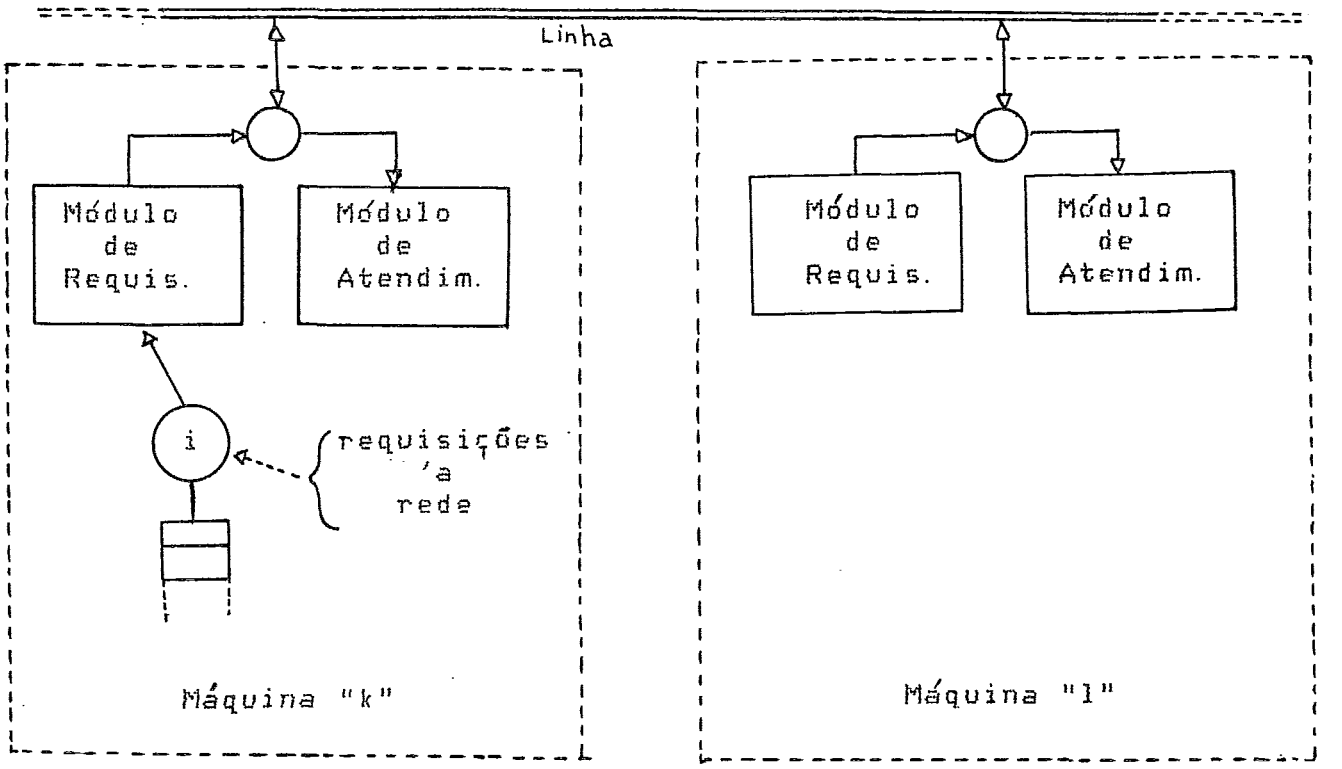


Fig 8.1 - O SISTEMA DE REDES

Neste capítulo, apenas sera' feita uma descrição suscinta do sistema de redes, e maiores detalhes podem ser vistos no trabalho especificado na referência [1].

As operações do sistema de redes podem ser divididas em 2 grupos: suporte a E/S remotas e suporte a operações do NÚCLEO remotas.

VIII.1 E/S REMOTAS

Qualquer operação de E/S pode ser feita remotamente, totalmente transparente ao processo que a fez. A determinação do roteamento dos pedidos de E/S a um arquivo, são feitos no momento da abertura do mesmo, pela operação ABRA-ARQUIVO. Na máquina no outro lado da rede, que atenderá os pedidos, tudo também é transparente, pois mesmos os gerentes de periféricos não necessitam saber que o pedido veio pela rede (embora a BCES tenha um campo que indica isto, para permitir proteção).

VIII.2 OPERAÇÕES DO NÚCLEO REMOTAS

As operações do NÚCLEO dadas na lista a seguir podem ser invocadas remotamente. Isto é feito com o auxílio da operação OPER-REM, que roteia operações ao sistema de rede. A operação roteada será executada na máquina destino, sujeita aos parâmetros de sistema lá existentes (tabela de nomes lógicos, hora local, filas de serviço, etc.). Porém, qualquer mensagem decorrente da mesma, será roteada à máquina origem.

Ao invocar uma operação remota (OPER-REM), o processo entra automaticamente em espera pelo término da mesma.

1. ABRA-SERVICO
2. ACERTE (2)
3. ASSOCIE (1)
4. ATIVE (1)
5. CONVERTA (1)
6. DATA
7. DEFAULT (2)
8. DEPÓSITO (1)
9. DESASSOCIE(1)
10. ESPERA (1)
11. HORA
12. TERMINE (1), (3)

OBSERVAÇÕES:

(1) - tem as características fundamentais do processo transferido , p/ validação do pedido.

(2) - somente processo ou usuários privilegiados podem

invocar esta operação.
(3) - somente se o processo pertencer ao mesmo usuário (mesma IDUS) ou for privilegiado.

Os parâmetros para a operação a executar devem estar numa lista que e' igual 'aquela usada em execuções locais da operação. Adicionalmente deve ser dada uma segunda lista, que no primeiro byte contém o nro. da máquina destino, nos segundo e terceiro o endereço da rotina a executar e no quarto o nro. de parâmetros da operação. Nos bytes subsequentes devem ser dados os tamanhos de cada parâmetro dado na primeira lista.

VIII.3 IMPLEMENTAÇÃO DA TRANSFERÊNCIA

A princípio, quando um processo faz uma requisição, vários parâmetros do seu descritor de processo são passados 'a rede. Assim, na outra máquina, o processo de redes que atende às requisições, assume temporariamente estas características, simulando o processo que fez a requisição. Tudo então se passa como se o pedido estivesse sendo feito na própria máquina.

CAPITULO IX

CONCLUSÕES

E' apresentado um rápido histórico, de modo a mostrar o tempo gasto em cada uma das etapas do projeto. Notar que na maior parte do projeto ou pelo menos na sua fase mais importante, as pessoas que nele trabalhavam o fizeram no tempo livre que tinham (noites e fins de semana).

IX.1 HISTÓRICO

As idéias de se fazer um sistema genuinamente nacional remontam a 1979, mas somente em 1980 o grupo de 3 pessoas (o autor, Albert Besso e Marcos Cosendey) começaram a esboçar o HARDWARE e o SOFTWARE do sistema, em sua primeira versão.

Em 1981, a definição geral do sistema e seu NÚCLEO, se tornaram o objetivo desta tese, orientado por Suely Mendes. Paralelamente outros trabalhos começaram a ser desenvolvidos ([1] , [3] e outros).

Em meados de 1982, ja' com os algoritmos do NÚCLEO escritos, levei 2 meses codificando-os em ASSEMBLER do micro MC6809. De setembro em diante, começou a depuração do NÚCLEO, e o trabalho da redação deste texto em sua forma final. Nesta 'epoca o HARDWARE passou por uma grande mudança estrutural, (versão II) o que facilitou muito a programação do sistema

Em maio de 1983 foi redefinido gerente de memória, o que acarretou algumas mudanças no resto do sistema e em toda a documentação.

IX.2 DEPURAÇÃO

A depuração foi feita no micro computador COLOR COMPUTER, que usa o processador MC6809. Tínhamos um editor de textos, montador assembler e depurador simbólico. O meio de armazenamento de programas era gravador K7. Este computador foi comprado

particularmente com este objetivo.

A maioria das operações do NÚCLEO, conforme apresentadas no apêndice D foram depuradas. A principal exceção foram as operações da gerência de memória, por serem as mais dependentes de HARDWARE e o COLOR não ter circuitos de relocação/paginação de memória.

De um modo geral, o pequeno nível de problemas encontrados na lógica durante a depuração, surpreendeu. A princípio, nao ha' registro de nenhuma mundança maior na lógica. A maior taxa de erro ficou por conta de uso múltiplo de registradores, falta de inicialização de variáveis, etc.

A geração de eventos externos (interrupções) foi a parte da depuração que apresentou maiores problemas, devido a falta de documentação do COLOR neste sentido. Assim, poucos testes foram feitos com acontecimento paralelo de eventos externos, e somente em algumas partes do código.

Registre-se que o financiamento de protótipo em HARDWARE do sistema proposto (em fase de montagem) foi todo financiado (bem como o COLOR) pelas 3 pessoas citadas anteriormente, num custo estimado em US$ 6.000.

IX.3 MEDIDAS/COMPARAÇÕES

Foram feitas medidas das partes do código que mais impactam na performance do sistema. Estes valores mostraram-se bastante satisfatórios, quando comparados a outros sistemas [5], [25].

A seguir são apresentadas duas tabelas. A primeira compara as medidas obtidas com algumas encontradas no sistema VAX/VMS - 780 e com a do sistema SOLO. A segunda apresenta as características gerais do sistema com similares encontradas ou não em outros sistemas.

Convem notar que as comparações nao levam em conta o porte do sistema. Assim, o VAX e' uma máquina várias vezes mais rápida e potente que o nosso micro, o PDP-11 (SOLO) e' da mesma faixa de performance e o CP/M destina-se a máquinas bem inferiores.

TABELA I : PERFORMANCE

Operação	Sistema CARCARÁ	VAX	SOLO	Unid.
Mudança de contexto de processo	250	328/678	600	µs
E/S (passagem do pedido ao sistema) (DEPÓSITO/RETIRADA/ FINALIZA)	1.000	1.287	?	µs
Ativação de página na memória	300*	393/638	---	µs
Preempção	100*	?	?	µs

- * indica valores estimados a partir do código escrito; não foram medidos.
- --- indica característica não existente
- ? indica valor não encontrado na documentação disponível.

TABELA II: CARACTERÍSTICAS

Característica	Sistema CARCARÁ	VAX	UNIX	CPM
Quant. mínima de memória real	64K	1Mb	256K	48K
Suporte a REDES intrinseco	Sim	Sim	Nao	Não
Multiprogramação	Sim	Sim	Sim	Não
Estrutura de "pipes"	Sim	Sim	Sim	Não
Fácil de usar	Sim	Sim	Nao	Não
Memória virtual	Sim	Sim	(1)	Não
Acesso a volumes por seu nome	Sim	Não	Sim	Não
Gerência de periféricos fora do NÚCLEO	Sim	Não	Não	Sim
Escrito em linguagem de alto nível	Não	Não	Sim	Não

(1) - depende da implementação e da máquina que a abrigará.

APÊNDICE A

ESCALONAMENTO/PREEMPÇÃO

E' utilizado um sistema de PILHA para o escalonamento a curto prazo. Este método, desenvolvido neste trabalho, provê automaticamente uma prioridade maior aos processos com muita E/S e menor aos com muito uso de CPU. Por outro lado, como a maioria dos processadores possui embutido no HARDWARE instruções para manipulação de pilhas, e' muito eficiente o método.

A distribuição das prioridades e' probabilística e não determinística. Portanto, existe a PROBABILIDADE de um programa base E/S ter uma prioridade maior que um base CPU. Além disto, esta distribuição de prioridades e' constantemente avaliada, permitindo que uma mudança na relação "E/S x CPU" de um processo se reflita rapidamente em sua prioridade. Nos tópicos seguintes e' apresentado o método.

A.1 ESCALONAMENTO

O funcionamento do escalonamento e' detalhado nos itens abaixo. A figura a seguir da' um exemplo da pilha dos processos 'a espera de CPU. Cada entrada e' composta pela ID do PROCESSO.

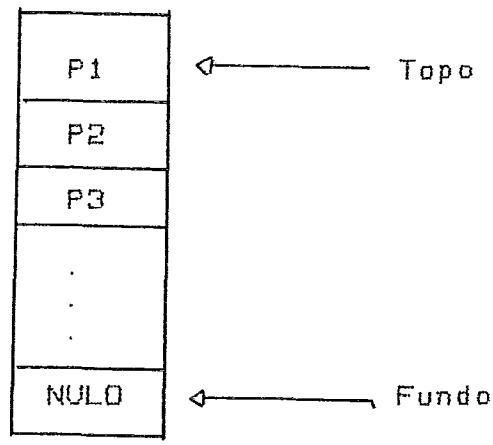


Fig A.1 - A PILHA DE EXECUÇÃO

1. O processo apontado pelo "TOPO", e' aquele que no momento esta' usando a CPU. Os demais, entre o "TOPO" e o "FUNDO" estão 'a espera de CPU.

2. O processo que esta' no TOPO, ao fazer uma operação que necessite entrar em espera (E/S, evento externo ou relógio) sai da pilha, indo para a fila de requisições ao evento. Passo seguinte, "TOPO" e' decrementado, e o processo imediatamente abaixo passa a ser executado. Este procedimento pode se repetir ate' chegar ao FUNDO da pilha, onde esta' o processo NULO. Este e' um processo que basicamente fica em LOOP eterno (eventualmente testando a máquina).

3. Ao terminar a espera pelo evento, o processo volta para a pilha, sendo colocado no seu TOPO, voltando portanto a executar. Naturalmente, o processo que estava executando e' interrompido.

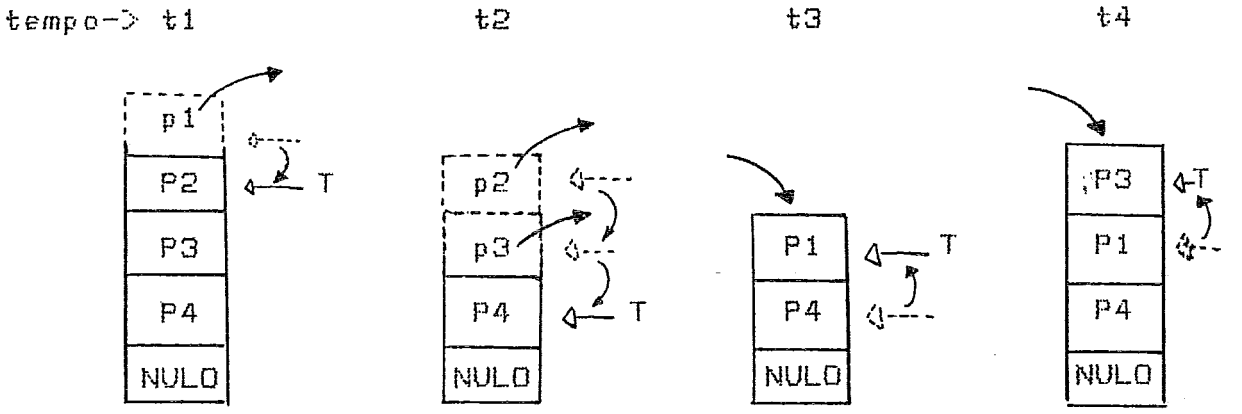


Fig A.2 - ESCALONAMENTOS

A figura acima exemplifica a explicação dada.

E' facil notar que um processo "A" com muita E/S pouco ficara' na pilha, pois esta' nas filas de espera a maior parte do tempo. Ficando pouco no Topo, a probabilidade de ter sua execução interrompida e' pequena, se comparado com um processo "B" de muita CPU, que normalmente esta' na pilha. Este efeito pode ser traduzido em prioridade, uma vez que "A" tem alta probabilidade de não ser interrompido quando em uso de CPU e "B" tem baixa probabilidade de não ser interrompido.

Porém, por exemplo, se um processo de muita E/S ao voltar a executar se tornar 100% CPU, os processos "abaixo" do TOPO não mais executarão. Torna-se necessário um mecanismo adicional para resolver este tipo de problema que e' a PREEMPÇÃO.

A.2 PREEMPÇÃO

Consiste de a cada intervalo T (parametrizável) trocar-se o processo que esta' no TOPO com um outro "abaixo". A escolha do "abaixo" na presente versão segue uma distribuição uniforme.

A implementação foi feita com um ponteiro auxiliar, TROCA.

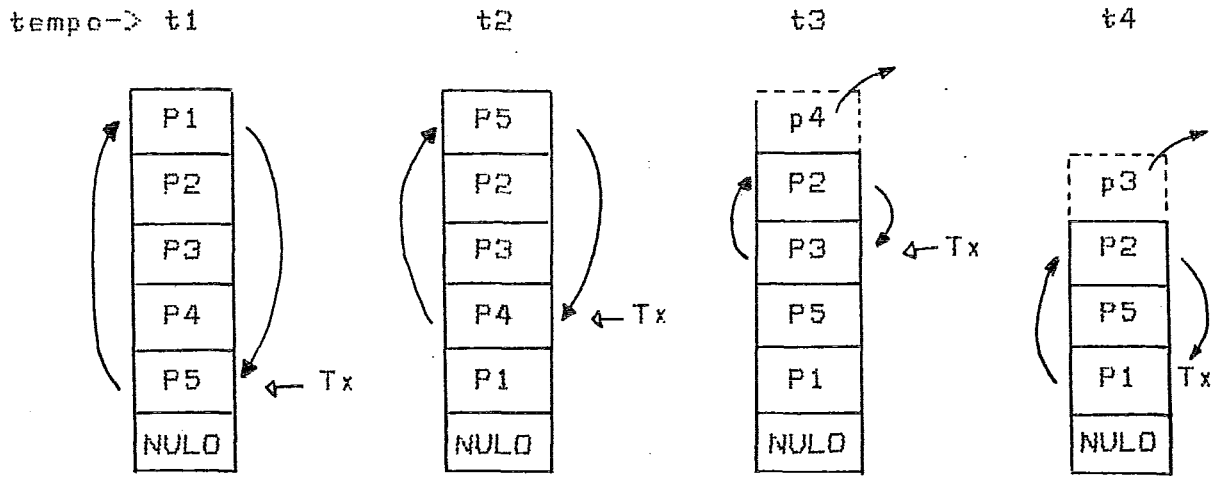


Fig A.3 - PREEMPÇÕES

A cada intervalo t, o processo do TOPO troca de posição com o apontado por TROCA; em seguida TROCA e' deslocado de uma unidade em direção a TOPO. Isto se repete ate' TROCA se igualar ou ultrapassar TOPO, quando então e' "resetado" para a posição imediatamente acima de FUNDO.

De novo, os processos com muita E/S pouco estão no TOPO e portanto pouca probabilidade têm de serem "trocados", se comparados aos de muita CPU.

APÊNDICE B

E/S - IMPLEMENTAÇÃO DE GERENTES

Serão apresentados os detalhes de implementação e funcionamento de GERENTES de E/S. A maioria dos conceitos apresentados nos capítulos anteriores deste trabalho e nas ref. [12], [13] são usados no texto que se segue.

A gerência de cada periférico normalmente e' feita por 2 módulos, que são respectivamente um processo chamado "GERENTE do PERIFÉRICO" e o MANIPULADOR de INTERRUPÇÕES.

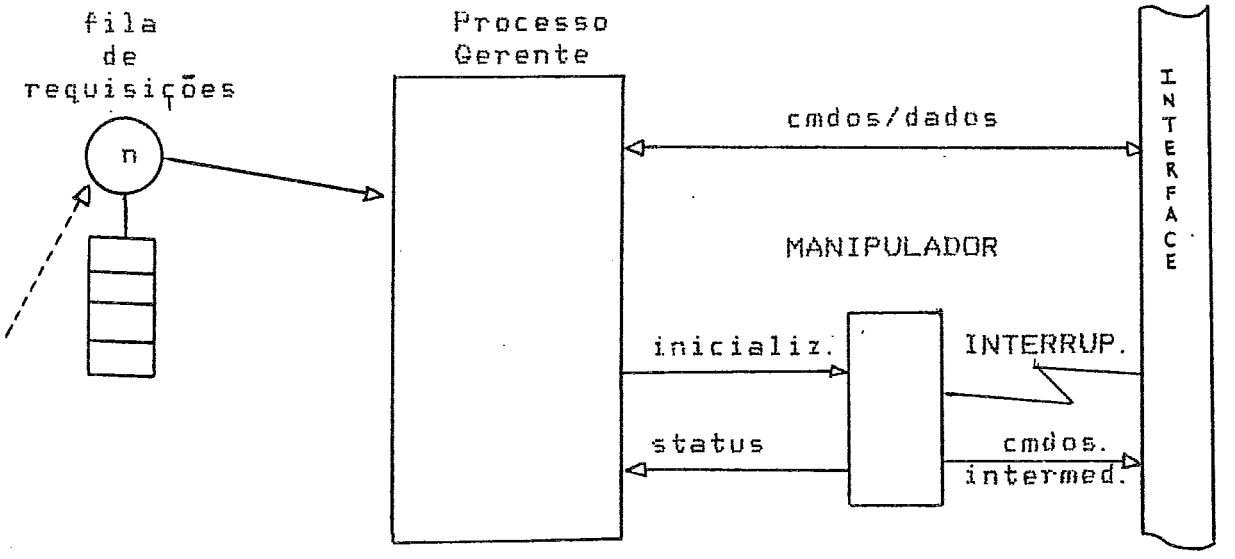


Fig B.1 - GERENTE de E/S TÍPICO

O processo GERENTE do PERIFÉRICO e' a parte central deste conjunto, sendo de sua responsabilidade:

* Toda a comunicação com o processo que esta' fazendo o pedido.

* A decodificação e o preprocessamento do pedido (blocagem, deblocagem, etc.).

* A eventual ativação do periférico, inclusive preparando o MANIPULADOR para receber a(s) interrupções.

* O recebimento e tratamento de eventuais dados e status do MANIPULADOR.

O MANIPULADOR, por sua vez tem as seguintes tarefas:

* Ativado por interrupções, deve fazer o primeiro tratamento dos dados/status.

* Tratar as interrupções intermediárias, não as passando ao Gerente.

* Ao término da operação, ativar o gerente, passando-lhe o status e eventuais dados resultantes da operação.

Cada gerente, sendo um processo, e' independente dos demais gerentes e processos em execução na máquina. Um mesmo gerente pode atender a mais de um periférico, com a restrição que as operações nos mesmos sera' sequencial, pois um processo só pode executar uma tarefa de cada vez. Deste modo, o grau de paralelismo nas operações de E/S e' configurável facilmente no momento da criação de cada gerente.

B.1 GERENTE DE PERIFÉRICO

Em linhas gerais, este elemento e' um processo servidor, que atende requisições de uma fila. O funcionamento das filas de requisição podem ser vistos no capítulo sobre comunicação entre processos.

B.1.1 INICIALIZAÇÃO

Para iniciar suas funções, o gerente deve criar a fila e dar-lhe um ou mais nomes. Estes nomes devem ser o do(s) VOLUME(S) que o gerente ira' atender. Notar que um gerente pode atender a mais de um VOLUME, sendo portanto necessário dar tantos nomes 'a fila quanto forem estes VOLUMES. A operação CRISER do NÚCLEO deve ser usada na criação da fila e NOMSER, para dar-lhes os nomes.

Este nome e' usado posteriormente pelos processos que farão E/S, no momento da abertura dos arquivos. Ele permite que o número da fila seja determinado. Maiores detalhes podem ser encontrados no

capítulo de E/S, no tópico ABRA-ARQUIVO.

Passo seguinte, deve ser estabelecida a ligação processo gerente <-> manipulador. Em cada manipulador existe uma tabela que o descreve. O endereço desta tabela pode ser conhecido via uma chamada 'a operação INIMAN do núcleo. Com o endereço e mapeamento obtidos, e' possível preencher a dita tabela e quaisquer parâmetros particulares no manipulador.

Em seguida e' necessário mapear uma página virtual sobre o periférico, o que deve ser feito por intermédio da operação MAPFIS do núcleo.

A programação inicial do periférico, caso necessário, deve ser feita neste ponto.

B.1.2 ATENDIMENTO

Normalmente o atendimento das requisições de E/S e' cíclico, sendo iniciada com a operação "RETIRE" do NÚCLEO. Obtem-se assim a ID do processo que fez o pedido de E/S e o endereço da BCES que descreve a operação.

O endereço obtido e' relativo ao processo que fez o pedido; portanto, e' necessário mapear 2 páginas, ou seja, 4K (operação MAPVIR, gerente de memória) sobre o mesmo.

Obtida a BCES, o campo BCOPER deve ser examinado, de modo a determinar a operação a efetuar. Dai' pode acontecer:

1. Se ABERTURA, o gerente deve alocar todos os recursos necessários 'as operações que se sucederão neste arquivo; eventualmente privatizar o arquivo ou periférico. Caso ja' esteja privatizado por outra BCES, a operação deve ser rejeitada, voltando um código de erro em BCSTAT. O campo BCAPTR e' reservado para uso qualquer pelo gerente, servindo normalmente para indicar o arquivo a tratar, em gerentes que manipulem vários arquivos/periféricos ao mesmo tempo.

2. Se LEITURA, ESCRITA ou ESPECIAL, a BCES também traz no campo BCEND e BCTAM o endereço do buffer na área do usuário e seu tamanho. Novamente um mapeamento e' necessário para acessar-se o buffer.

A atitude a tomar nas operacoes de E/S varia de acordo com o periférico. Alguns fazem blocagem antes de escrever fisicamente e deblocam na leitura, como e' o caso do gerente de disco, cuja unidade de transferência e' 256 bytes; portanto, nem sempre uma requisição de E/S se traduz numa operação física sobre o periférico.

Se uma operação física for necessária, o gerente deve preparar o MANIPULADOR para a INTERRUPÇÃO que se dara' ao fim da mesma. Isto consiste em setar STAGER no manipulador, informando-o que o gerente entrou em espera pelo término da operação, e portanto deve ser ativado. Também informações particulares entre os 2 podem ser feitas.

Passo seguinte, deve ser passado ao periférico os dados e comandos; em seguida o processo entra em ESPERA-FISICA (operação ESPFIS do NÚCLEO), somente sendo ativado pelo manipulador ao fim da operação fisica.

O status da operação física deve ser obtida do manipulador. Se tudo ok, podem ser necessárias outras operações físicas para atender completamente o pedido, e ai' os procedimentos dos parágrafos acima devem ser repetidos. Finalmente, o campo BCSTAT deve ser preenchido devidamente, a operação FINALIZE invocada e voltar-se ao princípio para pegar uma nova requisição.

B.2 MANIPULADORES

São programas obrigatoriamente "INDEPENDENTES de Posicao" (PIC). Além disto, nos seus primeiros 32 bytes deve residir uma tabela com a forma abaixo:

1. INT - (2 bytes) - endereço da entrada no vetor de interrupções que contém o endereço da rotina de atendimento das interrupções. Preenchido pelo CONFIGURADOR

2. SLOT - (2 bytes) - endereço físico do periférico atendido pelo manipulador. Preenchido pelo CONFIGURADOR.

3. ENDMAP - (2 bytes) - inicialmente deve conter o número de janelas desejadas pelo manipulador para mapear sobre outros processos (passagem de dados/ comandos/status). Após a configuração contém o endereço correspondentes 'as janelas alocacadas.

4. IDGER - (2 bytes) - ID do processo gerente, e' preenchido pelo próprio quando necessário.

5. STAGER - (1 byte) - Status do processo gerente em relação ao manipulador:
   bit 7 - esperando pelo término da operação
   bit 6-0 - qq. valor, normalmente o cod. da operação física a realizar.

6. NOME - (6 bytes) - nome pelo qual o gerente encontra o manipulador. Deve ser único.

7. PRÓXIMO - (2 bytes) - endereço do próximo manipulador. Prenchido pelo CONFIGURADOR.

Os manipuladores residem num processo especial, denominado PROCESSO DE MANIPULADORES, mostrado na Fig. B.2. A região I esta' mapeada sobre a área do sistema operacional. A região II contém em seus 4K iniciais a área GLOBAL e as tabelas de vídeo. No espaço restante estão os manipuladores propriamente ditos. A região III fica permanentemente mapeada sobre os periféricos, cada 2K sobre um deles. Seus últimos 2K ficam mapeados sobre o sistema operacional , como os demais processos.

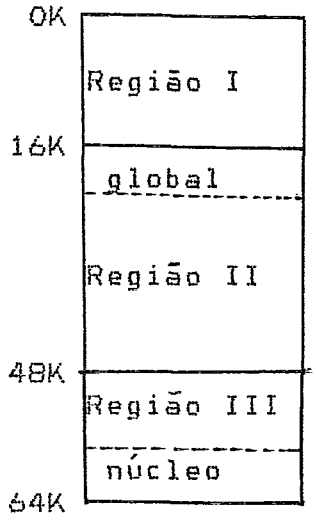


Fig B.2 - PROCESSO DE MANIPULADORES

O VETOR DE INTERRUPÇÃO se encontra na área global. Cada uma de suas entradas esta' associada a um periférico. E' composta por 4 bytes, sendo o primeiro o código "JUMP" e o segundo e terceiro o endereço rotina que atendera' 'a interrupção. O campo INT de cada manipulador aponta para a entrada no vetor de interrupção correspondente ao mesmo. E' importante ressaltar que a qualquer momento o manipulador pode alterar o vetor. Assim, na próxima interrupção o desvio sera' automático para a rotina, diminuindo o overhead.

No momento da geração do sistema operacional (via o programa CONFIGURADOR) , os manipuladores dos perféricos presentes na configuração são aglutinados neste processo [20].

B.2.1 DETALHES PARA IMPLEMENTAÇÃO

Um manipulador esta' bastante ligado ao HARDWARE. Ao acontecer uma interrupção externa, os registradores PC e CCR são salvos na pilha "S" [11]. Em seguida , o contexto de processo e' automaticamente mudado para o PROCESSO DE MANIPULADORES; também novas interrupções ficam inibidas. Passo seguinte e', feito o desvio para a rotina específica de tratamento daquela interrupção, via o vetor de interrupções.

O registrador "S" não esta' válido neste novo mapeamento, sendo necessário recarregá-lo. Também as interrupções do relógio de tempo real, vídeo e refresh de memória dinâmica ficam inibidas, sendo necessário liberá-las imediatamente.

Segue abaixo a forma geral que um manipulador deve ter. As instruções iniciais e finais são obrigatórias em qualquer manipulador.

```
          STS     salvaS  *salva reg. S
          LDS     #pilman *aponta pilha S
          ANDCC   #libvid *libera relóg, vid. rfrsh
            -
            -             * salva regs. que vai usar
            -
            -             *trata interrupção
            -
            -
          JMP     retint  *termina
salvaS    EQU     16*1024 (16K)
pilman    EQU     48*1024-1 (48K)
libvid    EQU     $EF     (lib. IRQ)
retint    EQU     $xxxx
```

APENDICE C

CÓDIGOS PARA BCES

Os códigos para BCES são bastante simples. De um modo geral, cada operação tem bits que identificam-na, sendo os demais usados como modificadores. Os bits não declarados abaixo, podem ser usados para informar características específicas de periféricos. Um exemplo e' a leitura a nível de carácter do teclado [3]. Notar que alguns modificadores não têm sentido em alguns periféricos, sendo protanto ignorados ou rejeitados, dependendo do caso, nestes periféricos.

Segue o formato do campo BCOPER, para cada operação básica:

ABERTURA : bits 7,6,5 = 000
modificadores: bits 2..0

- 000 - abertura simples
- 001 - abertura compartilhada
- 100 - criação simples (arq. sequencial)
- 101 - criação simples (arq. randomico)
- 110 - criação com deleção (sequencial)
- 111 - criação com deleção (randômico)

LEITURA : bits 7,6,5 = 010
modificadores : bits 2..0

- 000 - leitura simples
- 001 - leitura acesso direto
- 100 - leitura indexada

GRAVACAO : bits 7,6,5 = 100
modificadores : bits 2..0

- 000 - gravação simples
- 001 - gravação acesso direto
- 100 - gravação indexada

ESPECIAIS : bits 7,6,5 = 111
modificadores : bits 2..0

000 - feche arquivo
001 - atualize diretório
111 - apague arquivo

APENDICE D

OPERAÇÕES DO NÚCLEO

D.1 INVOCANDO AS OPERAÇÕES

As operações do NÚCLEO são rotinas que residem na memória do sistema. Estas rotinas devem ser chamadas diretamente via instruções de chamada 'a sub-rotinas, uma vez que estão mapeadas na área de endereço virtual de cada processo (nos ultimos 2K).

De modo geral, estas rotinas são disponíveis apenas aos PROCESSOS. As invocáveis por MANIPULADORES são explicitamente indicadas.

Cada OPERACÃO devolve ao seu término um STATUS no ACUMULADOR "A" do processador, informando seu sucesso ou não. O valor zero indica sucesso total. Outros códigos indicam advertência ou erro, e dependem de cada uma delas. Os códigos entre 1 e 127 indicam ADVERTÊNCIA; entre 128 e 255 ERRO. Nos APÊNDICES encontra-se uma lista c/ os erros e sua explanação. Vide como proceder p/ exibir na tela uma mensagem, na operação MENSAG.

D.1.1 PASSAGEM DE PARÂMETROS

Existem 2 modos de passar parâmetros p/ as operações do NÚCLEO: via a PILHA "U" ou via os REGISTRADORES do processador.

Para passar via a PILHA "U", o primeiro parâmetro deve estar no topo da pilha, sendo seguido pelos demais, ficando o último parâmetro no fundo da pilha. O reg. "U" deve estar apontando p/ o topo desta pilha antes de efetuar-se a chamada.

Para passar via REGISTRADORES, em primeiro lugar o reg. "U" deve ser zerado. Em seguida, de acordo com a descrição da operação em questão, cada reg. deve ser carregado com o respectivo parâmetro. Na descrição das operações, em cada parâmetro, e' dado o nome do reg. que lhe corresponde, entre parênteses.

Alguns parâmetros são OPCIONAIS. Se a passagem for via pilha "U" e não se quiser passar um destes parâmetros, colocar o valor ZERO em sua posição na pilha. Ja' se for via registradores, ZERAR o registrador correspondente ao parâmetro.

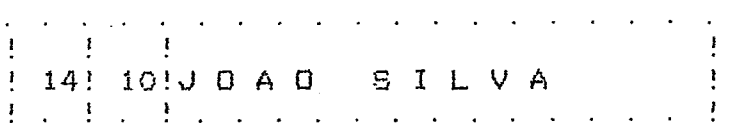
Os parâmetros bem como a pilha "U" devem estar numa das páginas ativas no momento da chamada 'a qualquer das operações. Além disto, devem estar entre os endereços 16K e 62K-1.

D.1.1.1 TIPOS DE PARÂMETROS -

Existem 2 tipos de parâmetros : IMEDIATOS e ENDEREÇOS. No primeiro caso, o valor e' passado como parâmetro. No segundo caso, o endereço onde se encontra o(s) valor(es) e' passado. O tipo e' especificado em cada parâmetro na descrição das OPERAÇÕES. Se nada for dito, assume-se IMEDIATO.

D.2 CADEIAS DE CARACTERES

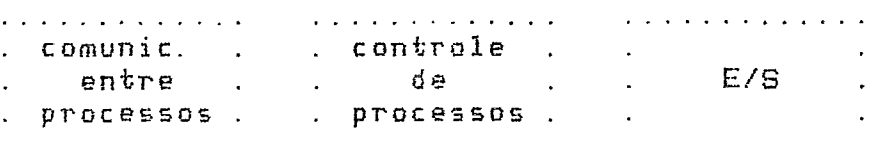
E' definida como um vetor que pode receber um conjunto de caracteres. E' precedido de um descritor que contém o tamanho ESTÁTICO (dimensão máxima do vetor) e o tamanho OCUPADO pelos caracteres presentes em determinado instante no vetor.

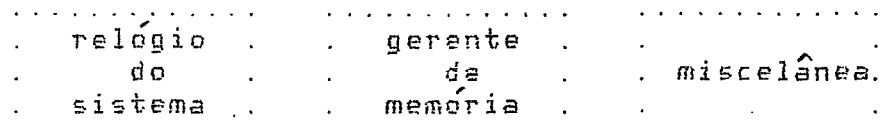


FORMATO DE UMA CADEIA

Cada um dos 2 campos do descritor ocupa 1 BYTE , o que significa que uma cadeia pode ter no MÁXIMO 256 CARACTERES.

D.3 MÓDULOS DO NÚCLEO





MÓDULOS DO NÚCLEO

1. MÓDULO COMUNICAÇÃO ENTRE PROCESSOS - Responsável pela passagem de dados , comandos e sincronização entre processos de qq. tipo. E' sobre este módulo que esta' montada a. espinha dorsal do sistema.

2. MÓDULO CONTROLE DE PROCESSOS - São as OPERAÇÕES e rotinas INTERNAS do NÚCLEO que permitem o ESCALONAMENTO e PREEMPÇÃO entre processos, o controle dos manipuladores de E/S, controle de interrupção, criação/ativação/término de processos, etc.

3. MÓDULO E/S - São as OPERAÇÕES que servem de interface entre os processos e os servicadores de E/S, padronizando e tornando indepeendente de periférico as mesmas. Também contém OPERAÇÕES de suporte 'as E/S, tais como conversão de nomes lógicos, "parsing" de nomes de arquivos, etc.

4. MÓDULO RELÓGIO do SISTEMA - Provê os recusos de calendário , hora do dia e temporização de processos. Também e' responsável pelas conversões de tempo do formato interno, binário, p/ formato externo (cadeia) e vice-versa.

5. MÓDULO GERENTE DE MEMÓRIA - E' o responsável pela alocação / dealocação de memória principal para os processos.

6. MÓDULO MISCELÂNEA - Contém as OPERAÇÕES genéricas tais como conversões ASCII <==> BINÁRIO, manip. cadeias e BCD'S, operações matemáticas, etc.

D.4 COMUNICAÇÃO ENTRE PROCESSOS

1. DEPÓSITO - Deposita em determinada fila de servico um valor, que normalmente e' o endereço dos parâmetros p/ o serviço. Pode ser COM ou SEM ESPERA, significando que o processo entrara' em espera ou não pelo término do atendimento da requisição. Caso o "VALOR" não seja dado, o serviçador assume que todos os recursos alocados p/ o processo devem ser liberados.

   == << PARÂMETROS >> ==
   1. (acc.A) - nro. do serviço + espera (BIT 7 = 1) ou sem espera (BIT 7 = 0)
   2. (rX) - valor a depositar na fila.

2. ESPERA - Faz o processo entrar em espera pelo término do atendimento a um "DEPOSITO" feito anteriormente. Caso não haja pedido na fila em questão, esta operação e' ignorada e o processo continua sua execução.

   == << PARÂMETROS >> ==
   1. (acc.A) - nro. do serviço

3. RETIRADA - Usada por processos que atendem as requisições das filas, e' o inverso de "depósito". Devolve o primeiro pedido da fila de serviço em questão (o "VALOR" e ID. do proc. que fez o "depósito"). Caso não haja requisições na fila, o processo entra em ESPERA. Se "VALOR" for igual a zero, o servidor deve liberar todos recursos alocados p/ o processo.

   == << PARÂMETROS >> ==
   1. (acc.A) - nro. do serviço
   2. (rX) - end. de "palavra dupla" onde sera' devolvido o "valor" depositado e a ID do processo que fez o depósito.

4. FINALIZA - Usada pelos processos que ATENDEM as requisições das filas, após o término do atendimento de cada requisição. Retira o pedido previamente atendido do topo da fila, ATIVANDO, se necessário, o processo que fez o deposito.

   == << PARÂMETROS >> ==
   1. (acc.A) - nro. do serviço

5. EXAMINE - Determina o nro. de requisições pendentes em determinada fila.
PARÂMETROS

   1. (acc.A) - nro. do serviço

   2. (rX) - end. de BYTE que recebera' o nro. de elementos na fila.

6. TRANSFIRA - Transfere um pedido de :
-> 0= topo de uma fila p/ o fim de outra
-> 1= topo de uma fila p/ região de memoria
-> 2= região de memoria p fim de uma fila
Usada normalmente para atender requisições em paralelo de uma so' fila, sendo as requisições copiadas pelo servidor. para região sua de memoria.

== << PARÂMETROS >> ==

   1. (acc.A) - código da operação (um dos valores acima) BYTE.
   2. (rX) - ORIGEM : se cod. 0 ou 1, nro. do serviço. Se cod. 2, end. de 5 bytes de memoria contendo o "VALOR" a depositar, a ID do processo dono e 1 byte informando se com ou sem espera (bit 7 em 1 ou 0 respectivamente).
   3. (rY) - DESTINO : se cod. 0 ou 1, nro. do serviço. Se cod. 2, end. de 5 bytes de memoria que receberão o "VALOR" depositado, a ID do processo que o fez e 1 byte informando se foi com ou sem espera (bit 7 em 1 ou 0 respectivamente).

7. CRIE-SERVICO - Cria uma fila de serviço, devolvendo o nro. dado a esta fila de serviços. Usada por processos que criam a fila para depois ATENDER as requisições depositadas nas mesmas.

== << PARÂMETROS >> ==

   1. (rX) - end. de um BYTE onde sera' devolvido o nro. do serviço criado.

8. NOME-A-SERVIÇO - Atribui um nome a uma fila de serviço que ainda não o tenha ou mais um nome à fila que ja' os possua. Assim, uma mesma fila pode receber vários nomes e o acesso a mesma pode ser feito por qq. um deles, via ABRA-SERVIÇO.

== << PARÂMETROS >> ==

   1. (rX) - end. de cadeia com o novo nome a atribuir a fila.

2. (Acc.A) - BYTE que contém o nro. da fila que recebera' o nome.

9. TIRE-SERVIÇO - O inverso de "crie-serviço", desativa determinado serviço . Usada por quem criou o serviço ao findar suas operaçoes.

== << PARÂMETROS >> ==

1. (acc.A) - nro. do serviço

10. TIRE-NOME-SERVIÇO - tira um ou todos os nomes de uma fila de serviços. Para tirar todos, deve-se especificar somente o nro. da fila.

== << PARÂMETROS >> ==

1. (Rx) - end. de cadeia c/ nome a retirar [opcional]
2. (Acc.A) - nro. da fila a retirar os nomes. Só e' necessário qdo. parâmetro 1 não dado.

11. ABRA-SERVIÇO - Dado um NOME DE SERVIÇO, retorna o nro. associado ao mesmo. Usado por processos que querem fazer um "deposito" em deter. fila, e querem saber o nro. da mesma a partir de seu nome. (o nome que foi dado por CRIE-SERVIÇO).

== << PARÂMETROS >> ==

1. (rX) - end. de cadeia c/ nome do serviço
2. (rY) - end. de BYTE que recebera' o nro. do serviço.

12. MARSER - marca o serviço como sendo usado pelo processo. O processo serviçador deve usar esta operação qdo. desejar ser avisado do término do processo.

== << PARÂMETROS > ==

1. (acc.A) - nro. do serviço a marcar
2. (rX) - ID do processo para o qual sera' marcado o serviço [opcional].

D.5 CONTROLE DE PROCESSOS

1. ATIVE - Ativa um processo , colocando a ID do mesmo na PILHA de EXECUÇÃO. Automaticamente e' feito um escalonamento , p/ verificar se o estado do sistema (qto. a escalonamento) mudou. Pode ser usada por processos e manipuladores.

   == << PARÂMETROS >> ==
   1. (acc.D) - ID do processo (palavra).

2. ESPERA-FÍSICA - O processo e' desativado, saindo da PILHA de execução. Usado normalmente p/ a espera de um evento externo (interrupção). P/ retomar a execução e' necessário ser ATIVADO por outro processo ou manipulador de interrupção.

3. TERMINE - Termina o processo especificado, liberando os recursos usados pelo mesmo. Caso a ID ou o nome do processo não sejam dados, termina o próprio processo. Se houver um outro processo 'a espera do término deste , ATIVA-O. Os 2 parâmetros abaixo são MUTUAMENTE EXCLUSIVOS, somente um dos dois pode ser especificado.

   == << PARÂMETROS >> ==
   1. (acc.D) - palavra c/ a ID do processo [opcional]
   2. (rX) - end. de cadeia c/ nome do processo [opcional]

4. RETINT - Retorna de um manipulador. Pode ser ao processo que foi interrompido ou ao escalonador , dependendo do estado do sistema (mudado p/ ATIVE)

5. INIMAN - Dado o nome de um manipulador, mapeia uma página do processo requisitante sobre o mesmo e devolve o endereço virtual do início deste manipulador (sua tabela de controle). E' necessário ativar a página após esta chamada.

   == << PARÂMETROS >> ==
   1. (Acc.D) - pag. a mapear sobre o manipulador
   2. (rX) - end. de 6 bytes que contém o nome do manipulador
   3. (rY) - edn. de palavra que recebera' o end. virtual inicial do manipulador,

6. OPER-REM - Invoca OPERAÇÕES DO NÚCLEO remotamente, i.e', permite que um processo possa invocar as operações de outra máquina. Para tanto e' necessário especificar-se o no' da máquina, a operação a executar e os parâmetros para a operação. O processo entra obrigatoriamente em ESPERA pelo término da operação remota. Maiores informações são

encontradas na ref. [1] ou no cap. VIII, que inclusive tem uma lista de operações invocáveis remotamente.

== << PARÂMETROS >> ==

1. (rX) - endereço da lista contendo o NO onde sera' executada a operação, o end. da operação, o nro. de parâmetros para a operação e adescrição do tamanho de cada parâmetro (vide cap. VIII).
2. (rY) - endereço da lista de parâmetros p/ a rotina que vai ser executada: vide "parâmetros" da rotina que sera' executada.

7. DEFAULT - Atribui valores aos "defaults" de arquivos de um processo. São atribuídos "NO", "VOLUME", "DIRETÓRIO". Se ID do processo for dada, este e' alterado. Caso contrário, o próprio processo e' alterado.

===>> PARÂMETROS <<===

1. (acc.D) - ID do processo a mudar os defaults. [opcional]
2. (rX) - end. de cadeia c/ no, volume, diretório a mudar - somente os campos dados são alterados - vide sintaxe nome de arquivo.

8. ESTADO - informa características de um processo. A seleção de uma ou todas as características e' feita via um código, passado como parâmetro. A ID do processo, se não dada, asssume o próprio processo. Ao retorno da operação, o parâm. 2 aponta p/ um novo processo. Assim, com chamadas sucessivas e' possível pegar-se as características de todos os processos. P/ pegar o primeiro processo, colocar o valor -1 no lugar da ID do processo , antes da primeira chamada.

== << PARÂMETROS >> ==

1. (acc.A) - código, que especifica o que pegar:
   0 = nome do processo. (cadeia tam. max =6)
   1 = defaults do processo (1byte no, 6 p/ volume, 6 p/ diretório)
   2 = IDUS de quem criou o processo. (palavra)
   3 = PRIVILÉGIOS do processo. (BYTE)
   4 = CARAC. do processo (BYTE)
   5 = HORA início (palavra dupla)
   6 = NROES , da o nro. E/S executadas pelo proc. (palavra)
   7 = nome do programa (completo, c/ no, volume, diretório, nome). Formato idêntico ao campo BCARQ da BCES.
   8 = TODOS os dados sobre o processo (vide formato no apêndice F, descritor de processos.

2. (rX) - end. de variável que contém a ID do processo. [opcional]
3. (rY) - end. de uma área de memória onde serão devolvidas as infos. O formato depende do parâmetro selecionado.

9. IDPROC - devolve a ID do PROCESSO que a chamou.

== << PARÂMETROS >> ==

1. (rX) - end. de palavra que receberá a ID do PROCESSO.

D.6 ENTRADA/SAÍDA

1. ABRA-ARQUIVO - Torna acessível e/ou privatiza determinado arquivo ou periférico. Suas funções são:

   - a) converte o nome lógico (caso dado),
   - b) analisa e separa os campos do NOME do arquivo, caso dado, colocando-os na BCES. Os campos que estiverem faltando recebem os defaults do usuário.
   - c) determina o nro. da fila de requisições do periférico (o VOLUME da' o "nome do serviço", sendo "abraserviço" invocada neste pto.)
   - d) deposita a BCES na fila deter. no item "c" ou no sist. de REDE (se o no' for diferente do desta máquina).

   == << PARÂMETROS >> ==

   1. (rX) - end. da BCES
   2. (rY) - end. de cadeia c/ nome do arquivo ou nome lógico. [opcional] O formato de um nome de arquivo e' "NO:VOLUME:DIRETÓRIO.NOME.EXT".

2. FECHE-ARQUIVO - Desativa ou dealoca o arq./periférico indicado pela BCES. Faz um depósito na fila de requisições indicada ppela BCES dada.

   == << PARÂMETROS >> ==

   1. (rX) - end. da BCES

3. LEIA , ESCREVA , ESPECIAL - lê/grava/executa operação específica num arquivo/periférico previamente aberto. A leitura/gravação e' de um elemento de informação (caráctер, registro). Os campos da BCES pertinentes 'a operação devem ser previamente preenchidos. Deposita a BCES na fila de requisições especificada pela própria BCES.

   == << PARÂMETROS >> ==

   1. (rX) - end. da BCES

4. SISENT, SISSAI - São 2 rotinas que escrevem (SISSAI) e lêem (SISENT) uma cadeia de caracteres no periférico associado aos nomes lógicos SISENT e SISSAI respectivamente. Estes 2 nomes lógicos são padrões do sistema.

   == << PARÂMETROS >> ==

   1. (rX) - end. de cadeia c/ caracteres a gravar ou ler,

5. CONVER - Pega na TAB. NOMES LÓGICOS o nome de arquivo equivalente ao nome lógico dado. Se o nome do processo não for especificado, o default e' o proc. requisitante.

== << PARÂMETROS >> ==
1. (rX) - end. de cadeia c/ nome lógico
2. (rY) - end. cadeia c/ nome do processo [opcional]
3. (acc.D) - end. de cadeia onde será depositado o nome do arquivo equivalente

6. SEPARQ - Separa os campos do nome do arquivo dado, depositando-os na BCES especificada. Aos campos que estiverem faltando, são atribuídos os "DEFAULTS" do usuário que criou o processo. E' invocada p/ "abra-arquivo".

== << PARÂMETROS >> ==
1. (rX) - end. da BCES
2. (rY) - end de cadeia c/ nome do arquivo

7. ASSOC - Cria um NOME LÓGICO na TAB. NOME LÓGICOS e atribui-lhe uma cadeia como valor. O valor típico e' um nome de arquivo, mas qq. cadeia pode ser dada. O nome do processo e' opcional , sendo o default o nome do processo que esta fazendo a requisição. Associa um nome lógico a determinado processo.

== << PARÂMETROS >> ==
1. (rX) - end. de cadeia c/ nome lógico
2. (rY) - end. de cadeia c/ nome do processo [opcional]
3. (acc.D) - end. de cadeia c/ valor a atribuir ao nome lógico

8. DASSOC - Retira NOMES LÓGICOS da tabela. Se especificado o nome lógico, retira-o. Se não especificado, retira TODOS os nomes lógicos pertencentes ao processo especificado (ou do processo requisitante, caso não especificado).

== << PARÂMETROS >> ==
1. (rX) - end. de cadeia c/ nome lógico [opcional]
2. (rY) - end. de cadeia c/ nome do processo [opcional]

D.7 RELÓGIO DO SISTEMA

1. HORA - retorna a hora do dia em formato binário, unidade em "tics". Para convertê-la p/ caracteres usar TICCAR (abaixo).

   == << PARÂMETROS >> ==

   1. (rX) - end. de palavra dupla que recebera' a hora em TICS

2. DATA - fornece a data corrente como uma cadeia c/ formato "DD/MMM/AA".

   == << PARÂMETROS >> ==

   1. (rX) - end. da cadeia que recebera' a data (tamanho mínimo = 9 caracteres).

3. ACERTE - acerta p/ o valor dado a DATA e HORA do sistema. Estes valores são atualizados automaticamente pelo NÚCLEO, a partir do valor dado.

   == << PARÂMETROS >> ==

   1. (rX) - end. da cadeia c/ data a acertar, no formato "DD/MMM/AA" [opcional]
   2. (rY) - end. de palavra dupla c/ hora a acertar, em unidades "TIC" [opcional]

4. TEMPORIZE - permite que um processo entre em espera ate' que:

   - a) um tempo delta-T dado se esgote
   - b) uma determinada hora-do-dia chegue.

   == << PARÂMETROS >> ==

   1. (acc.A) - tipo da temporização : 0= delta-T , 1= hora-do-dia
   2. (rX) - end. de palavra dupla c/ delta-T ou hora-do-dia, unidade "TIC"

5. TICCAR - converte um valor inteiro que está em unidade de "TICS" para uma cadeia no formato "HH:MM:SS".

   == << PARÂMETROS >> ==

   1. (rX) - end. palavra dupla a converter , em unidades "TIC"

2. (rY) - end. cadeia que contera' o valor convertido, no formato "HH:MM:SS"

6. CARTIC - Converte uma cadeia no formato "HH:MM:SS" ou "HHMMSS" p/ um valor inteiro, unidade "TIC" (inverso de TICCAR).

== << PARÂMETROS >> ==

1. (rX) - end. de cadeia no formato "HH:MM:SS" ou "HHMMSS" a converter
2. (rY) - end. pal. dupla que recebera' o valor em unidade "TIC"

D.8 GERENTE DE MEMÓRIA

1. DEFPAG - Cria ou redefine páginas para processos. Para criar, a palavra apontada pelo primeiro parâmetro deve conter $FFFF. Sera' devolvido nela o número da primeira página criada. Para redefinir, deve-se dar o nro. da página inicial nesta palavra e as páginas não devem ter memória alocada

   == << PARÂMETROS >> ==
   1. (rX) - end. de palavra c/ nro. da página inicial (redefiniçao) ou $FFFF (criaçao).
   2. (acc.D) - nro. de páginas a criar/ redefinir. Se não dado, assume 1. [opcional]
   3. (rY) - palavra c/ atributos da página (vide apendice F). [opcional]

2. ALOMEM - Aloca memória para um conjunto sequencial de páginas que devem estar desativadas e não ter memória alocada.Eventualmente retira outras páginas da memória para conseguir o espaço.

   == << PARÂMETROS >> ==
   1. (rX) - nro. da página início
   2. (acc.D) - nro. de páginas. SE não dado, assume 1. [opcional]

3. DLOMEM - Libera a memória usada por um conjunto de páginas. Páginas compartilhadas so' são liberadas fisicamente qdo. não houver mais nenhum processo utilizando-as.

   == << PARÂMETROS >> ==
   1. (rX) - nro. da página inicial
   2. (acc.D) - nro. de páginas a liberar memória. Se não dado assume 1. [opcional]

4.

5. ATVPAG - ativa um conjunto de páginas sequenciais. As ja' ativas do grupo não são alteradas. As páginas não residentes em memória são trazidas e depois ativadas. Pode-se especificar as janelas sob as quais serão mapeadas as páginas.

   == << PARÂMETROS >> ==
   1. (rX) - nro. da pág. início

2. (acc.D) - nro. de páginas a ativar. Se não dado, assume 1. [opcional]
3. (rY) - nro. da janela que será mapeada sobre a primeira página. Se não dado, usa a janela default da página. [opcional]

6. ALOCOM - Aloca memória compartilhada. A memória já deve pertencer a um outro processo (caso típico) ou ao próprio processo (multialocação). Usada para mapeamento dinâmico sobre bibliotecas, tabelas de outros processos, etc.

== << PARÂMETROS >> ==

1. (rX) - nro. da pág. início a mapear.
2. (acc.D) - nro. de páginas a alocar. Default = 1 [opcional]
3. (rY) - endereço de duas palavras que contém:
   1. ID do processo com o qual vai se compartilhar a memória.
   2. nro. da pag. início (no proc. destino) sobre a qual vai se mapear.

7. MAPVIR - mapeia 2 janelas (4K) na área virtual ativa de outro processo a partir de um endereço do mesmo. O endereço é automaticamente convertido para o espaço virtual do processo requisitante, de modo a permitir o acesso direto à área.
IMPORTANTE : por motivo de performance, esta rotina NÃO SEGUE OS PADRÕES DE PASSAGEM DE PARÂMETROS. Seus parâmetros só são passados nos registradores do processador.

== << PARÂMETROS >> ==

1. (acc.A) - pág. inicial do proc. requisitante a mapear
2. (rX) - endereço virtual no processo que se quer acessar
3. (rY) - ID do processo sobre o qual será feita o mapeamento.
4. (rX) - endereço virtual CONVERTIDO. Está relativo a uma das páginas mapeadas.

8. MAPFIS - permite o mapeamento de uma janela numa região física qualquer (RAM, PROM, Periféricos, Vídeo).

== << PARÂMETROS >> ==

1. (acc.A) - BYTE c/ nro. da página a mapear. (págs. entre 8 e 30 = 16K a 60K)
2. (rX) - palavra que contém o end. físico a ser mapeado.

9. PROPAG - proteje por HARDWARE um conjunto de páginas contra escrita. Usada para tornar páginas fisicamente não modificáveis, o que não e' feito por DEFPAG.

== << PARÂMETROS >> ==

1. (rX) - nro. da pág. inicial
2. (acc.D) - nro. de páginas a proteger. Se não dado, assume 1. [opcional]

D.9 MISCELÂNEA

Devido a grande variedade de funções executadas por estas operações, elas serão divididas em grupos , de modo que as correlatas fiquem juntas.

D.9.1 MANIPULAÇÃO CADEIAS

1. CMPCAD - compara duas cadeias, retornando um código que indica se a ordem ALFABETICA da primeira e' IGUAL, MAIOR ou MENOR que da segunda. A comparação se da' ate' ser determinado maior ou menor ou ate' uma das cadeias se extinguir. O código e' retornado no reg. STATUS do processador ; portando basta dar um BRANCH "EQUAL", "NOT EQUAL", "HIGH" OU "LESS" após a chamada da rotina. Parâmetros so' via registradores.

   == << PARÂMETROS >> ==
   1. rX - cadeia1 - primeira cadeia
   2. rY - cadeia2 - segunda cadeia

2. MOVCAD - move uma cadeia p/ outra. Se o tamanho estático da cadeia destino for menor que o dinâmico da origem, não e' feito o movimento e o status volta "NOT EQUAL". Parâmetros so' via registradores.

   == << PARÂMETROS >> ==
   1. rX - end. da cadeia origem
   2. rY - end. da cadeia destino

3. CADCRI - move uma cadeia p/ uma região de memória, criando uma cadeia. A memória deve ser suficiente para conter a cadeia acrecida de 2 bytes (o descritor). Parâmetros so' via registradores.

   == << PARÂMETROS >> ==
   1. rX - end. da cadeia origem
   2. rY - end. da memória destino

4. CADMEM - move o conteúdo de uma cadeia p/ uma região de memória, mas NÃO CRIA uma cadeia (não faz descritor). Parâmetros so' via registrador.

   == << PARÂMETROS >> ==

1. rX - end. da cadeia origem
2. rY - end. da memória destino

5. MEMCAD - move um determinado nro. de caracteres de uma posição de memória p/ uma cadeia. Caso nao caibam na cadeia, o movimento não e' feito e o status volta "NOT EQUAL". Parâmetros so' via registradores.

== << PARÂMETROS >> ==

1. Acc.A - nro. de caracteres a mover
2. rX - end. da memória origem
3. rY - end. da cadeia destino

D.9.2 CONVERSÕES

1. ASCBIN - converte uma cadeia ASCII c/ caracteres DECIMAIS para binário.

== << PARÂMETROS >> ==

1. (rX) - cadeia c/ caracteres ASCII de entrada
2. (rY) - end. variável que recebera' o valor binário convertido (palavra)

2. BINASC - converte um valor binário para uma cadeia ASCII.

== << PARÂMETROS >> ==

1. (rX) - variável c/ valor binário (palavra)
2. (rY) - cadeia a receber o valor - os caracteres são CONCATENADOS c/ os anteriores existentes na cadeia. (vide "concatene" acima)

3. ASCBCD - a cadeia dada e' convertida no valor BCD

== << PARÂMETROS >> ==

1. cadeia dada ASCII
2. cadeia BCD que recebera' o valor - deve ser grande o suficiente p/ receber o valor.

4. BCDASC - O valor BCD e' convertido em ASCII.

== << PARÂMETROS >> ==

1. cadeia BCD
2. cadeia ASCII - deve ser grande o suficiente p/ receber o valor.

D.9.3 OPERAÇÕES BCD

1. SOMBCD/SUBBCD - soma/subtrai 2 nros. BCD.

== << PARÂMETROS >> ==

1. operando1 , em BCD
2. operando2, em BCD
3. resultado, em BCD . Pode ser um dos operandos acima ou outro qq.

D.9.4 "ANÁLISE"

São operações destinadas a auxiliar na decomposição e reconhecimento de elementos em uma cadeia .

1. PROXPAL - busca a próxima palavra dentro de uma cadeia. E' considerada "palavra" um conj. de caracteres separados de ambos os lados por início/fim da cadeia ou um dos separadores definidos no parâmetro "separadores". Esta operação SOMENTE pode receber os parâmetros via a pilha "U".

== << PARÂMETROS >> ==

1. cadeia onde se dara' a busca
2. end. de BYTE que contém a posição corrente de busca ; e' atualizado
3. cadeia onde será depositada a "palavra" encontrada [opcional]
4. cadeia onde se encontram os caracteres a considerar separadores.
5. end. de palavra contendo: BYTE 1 => direção, 0 p/ direita , 1 p/ esquerda; BYTE 2 => o separador encontrado. [opcional]

1 MENSAG - Envia uma mensagem de sucesso, advertência ou erro para a tela. Invocada normalmente pelos processos após a invocação de uma operação do NUCLEO que não teve sucesso. É responsável por converter o código em uma cadeia de caracteres, exibindo-a na tela (ou arquivo associado ao nome lógico MSG). Também é responsável em determinar se a mensagem deve ser roteada a esta máquina ou a outra maquina da rede. Além disto, pode ser dado como parâmetro uma cadeia complementar, que sera' exibida após a mensagem convertida.

== [PARÂMETROS] ==

1. (acc.A) - código da mensagem (BYTE), e' o valor devolvido por todas as operações do NUCLEO. Vide apendice E.
2. (rX) - palavra c/ ID-processo "dono" da mensagem - se não dado, assume "dono" o processo que a invocou. [opcional]
3. (rY) - endereço de cadeia c/ a mensagem complementar [opcional]

APÊNDICE E

MENSAGENS DE ERRO

As mensagens podem ser de ADVERTÊNCIA ou de ERRO. No primeiro caso, a condição anormal ou de sinalização detetada não e' fatal, permitindo normalmente que a execução continue. Os códigos para estas mensagens estão entre 1 e 127.

No caso de ERRO, os códigos estão entre 128 e 255, e indica uma situação anormal de gravidade.

A mensagem e' formada do seguinte modo:

ADVR nnn pppppp : mmmmm....mmmmm
ou
ERRO nnn pppppp : mmmmm....mmmmm

onde: nnn = código da mensagem
pppppp = nome do processo que pediu a mensagem
mm...mm= mensagem, composta de palavras que expliquem o erro.

E.0.0.1 MENSAGENS ADVERTÊNCIA -

010 - TERMINADO : processo terminado normalmente
012 - MEMO NÃO LIBERADO : por algum motivo não foi possível liberar o espaço pedido. Verifique os parâmetros e valores passados p/ a operação.

E.0.0.2 MENSAGENS ERRO -

130 - NRO SERVIÇO INVÁLIDO : o nro. dado não existe ou não está ativo neste sistema.
134 - NÃO HA ESPAÇO NO SIST P/ OPERAÇÃO - a alocação de áreas internas do sistema, necessárias 'a operação, não foi posível. Indica que a configuração do sistema esta' mal dimensionada ou sobrecarregada.

136 - HÁ PROC SERVIÇO : mais que um serviçador tentou "servir" uma mesma fila
138 - NÃO HÁ SERVIÇO : não foi possível criar uma nova fila, por ter exedido o máximo nro. de filas permitido p/ esta configuração.
140 - NOME SERVIÇO INVÁLIDO : tentou-se atribuir a um serviço um nome já existente.
142 - PROC INVÁLIDO : o processo especificado na operação e' inválido.
144 - PROC NÃO TERMINADO : normalmente porque tentou-se cancelar um processo permanente.
146 - HÁ REQUIS P/ SERVIÇO : A fila de serviço não pode ser eliminada por haver requisições ainda pendentes na mesma.
148 - NÃO HÁ MEMO : não foi possível alocar a quant. pedida de memória
150 - NOME ARQUIVO INVÁLIDO : erro de sintaxe ou inválido p/ o periférico em questão.
152 - NOME LÓGICO INVÁLIDO : erro de sintaxe ou nome lógico não existente.
160 - ARQUIVO NÃO MODIFICADO
162 - ARQUIVO NÃO APAGADO
164 - ARQUIVO NÃO CRIADO
166 - NAO HÁ ESPAÇO NO VOLUME :
168 - ARQUIVO NÃO ENCONTRADO
169 - PERIFÉRICO INVÁLIDO
170 - COMANDO INVÁLIDO : normalmente usada pela LICO
172 - SEGTO INVÁLIDO : o segmento dado na operação não existe no processo.
174 - NO INVÁLIDO : a máquina dada não se encontra na rede no momento ou e' inválida.
178 - CÓDIGO INVÁLIDO : o código de seleção de opção dentro da função não e' suportado.
180 - NRO INVÁLIDO : mensagem genérica

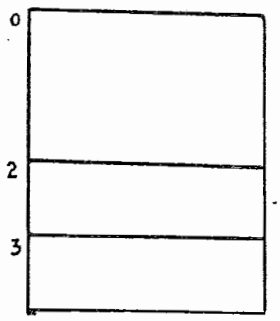
APÊNDICE F

TABELAS

F.1 DESCRITOR DE PÁGINAS

Cada processo tem uma tabela na área do sistema que descreve suas páginas. Uma entrada nesta tabela ocupa 4 bytes e descreve uma página. Como na versão atual do NÚCLEO, por questão de eficiência somente estas tabelas estão limitadas em 2K (uma página), o limite do tamanho do processo e' 1 Mbyte.

Segue o formato de cada entrada:



PGMAP - endereço da pág. na memória FÍSICA ou nro. do registro no arq. de páginas

PGEND - nro. da janela (end. virtual) default da página.

PGSTAT - status e características da página. Fazer bits não especificados = 0.

bit 6:
- 0 - mem. comum
- 1 - mem. de vídeo

bit 5:
- 0 - não modificável
- 1 - modificável

bit 2:
- 0 - pag. removível da memória
- 1 - pag. fixa na memória

bit 0:
- 0 - alocar memória na criação da página.
- 1 - não alocar memória.

F.2 DESCRITOR DE PROCESSOS

Cada processo em execução tem um descritor de processos. Este reside em uma tabela na área do sistema. O número de entradas nesta tabela determina a quantidade de processos que podem existir na máquina em determinado momento.

A chamada a operação ESTADO do NÚCLEO retorna alguns ou todos os campos da entrada referente ao processo especificado.

Segue o formato de cada entrada nesta tabela:

1. DPSTK - Stack pointer "S" ...(2)
2. DPBASE - nro. da base de mapeamento do processo ....(1)
3. DPID - ID. deste processo ...(2)
4. DPIDUS - Id. do usuário que criou o processo ...(2)
5. DPPRIV - privilégios do processo ...(1)
6. DPDEFT - Defaults p/ arquivo
   - DPFNO - no default (1)
   - DPFVOL - volume default (6)
   - DPFDIR - diretório default (6)
7. DPNOME - Nome completo do processo, dividido nos campos
   1. DPNNO - nro. do nó do processo ...(1)
   2. DPNVOL - volume onde está o programa ...(6)
   3. DPNDIR - diretório onde está o programa ...(6)
   4. DPNNOM - nome do processo ; este campo identifica o processo . P/ processos c/ nomes iguais, a ID os diferencia. ...(6)
8. DPSERV - Serviços em uso pelo processo - 64 bits ...(8)
9. DPCPRO - carac. do processo : permanente ou reexecutável e status do processo ...(1)
   - bit 7 => permanente - o processo não e' cancelável nem sua área de memória liberável. Somente o próprio pode se cancelar.
   - bit 6 => reexecutável - após o término, as áreas sob as bases relocáveis são mantidas o máximo possível na memória.
   - bit 5 => status : 0 - desativado; 1 - ativo; 2 a 15 - reservados
10. DPPGFI - end. físico do descritor de páginas do processo ...(2)
11. DPTBPG - end. virtual do descritor de páginas do processo ...(2)
12. DPPGSI - end. do descritor de páginas mapeadas sobre o sistema ...(2)
13. DPPAI - ID do proc. a ativar no final da execução ...(2)
14. DPHORA - hora início da execução ...(4)

15. DPNRES - nro. E/S executadas pelo processo (2)

APÊNDICE G

REFERÊNCIAS

1. Fujihara, Nelson - Um Sistema de Rede Local para Micros. Tese de Mestrado, COPPE-UFRJ (em andamento).

2. Hansen, Brinch - Operating System Principles , Prentice Hall, 1973.

3. MALHEIROS, Pedro Luiz - SIRIUS : Um Sistema de Recepção e Informação ao Usuário . Tese de Mestrado, COPPE-UFRJ, 1983.

4. PETERSON, James L. - Petri Nets. Computing Surveys, Vol. 9 No. 3, setembro 1977.

5. HANSEN, Brinch - The Architeture of Concurrent Programs, Prentice Hall, 1978.

6. BESSO, Stela - Sistema de Gerência de Arquivos em Disco. Trabalho fim de Curso, ENGENHARIA - UFRJ, 1983.

7. CAMPOS, Ricardo Dias - Compilador Pascal com Gerência de Memória. Tese de Mestrado, COPPE-UFRJ (em andamento).

8. HILBURN, Donald - Microcomputers : Hardware, Software and Aplications, Prentice Hall, 1976.

9. WELLER, Walter J. - Practical Microcomputing Programming The 6502, Nothern Technology Books, 1980.

10. MICRO WORKS - CBUG Monitor Owner's Manual -1981.

11. WARREN, Carl D. - MC6809 COOKBOOK, TAB Books Inc., 1980.

12. Leventhal, Lance - 6809 Assembly Language, Orsbone, 1980.

13. DIGITAL - VAX/VMS Command Language User's Guide, cap. 2 1982.

14. TANDY Corporation - TRS-80 Color Basic System Software - 1980.

15. TANDY Corporation - TRS-80 Extended Color Basic, 1980.

16. TANDY Corporation - Color Computer Operation Manual - 1980.

17. MICRO WORKS - Software Development System Owner's Manual - 1981.

18. RICHIE, Denis M. e THOMPSON, Keni - The UNIX Time Sharing System, Com. ACM 17(7) - jul/74

19. BREITINGER, José Lavaquial - O CARREGADOR do sistema Carcara'. A ser publicado.

20. BREITINGER, José Lavaquial - O CONFIGURADOR do sistema Carcara'. A ser publicado.

21. DIGITAL, VAX/VMS internals, 1982.

22. BESSO,e BREITINGER - Especificações do sistema CARCARA' . A ser publicado.

23.

24.

25. DIGITAL, SWAPPER (Newsletter of the VAX systems SIG) - Vol. 4, No. 7, maio 1983.