" S I R I U S "

UM SISTEMA DE RECEPÇÃO E INFORMAÇÃO AO USUÁRIO

PEDRO LUIZ MALHEIROS GUIMARÃES

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:

Profa. SUELI MENDES DOS SANTOS
-Presidente-

Prof. EDIL S. TAVARES FERNANDES
-COPPE/UFRJ-

Prof. MICHAEL STANTON
-PUC/RJ-

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 1983

GUIMARÃES, PEDRO LUIZ MALHEIROS

"SIRIUS" : UM SISTEMA DE RECEPÇÃO E INFORMAÇÃO AO USUÁRIO. (Rio de Janeiro) 1983.

XIII, 130 p., 29,7 cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1983).

Tese - Universidade Federal do Rio de Janeiro - Faculdade de Engenharia.

1. Sistemas Operacionais I.COPPE/UFRJ II.Título (Série).

A O S M E U S P A I S

A G R A D E C I M E N T O S

Em primeiro lugar agradeço aos meus pais, cuja orientação e esforços foram fundamentais para a minha formação e elaboração deste trabalho.

Agradeço à Professora Suely Mendes dos Santos pela orientação e acompanhamento dado a este trabalho.

Agradeço, também, ao primo Cesar e às amigas Maria Luiza, Alice, Miriam, Inês e Ana pela compreensão e incentivo.

Aos amigos José Lavaquial Breitiger e Henrique Mariano do Amaral pelas discussões e sugestões dadas.

Aos amigos José Antonio Monteiro de Queiroz e José Ubaldo Baião pelo apoio e estímulo.

A Elenir Coutinho e Socorro Araújo pela contribuição dada à elaboração deste documento.

Enfim, agradeço a quem sempre acreditou e foi o principal responsável pela elaboração deste trabalho, eu mesmo.

R E S U M O

Este trabalho, intitulado "SIRIUS", foi desenvolvido com o objetivo de prover um microcomputador de um conjunto de funções, as quais definem um "SISTEMA DE RECEPÇÃO E INFORMAÇÃO AO USUÁRIO". Essas funções foram definidas para recepcionar o usuário, através de diretivas oferecidas por uma linguagem de comandos, e informar-lhe os resultados da execução das mesmas. O sistema é subdividido em módulos cujas funções definem a fase do processamento da informação, junto com a utilização das funções intrínsecas do sistema operacional utilizado.

A B S T R A C T

This work, entitled "SIRIUS", was developed with the aim of providing a microcomputer with a set of functions which define a "SYSTEM OF RECEPTION AND INFORMATION TO THE USER".

These functions were defined to interact with the user, through directives offered by a command language, and inform him the result of its execution. The system is subdivided into modules whose functions define an information processing phase, along with the utilization of intrinsic functions of operational system in question.

Í N D I C E

PÁGINA

PÁGINA

PÁGINA

PÁGINA

I N T R O D U Ç Ã O

Este trabalho é um subconjunto de um "Sistema Operacional" cujo desenvolvimento originou-se do projeto de um mi - crocomputador.

O projeto do micro compreende o desenvolvimento paralelo do hardware e do sistema operacional, sendo este composto pelos seguintes módulos: O Núcleo |[2]|, o Loader, o Sistema de Gerenciamento e Manipulação de Entrada e Saída |[1]|, o Editor de Textos, o Sistema de Redes e o Sistema de Recepção e Informação ao Usuário - SIRIUS.

Durante o desenvolvimento dos módulos seguiu-se uma linha de pesquisa onde as definições de suas estruturas, juntamente com suas funções, baseiam-se na atual arquitetura do hardware e do "sistema de monitoramento" feito pelo núcleo. A atual arquitetura do hardware compreende: 2 k de memória PROM , 64-320 k de memória RAM, processador MCS6809 (Motorola), K7, im - pressora, vídeo, teclado, disquete e linhas de comunicação.

O sistema de monitoramento feito pelo núcleo, que chamaremos de MONITOR, além de outras funções internas, compreende a comunicação entre quaisquer dois processos ativos do siste - ma, onde é seguida a filosofia de "depósito e retiradas" utilizando-se filas de requisições |[5]|. Baseando-se nessa filosofia o SIRIUS, juntamente com outros módulos, foi desenvolvido de forma que utilize as funções do núcleo que definem o monitoramento, para qualquer comunicação entre processos.

Dentro do ambiente do sistema operacional qualquer transação é feita através do monitor, onde são chamadas subroti -

nas que atendem a uma determinada solicitação de processos. Dentre outras, as seguintes subrotinas estão contidas no monitor , as quais são referenciadas na estrutura do SIRIUS, definidas no Capítulo III.

DEPOSITE - Deposita um pedido

ASSOCIE - Associa um nome lógico a um arquivo para determinada tarefa.

ESTADO - Informa as características de um processo cujo estado é identificado pela fila onde o mesmo se encontra.

TERMINE - Termina um determinado processo, liberando os recursos por ele utilizados.

ESPERA_FÍSICA - Desativa um processo, colocando-o numa fila de espera.

ATIVA - Ativa um determinado processo.

A subrotina "deposite" determina a comunicação entre esse sistema e outros processos, quando a diretiva solicitada não está contida nas funções das demais subrotinas que compõem o monitor.

O SIRIUS, dentro do ambiente recepção/informação, dispõe de um conjunto de diretivas que estabelecem a comunicação usuário/sistema. A comunicação sistema/usuário é definida quando informações de processos são enviadas para o SIRIUS, tidas como resposta de uma diretiva enviada pelo usuário.

Em seguida é feita a descrição do sistema, onde é definida a "linguagem de comandos" - LICO, contendo as diretivas disponíveis, como também os meios de utilização da mesma.

CAPÍTULO I

1. D E F I N I Ç Ã O

1.1 - MEIOS DE UTILIZAÇÃO

Para que um usuário utilize o SIRIUS é necessário que o mesmo entre com informações que resultarão numa saída enviada pelo sistema. As informações de entrada são enviadas ao sistema através de um teclado, tipo máquina de escrever, e são recebidas pelo usuário através de impressão num vídeo 30x80 L/C. Durante a entrada das informações, o usuário terá como referência um cursor que indicará a posição do vídeo onde serão impressas as informações por ele tecladas.

1.1.1 - VÍDEO

O vídeo é composto por duas janelas separadas por uma linha contínua: a janela inferior, onde são impressas as informações tecladas; e a janela superior, onde são impressas as informações de saída do sistema.

Além da linha de entrada, a janela inferior é também utilizada para enviar informações ao usuário, caso a janela superior esteja sendo utilizada. Na linha de separação das janelas, as últimas colunas são utilizadas para informar ao usuário o "MODO" em que se encontra o sistema. Nessas colunas aparecerá o nome do módulo que receberá as informações tecladas. Caso o teclado se encontre em "estado de erro" ou "inserção de caracter", o sistema informará ao usuário através desta linha de separação.

Quando uma informação é teclada esta pode ser enviada para um processo que requisitou uma entrada ou para a linguagem de comandos do SIRIUS (LICO). Todo comando dirigido para a LICO é feito com o cursor posicionado no início da próxima linha disponível da janela inferior, a menos que comandos anteriormente teclados sejam aproveitados. Os caracteres são impressos na ordem em que são teclados, e serão enviados ao sistema quando for pressionada a tecla "fim-de-comando"(ENTER). Após o recebimento do comando pelo sistema, as informações são impressas na janela superior, fazendo o remanejamento das linhas no sentido vertical para cima, caso a janela esteja cheia. Com isso, qualquer comando enviado para a LICO aparecerá na janela superior após seu recebimento, e o cursor se posicionará na próxima linha da janela inferior.

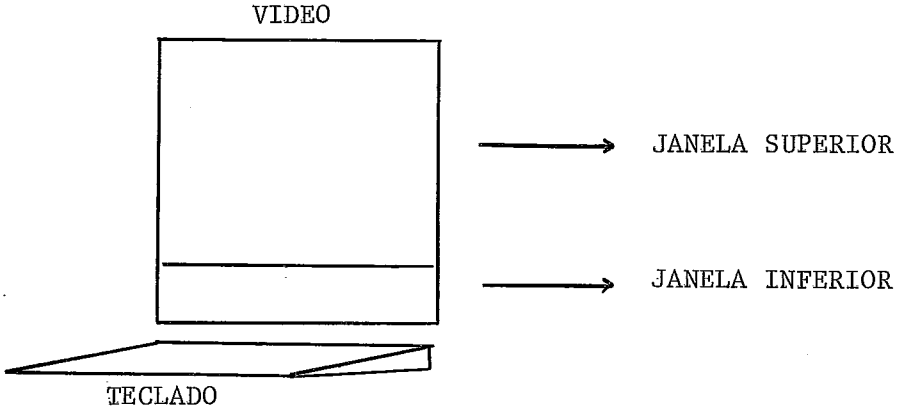


FIG. 1

Para que o funcionamento das janelas seja independente, existem dois cursores para direcionamento da impressão dos dados. O cursor da janela superior, transparente ao usuário, direciona a informação de saída de maneira que esta seja impressa na posição por ele apontada. O cursor da janela inferior indica a posição onde são impressas as informações de entrada e as informações de saída, caso a janela superior esteja sendo utilizada pelo sistema.

O usuário poderá dirigir o cursor inferior para a janela superior, atingindo uma linha desejada. Isso significa que o usuário poderá dar entrada num comando anteriormente te - clado posicionando o cursor, alterando informações, caso necessário, e pressionando a tecla "entre". Após a entrada da informação o cursor retornará à posição inicial, "linha-de-entrada", apontando para a primeira posição a ser impressa.

1.1.2 - TECLADO

O teclado, tipo máquina de escrever, é composto por teclas que representam letras de "A" a "Z", dígitos de "0" a "9", caracteres de pontuação (caracteres . , ;), caracteres especiais, caracteres de controle, caracteres funcionais e os de tela.

Os caracteres de controle são aqueles que exercem funções sobre o cursor, direcionando-o para a posição desejada pelo usuário. Os funcionais são aqueles que têm influência sobre as informações enviadas ao sistema. Quando o teclado estiver sendo utilizado como entrada de um processo, a informação é dirigida para a LICO caso o usuário pressione a tecla <CMD>, antes do comando desejado. Além da tecla <CMD>, existe a tecla <FUN> que define uma diretiva para o editor, e a tecla <ENTRE> que define o final de um comando para a LICO, ou um final de registro para processos que utilizem o teclado como entrada de informações.

- ATENDIMENTO

Quando um comando é recebido pelo sistema, após sua interpretação, é gerado um pedido de execução de tarefa ao módulo apropriado. Os pedidos são depositados na fila e são a-

tendidos de acordo com sua ordem de chegada (FIFO). Com esse esquema de atendimento o Manipulador de Entrada do SIRIUS, módulo que recebe os comandos, após encontrar um "final-de-comando" <ENTRE>, retorna o cursor para a linha de entrada e fica pronto para receber o próximo comando, sem depender da execução da última tarefa solicitada. Essa liberação do teclado faz com que o usuário defina suas entradas de maneira que a LICO não se torne responsável por execuções indevidas de tarefas já que, para cada processamento de comando, existe uma resposta do sistema para o usuário.

1.2 - LINGUAGEM DE COMANDOS

1.2.1 - INTRODUÇÃO

A linguagem de comandos foi definida com o objetivo de tornar todos os recursos do computador disponíveis ao usuário, possibilitando a compilação, execução e depuração de programas, assim como a inserção, deleção e edição de arquivos.

O objetivo primário da linguagem de comandos é de caráter didático, possibilitando ao usuário inexperiente acesso aos recursos do computador sem a utilização de manuais.

A linguagem é dividida em três classes de comandos: Os COMANDOS DE PROGRAMAS, que executam tarefas referentes a programas fontes e objetos; os COMANDOS DE ARQUIVOS, que são responsáveis pelas tarefas que manipulam os arquivos; e os COMANDOS DE SISTEMA, que tratam do sistema como um todo.

Nesse sistema, qualquer massa de informações é tratada como um arquivo, independente de ser programa-fonte, programa-objeto ou massa de dados e, independente, ainda, do peri-

férico. Para se especificar um nome de arquivo existe a sintaxe a seguir: "nó:vol.dir.nome.ext", onde "nó" é o número do sistema dentro da rede de comunicações; "vol" é o volume ou periférico onde o arquivo se encontra; "dir" é o diretório a ser associado ao nome do arquivo; "nome" é uma identificação dada pelo usuário (no máximo 6 dígitos); e "ext" (extensão) é uma informação que define o dado como sendo programa-fonte, programa-objeto, dados, etc. Outra possibilidade de se especificar um arquivo é via um NOME LÓGICO. Um NOME LÓGICO pode ter, no máximo, 6 caracteres e foi previamente definido como um NOME DE ARQUIVO. Durante a execução do comando, esse nome é substituí - do pelo NOME DO ARQUIVO. Para maiores informações sobre NOMES LÓGICOS vide comando "associe". No caso de VOLUMES ESTRUTURA - DOS (disco), serão usados os "defaults" definidos no início da sessão, a menos que os mesmos sejam especificados pelo usuário. Já para VOLUMES NÃO-ESTRUTURADOS (tela, vídeo, etc.), somente "nó:volume:" devem ser especificados. Se mais de um arquivo têm o mesmo nome, estes são ditos ARQUIVOS IRMÃOS e podem ser tratados por um só nome, substituindo-se a "extensão" por um "*". Para alguns comandos poderá ainda ser usada a forma "*.ext", a qual faz referência à todos os arquivos daquele diretório com mesma extensão e, ainda, a forma "*.*", que faz referência à todos os arquivos daquele diretório.

São as seguintes as extensões existentes:

DAD - Dados de programas

COB - Programa-fonte em linguagem COBOL

BAS - Programa-fonte em linguagem BASIC

PAS - Programa-fonte em linguagem PASCAL

EXE - Programa em código executável

LIC - Arquivos de comandos LICO

TXT - Arquivos de trabalho

LST - Arquivos de listagem

SIS - Arquivos de sistema

A identificação de um arquivo para compilação ou execução é feita através das extensões acima descritas. Caso somente o NOME e a EXTENSÃO sejam mencionados, é considerado o NÓ em que se encontra o usuário, o VOLUME é o disco de sistemas, e o DIRETÓRIO é o do próprio usuário (defaults mencionados no início da sessão).

1.2.2 - ARQUIVOS DE COMANDOS

Os arquivos de comandos, "arquivos LICO", são arquivos compostos por um conjunto de comandos os quais foram definidos previamente pelo usuário. Esses arquivos têm como objetivo privar o usuário de teclar sempre os mesmos comandos que compoem uma tarefa a ser executada.

Os arquivos LICO são guardados em disco pelo usuário com o nome "nome.LIC", onde "nome" é definido pelo usuário e "LIC" é a extensão que identifica o arquivo LICO para a linguagem de comandos.

Para ser executada uma tarefa composta pelos comandos contidos no arquivo LICO, o usuário deverá teclar o comando "SUBMETA" seguido do nome do arquivo LICO, o que implicará na execução de todos os comandos que compoem aquele arquivo. Os ararquivos LICO são compostos por comandos de programas, sistema ou arquivo, podendo ainda ter comandos que definam outros arquivos LICO.

1.2.3 - ARQUIVOS DE TRABALHO

Os arquivos de trabalho são arquivos temporários utilizados por tarefas e que fazem certos passos do processamento, transparentes ao usuário.

- FUNCIONAMENTO

Na maioria das vezes, a execução de uma tarefa conta com alteração de informações em arquivos previamente guardados pelo usuário. Com isso, o usuário acessa o arquivo, altera as informações e executa a tarefa desejada.

Tomando como base esses procedimentos, define-se a seleção do arquivo a alterar, por parte do usuário. Desta forma, qualquer arquivo a ser editado deve ser selecionado pelo usuário, implicando na sua cópia do FONTE para um ARQUIVO DE TRABALHO.

Uma vez selecionado o arquivo o próximo passo pode ser uma edição, execução ou compilação, onde não se dá o nome do arquivo. Isso ocorre em virtude de ser objetivo primário dos comandos de arquivo e programas, o tratamento das informações do arquivo selecionado.

Após o programa estar totalmente correto, ou no final de uma sessão do usuário, o mesmo poderá salvar seus arquivos onde são recuperados o fonte e o objeto caso exista.

- CASOS ESPECIAIS

1. Armazenamento da listagem da compilação

É feito através do comando de arquivo que copia o arquivo-listagem (trabalho) para o arquivo desejado.

2. Listagem da compilação na impressora

É feita através do mesmo comando anterior, co - piando do arquivo desejado para a impressora.

- OBSERVAÇÕES:

a. Qualquer COMPILAÇÃO é seguida de uma LINKEDIÇÃO transparente ao usuário. A LINKEDIÇÃO é feita entre a COMPILAÇÃO e a EXECUÇÃO, criando o arquivo intermediário (objeto) e ficando implícita no passo COMPILAÇÃO.

b. O LINKEDITOR usa um nome padrão de biblioteca (de onde tira os módulos objetos), que deve estar no disco do sistema.

c. A listagem da compilação e erros é armazenada num arquivo temporário, "trab.LST", que é recuperado quando salvo pelo usuário, ou deletado sempre quando uma compila - ção é feita.

d. O EDITOR terá recursos para manipular (verifi - car) o arquivo "trab.LST" em paralelo com o arquivo "trab. TXT", que é o arquivo previamente selecionado. O objetivo é auxiliar na depuração, sem o auxílio da listagem (vide comando EXAMINE no EDITOR).

Os comandos são descritos a seguir com sua forma geral (FG), e suas funções, as quais estão diretamente ao nome do comando. Uma descrição mais detalhada dos COMANDOS é dada ao usuário através do comando "AJUDE", descrito a seguir. Para representar a natureza do parâmetro no comando, é utilizada a notação "nome", representando os parâmetros obrigatórios, e <nome>,

representando os opcionais.

1.2.4 - COMANDOS DE SISTEMA

1.2.4.1 - AJUDE

FG: AJUDE <NOME DO COMANDO>

AJU

Este comando tem como objetivo informar ao usuário como ter acesso aos recursos do sistema, através da linguagem de comandos. Quando um nome de comando é mencionado, são dadas as informações sobre este comando, tornando explícito ao usuário sua utilização e principais funções. Para que informações sobre o sistema sejam listadas, o usuário deverá teclar "ajude sistema".

1.2.4.2 - ASSOCIE

FG: ASSOCIE "NOME-LÓGICO ARQUIVO NOME - DA-TAREFA"

ASS

O comando "ASSOCIE", associa a um arquivo um nome lógico que define em qual periférico o arquivo se encontra. Essa associação libera a responsabilidade de ter-se controle do periférico onde o arquivo está armazenado, fazendo com que a referência seja feita através do nome lógico associado. A associação pode ser feita somente para uma tarefa, identificada pelo nome mencionado no comando. Quando no lugar da tarefa coloca-se um "$" (dolar) a associação é feita a nível de sistema, sem nenhuma vincu

lação a qualquer tarefa.

1.2.4.3 - DATA

FG: DATA

DA

Este comando informa a hora e a data no formato hh:mm:ss, DD/MM/AA.

1.2.4.4 - OI

FG: OI

O objetivo desse comando é tornar o sistema disponível somente para usuários autorizados. Este comando, através de uma senha dada pelo usuário, verifica a autorização do mesmo. Caso seja um usuário habilitado, o sistema fica disponível, assim como a linguagem de comando para o acesso aos recursos. Após a aceitação do usuário pelo sistema, serão solicitados o NÓ, o VOLUME e o DIRETÓRIO a serem utilizados, os quais serão considerados "defaults" para qualquer transação usuário/sistema, durante a atual sessão.

1.2.4.5 - TCHAU

FG: TCHAU

TCH

Este comando finaliza a sessão de um usuário, ficando o sistema disponível para outro usuário , através do comando "OI".

1.2.4.6 - SUSPENDE

FG: SUSPENDE "NOME-DA-TAREFA"
SUS

Uma vez que uma tarefa pode ser executada através de um comando, qualquer tarefa pode ser sus - pensa pelo usuário, desde que ele saiba o nome da mesma. O comando "suspende" cancela uma tarefa suprimindo os resultados da execução da mesma, e informando ao usuário sua ocorrência.

1.2.4.7 - SISTEMA

FG: SISTEMA <NOME-DA-TAREFA>

O comando "SISTEMA" tem como objetivo informar ao usuário a situação atual de todas as tarefas ativas do sistema. Caso uma tarefa seja mencionada, é in - formada a situação do sistema naquele momento, referente à quela tarefa, a qual tem o nome informado no início de sua execução.

1.2.5 - COMANDOS DE PROGRAMA

1.2.5.1 - EXECUTE

FG: EXECUTE <NOME-DO-ARQUIVO-OBJETO>
EXE

Neste comando o nome do arquivo é op - cional devido ao fato de ser pesquisada a existência do arquivo de trabalho, caso o nome do arquivo seja omitido .

Se o nome do arquivo não aparecer, é executado o código "trab.exe". Neste caso se o mesmo não existir, é chamado o compilador apropriado, identificado pela extensão do arquivo. No caso da última compilação ter dado erro ou alguma advertência, o código não é executado e as mensagens são apresentadas ao usuário. Caso o nome do arquivo seja mencionado, é suposta a existência do arquivo em código objeto e a execução é solicitada, sendo o usuário informado da existência de algum erro.

1.2.5.2 - SINTAXE

FG: SINTAXE

SIN

Este comando tem como único objetivo testar a sintaxe do arquivo de trabalho, identificado por "trab.txt". As mensagens de erro são geradas, caso existam, e o código objeto não é gerado.

1.2.5.3 - DEPURE

FG: DEPURE

DEP

O comando "DEPURE" proporciona a execução do programa de trabalho assistida pelo depurador do sistema, podendo o usuário verificar os resultados da execução de uma instrução ou conjunto de instruções, através das opções deste comando.

1.2.6 - COMANDOS DE ARQUIVO

1.2.6.1 - SELECIONE

FG: SELECIONE "NOME-DE-ARQUIVO"
SEL

Para que um usuário edite, compile ou apenas cheque a sintaxe de um arquivo, deverá selecionar o arquivo desejado através deste comando.

O arquivo a ser selecionado é copiado para um arquivo temporário, "trab.txt", caso já não exis - ta. No caso da sua existência, o usuário é interrogado , e decisões serão propostas pelo sistema.

1.2.6.2 - APAGUE

FG: APAGUE <NOME-DE-ARQUIVO>
APA

O comando "APAGUE" deleta do sistema o arquivo mencionado. Caso este seja omitido, o usuário é interrogado pelo sistema para que a deleção do arquivo de trabalho não seja feita indevidamente, já que sua referência é assumida, diante da omissão do nome do comando.

1.2.6.3 - EDITE

FG: EDITE
EDI

O comando "EDITE" proporciona ao usuário a alteração de informações contidas no arquivo de trabalho, previamente selecionado pelo usuário.

Caso o arquivo não exista, o sistema informará ao usuário para que o mesmo "crie" ou "selecio - ne" um novo arquivo.

1.2.6.4 - DIRETÓRIO

FG: DIRETÓRIO <NOME-DE-ARQUIVO>
DIR

Este comando informa ao usuário, todos os arquivos existentes no seu diretório. Caso um nome seja mencionado o sistema informará a existência ou não daquele arquivo , podendo ainda informar sobre todos os arquivos irmãos.

1.2.6.5 - COPIE

FG: COPIE "NOME-DE-ARQ1"
"NOME-DE-ARQ2" <PARAM.>
COP

Este comando proporciona ao usuário criar um novo arquivo de nome "nome-de-arquivo2", com as mesmas informações do arquivo original "nome-de-arquivo1", podendo este ser o arquivo de trabalho. A transferência pode se dar entre dois periféricos diferentes. É permitida a conversão de arquivos de registros de tamanho fixo em variável e vice-versa. Para isso, o campo <PARAM.> pode assumir os seguintes valores:

a. V - Arquivo terá tamanho variável - (brancos suprimidos).

b. FTN - Arquivo terá tamanho fixo

(=n), onde os registros cujos tamanhos forem maiores que "n" serão truncados.

c. FN - Arquivo terá tamanho fixo (=n), onde os registros cujos tamanhos forem maiores que "n" terão seu restante num novo registro.

1.2.6.6 - MUDE

FG: MUDE "NOME-DE-ARQUIVO1"
"NOME-DE-ARQUIVO2"

MUD

O comando "MUDE" troca o nome do arquivo mencionado, "nome-de-arquivo1", para "nome-de-arquivo2". Se o primeiro nome de arquivo for o de trabalho, um novo arquivo deve ser selecionado, caso necessário.

1.2.6.7 - SALVE

FG: SALVE <NOME-DE-ARQUIVO>

SAL

No comando "SALVE", o arquivo de trabalho é salvo em disco com o nome do último arquivo selecionado. Isto ocorre se o nome for omitido no comando, caso contrário o arquivo de trabalho é salvo com o nome mencionado "nome-de-arquivo". Qualquer ocorrência de duplicação de arquivo, o sistema informa ao usuário para que decisões sejam tomadas.

1.2.6.8 - LISTE

FG: LISTE <NOME-DE-ARQUIVO>

LIS

Este comando lista no vídeo o arquivo mencionado. Caso o nome do arquivo seja omitido é listado o arquivo de trabalho.

1.2.6.9 - CRIE

FG: CRIE "NOME-DE-ARQUIVO"

CRI

Este comando cria um novo arquivo em disco, ou em outro periférico anteriormente associado, habilitando o usuário a teclar as informações de entrada. Caso ocorra duplicação de arquivo, o usuário receberá informações a respeito.

1.2.6.10 - SUBMETA

FG: SUBMETA "NOME-DO-ARQUIVO-LICO"<NO>

SUB

Este comando faz com que todos os comandos contidos num arquivo LICO, anteriormente criado pelo usuário, sejam executados um por um. Caso um "nó" seja mencionado, os comandos serão executados no computador cuja identificação dentro da rede é o número especificado.

1.2.7 - COMANDOS DE CONTROLE

Os comandos de controle têm como função controlar operações referentes a tarefas que utilizam o vídeo como saída

de informações. Esses comandos têm sua entrada feita através de uma tecla especial que, uma vez teclada, gera a operação de parada ou execução do módulo a ser listado.

1.2.7.1 - <PL>

Interrompe a saída de informações pelo vídeo, sendo impressa à última linha enviada pelo sistema.

1.2.7.2 - <CL>

Dá continuidade à saída de informações referente à tarefa interrompida pelo comando <PL>.

CAPÍTULO II

2. E S T R U T U R A

2.1 - INTRODUÇÃO

O SIRIUS, desenvolvido para gerenciar a comunicação usuário/sistema, é uma rotina composta por módulos executáveis que são invocados de acordo com o comando teclado pelo usuário. Para que esses módulos sejam executados, é necessário que os comandos de chamada sejam submetidos a tratamentos feitos pela "LICO" e, caso seja necessário, o sistema interroga ou informa ao usuário dados complementares.

Tomando-se como base o caráter didático desta linguagem, é definido um sistema composto por arquivos de dados que contêm informações referentes aos comandos da linguagem e especificações do sistema.

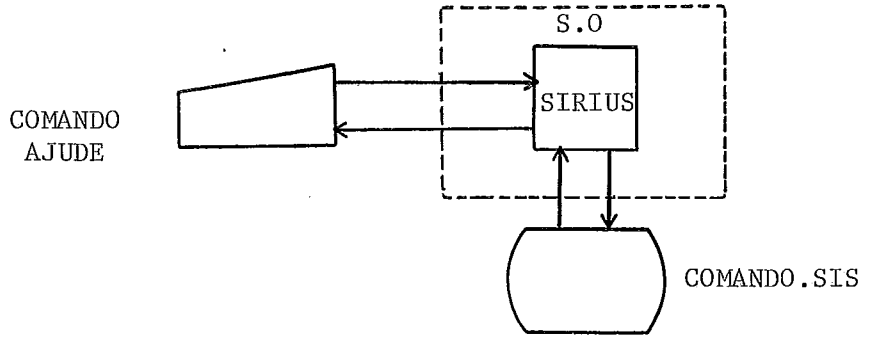


FIG. 2

Esses arquivos são enviados ao usuário, quando solicitados pelo mesmo, através do comando "ajude", e sua identificação é feita pelo nome do comando adcionado à extensão - "sis", o que define a utilização do arquivo só pelo sistema.

2.2 - FUNCIONAMENTO

Para que um usuário tenha acesso aos recursos do computador, é necessário que o mesmo seja um usuário habilitado. Com isso, qualquer usuário para abrir uma "sessão de utilização", deverá teclar, como primeiro comando, o comando "OI", o que implicará num pedido de senha feito pelo sistema. Quando a senha é teclada pelo usuário, o sistema verifica-a, abrindo a sessão caso seja um usuário autorizado. Uma vez que a sessão é aberta, o usuário tecla os comandos desejados, fechando no final com o comando "TCHAU".

Para que seja seguido o fluxo de utilização acima descrito, o SIRIUS foi dividido em módulos que processam as informações, segundo uma chamada hierárquica de processos, desde a entrada do usuário pelo comando "OI", até a sua saída pelo comando "TCHAU".

O sistema SIRIUS é composto por 5 módulos: O Manipulador, o Gerente, o Mestre, o Secundário e o Impressor.

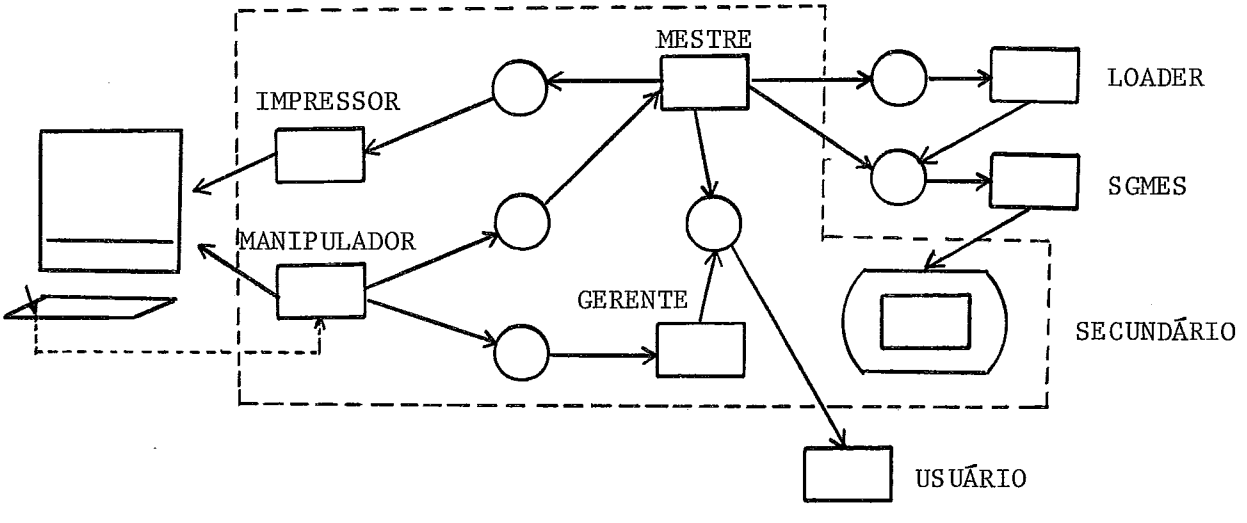
Quando um usuário tecla um caracter, este é recebido pelo "Manipulador" que o imprime no vídeo, guardando-o para que seja montado um comando o qual, posteriormente, será enviado para o módulo MESTRE ou módulo GERENTE, dependendo da informação teclada. O módulo MESTRE, por sua vez, interpreta o comando e invoca a rotina adequada, contida no módulo secundário, a fim de executar a tarefa solicitada pelo usuário. Em certos casos é feita a chamada de subrotinas do núcleo sem ser necessária a chamada de rotinas do módulo secundário.

O módulo MESTRE é composto pelas rotinas de abertura, tratamento e fechamento, e o módulo secundário é composto

pelos "módulos executáveis", referentes a determinados comandos. O módulo MESTRE, módulo residente, é responsável pelo tratamento das informações recebidas do Manipulador. No caso, a rotina de abertura recebe o comando "OI" do usuário, verifica sua autorização, abrindo a sessão caso a mesma seja positiva, e passando o comando do sistema para a rotina de tratamento. Uma vez que a sessão é aberta, a rotina de tratamento recebe os comandos, passando o comando do sistema para a rotina de "fechamento", caso o comando teclado seja o comando "TCHAU".

No módulo SECUNDÁRIO, encontram-se os MÓDULOS EXECUTÁVEIS que são invocados pela rotina de tratamento, após o recebimento e interpretação do comando. Esses módulos são referentes a alguns comandos que, de acordo com sua interpretação, geram um pedido de carga e execução ao monitor, sendo enviadas ao usuário as informações resultantes de sua execução, através do módulo "impressor" do SIRIUS. Por fim, o módulo GERENTE fica responsável pelo gerenciamento de entrega e recebimento de informações aos processos que têm o teclado como um arquivo de entrada.

A estrutura do SIRIUS é mostrada na figura abaixo:



2.3 - MANIPULADOR

O MANIPULADOR, módulo residente, é ativado através de interrupção gerada quando o usuário tecla um caracter.Como já descrito no ítem "MEIOS DE UTILIZAÇÃO", existem duas janelas que compoem o vídeo onde são impressas as informações do usuário e do sistema. Independente da janela onde esteja o cursor, o caracter é recebido pelo MANIPULADOR, onde é definida uma informação para posterior tratamento. Baseando-se no posicionamento do cursor, definem-se duas situações de impressão do caracter: o cursor na janela inferior, e o cursor na janela superior.

- CURSOR NA JANELA SUPERIOR

Como já descrito no ítem "MEIOS DE UTILIZAÇÃO" , os comandos teclados pelo usuário são impressos na útlima linha disponível da janela superior, causando o remanejamento dos demais, caso a janela esteja cheia. Com isso, durante a entrada de informações, o usuário poderá utilizar um comando anteriormente teclado que esteja sendo mostrado na janela superior. Para isso, o mesmo deverá direcionar o cursor a fim de atingir o comando desejado.

Se o processo impressor do SIRIUS estiver utilizando o vídeo, isto é, dando saída a informações resultantes de um pedido, o cursor não poderá ser direcionado para a janela superior, evitando-se, assim, a impressão de dados sobre a saída. Neste caso, o MANIPULADOR habilita o usuário a utilizar somente a janela inferior, onde serão impressos cada caracter teclado , segundo ítem anterior. No caso do vídeo não estar sendo utilizado, o cursor poderá ser direcionado para a janela superior, e

os caracteres teclados impressos de forma independente da posição do mesmo.

Durante a utilização da janela superior pelo MANIPULADOR, esta fica bloqueada para os impressores, evitando a impressão de novas informações sobre as já tecladas pelo usuário. Durante a utilização da janela superior pelo impressor do sistema, esta não fica bloqueada para o MANIPULADOR. Desta forma o MANIPULADOR poderá alterar informações de saída do impressor do sistema, o que define operações do editor de textos.

2.3.1 - COMPARTILHAMENTO DE VÍDEO

Durante o processamento, informações são impressas no vídeo, podendo estas ser: Listagem de arquivo, resposta de execução de tarefa, informações complementares do sistema , etc. Essas informações são enviadas por três processos que compartilham o vídeo: O Manipulador de entrada, o processo Impressor do SIRIUS e o processo Impressor do sistema.

A impressão dos dados, realizada pelos três processos, é definida de forma que durante a utilização de uma das janelas, pelo Impressor do SIRIUS ou pelo Manipulador, a mesma fique bloqueada para os outros dois, e no caso do Impressor do sistema, a janela utilizada fique bloqueada somente para o SIRIUS.

Para isso, existe um sistema de semáforos que estabelece a sincronização entre os processos durante a utilização da janela superior ou inferior [4]. O esquema de funcionamento segue as operações de ativação e desativação, sendo descrito a seguir.

2.3.2 - FUNCIONAMENTO

Existem três semáforos, chamados S1, S2 e S3, que controlam a utilização das janelas pelos processos. O semáforo "S1" estabelece a sincronização entre os processos de impressão do sistema e do SIRIUS, referente a janela superior. O "S2" sincroniza o Impressor do SIRIUS e o Manipulador, durante a utilização da janela superior. E o "S3" sincroniza o Impressor do SIRIUS e o Manipulador, durante a utilização da janela inferior.

Quando o processo impressor do sistema tem informações a imprimir, o semáforo "S1" é testado e, caso esteja ativado, o processo deposita o seu número de identificação num campo auxiliar e fica esperando posterior ativação pelo processo que utiliza a janela, através do "núcleo". O mesmo ocorre com o Impressor do SIRIUS quando este vai utilizar a janela superior.

Além do teste do "S1", o SIRIUS testa o "S2", depositando seu número num campo auxiliar e ficando a espera de uma posterior ativação, caso o semáforo esteja ativado.

O Manipulador, por sua vez, só aceita o caracter caso a janela de utilização esteja desbloqueada. O estado de espera do Manipulador é imediato, já que sua ativação é feita via interrupção.

A ativação dos processos em espera é feita através do campo auxiliar, ou seja, caso exista um número depositado, o processo pede a ativação do mesmo ao "núcleo" e limpa o campo para posterior utilização. Note que o Manipulador poderá utilizar a janela superior, juntamente com o impressor do sistema, podendo ser feita a alteração de informações da janela supe

rior, tendo essas sido impressas pelo processo impressor do sistema.

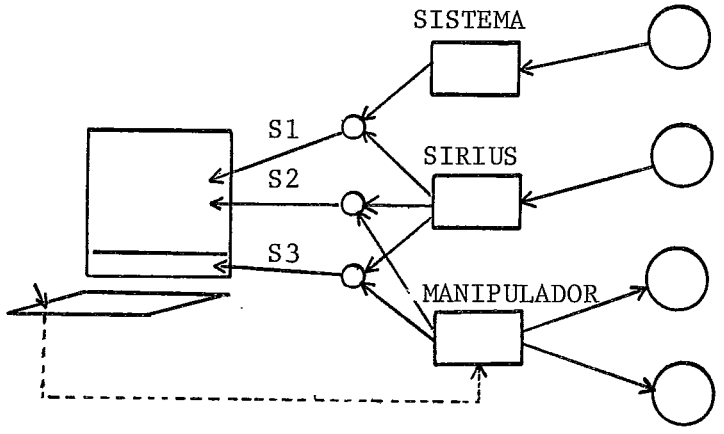


FIG. 4

O teste do semáforo "S2", só é feito pelo Manipulador caso o cursor seja direcionado de forma que tente ultra - passar a linha divisória das janelas. A utilização do vídeo pelos processos segue o funcionamento dos algoritmos a seguir [4].

```
/*  UTILIZAÇÃO DO IMPRESSOR DO SISTEMA */

    1. SE S1 = 0 ENTÃO FAZ S1 = 1 ;
                           UTILIZA ;
                           S1 = 0 ;
                           ATIVA-ESPERA ;
                    SENÃO FICA ESPERANDO ;


/*  UTILIZAÇÃO DO IMPRESSOR DO SIRIUS  */

    1. SE S1 = 0
            ENTÃO SE S2 = 0
                  ENTÃO FAZ S1 = 1;  S2 = 1
                           UTILIZA ;
                           S1 = 0;  S2 = 0
                           ATIVA-ESPERA ;
```

```
                    SENÃO FICA ESPERANDO ;
            SENÃO SE S3 = 0
                    ENTÃO FAZ S3 = 1 ;
                              UTILIZA ;  /* INF */
                              S3 = 0 ;
                              ATIVA-ESPERA ;
                    SENÃO FICA-ESPERANDO ;

/*  UTILIZAÇÃO DO MANIPULADOR  */

    1. SE "JANELA SUPERIOR"
       ENTÃO SE S2 = 0 ENTÃO FAZ S2 = 1 ;
                                 UTILIZA ;
                                 S2 = 0 ;
                                 ATIVA-ESPERA ;
       SENÃO SE S3 = 0 ENTÃO FAZ S3 = 1 ;
                                 UTILIZA ;
                                 S3 = 0 ;
                                 ATIVA-ESPERA ;
                  SENÃO "NÃO-ACEITA" ; /* USO TEMP */
```

A impressão dos dados na janela superior tem como referência um cursor transparente ao usuário, cujo controle é feito pelo último impressor a utilizá-lo. Na janela inferior é utilizada a última linha disponível sendo esta a linha de entrada do usuário.

2.3.3 - TRATAMENTO

Quando uma informação é recebida, a mesma pode ser entregue ao módulo MESTRE ou ao GERENTE de tecla

do, dependendo do tipo de mensagem que foi requisitada pelo processo em espera. Os caracteres de controle são tratados de maneira que não sejam enviados para o monitor. Uma vez que esses caracteres não têm efeito sobre as informações a processar, os mesmos só são utilizados pelo Manipulador que dá o tratamento adequado. Além dos caracteres de controle existem outros que não são impressos pelo Manipulador e resultam nas definições do ítem "MEIOS DE UTILIZAÇÃO". Quando o Manipulador recebe um caracter "fim-de-comando", uma decisão é tomada e, de acordo com um parâmetro, a informação é entregue a um módulo através do monitor que executa a tarefa adequada. Para cada "fim-de-comando" o cursor é liberado para que seja dada entrada a um novo comando.

2.3.4 - MANIPULAÇÃO DAS INFORMAÇÕES

Como já descrito no ítem "MEIOS DE UTILIZAÇÃO" , o vídeo é composto por 30 linhas, cada uma com 80 colunas, onde são impressos os caracteres. Das 30 linhas, 26 são utilizadas pela janela superior, 3 pela janela inferior e 1 pela linha divisória das janelas. A impressão dos dados no vídeo consiste na projeção de uma matriz 30x80, onde são depositados os caracteres pelos processos que utilizam o vídeo. A passagem dos dados entre o Manipulador e o Monitor é realizada através de subrotinas do núcleo, onde é feito o preenchimento do primeiro buffer disponível da fila de requisições.

Para cada caracter é feito um tratamento, e só após a identificação do tipo da informação a passagem é iniciada. O tratamento dado ao caracter consiste na identificação do mesmo, como um caracter de controle do cursor, um fim-de-comando

ou qualquer outro caracter que resulte num procedimento a ser executado pelo Manipulador, antes da informação ser enviada para o monitor. A linha divisória das janelas é protegida de forma que o usuário não deposite caracteres sobre a mesma. Quando é recebido um caracter de controle do cursor, no caso um caracter de direcionamento, o Manipulador altera a linha ou coluna referente à posição do cursor no vídeo. Isto é feito para que o GERENTE de teclado receba os fatores linha/coluna das informações a serem tratadas, caso a passagem esteja à nível de caracter.

Nota-se que o papel do Manipulador resume-se em receber os dados diretamente do usuário e enviá-los através do monitor para o módulo em questão que, de acordo com suas funções, define o tratamento adequado para as informações contidas no buffer. Quando um pedido de leitura é depositado no GERENTE de teclado, o mesmo atualiza o parâmetro do Manipulador, para que o tipo da mensagem a receber seja identificado, antes da passagem da informação feita pelo Manipulador. As informações utilizadas pelo Manipulador estão contidas numa "BCES" (bloco de controle de entrada e saída), depositadas pelo processo solicitante. A utilização do vídeo por um processo é feita através do monitor, onde é invocado o processo de impressão desejado. A escolha do módulo adequado para impresssão é definida pela natureza do processo dentro do sistema. Com isso, processos que têm o vídeo como um arquivo, depositam pedidos no módulo de impressão do sistema, onde o vídeo é bloqueado durante toda a sua utilização. Por outro lado, processos que utilizam o vídeo como um meio informante, ou seja, requisitam a impressão de perguntas, respostas ou informações complementares de mensagens, utilizam o módulo de impressão da SIRIUS que bloqueia o vídeo somente durante a

impressão da mensagem. Esses processos que utilizam o impressor do SIRIUS, normalmente são processos resultantes de um comando.

2.3.5 - DECISÃO E ENTREGA DOS DADOS

O Manipulador é também responsável pela entrega das informações tecladas ao processo adequado. Esse processo pode ser a LICO que recebe comandos ou qualquer programa que tenha sua "entrada por teclado". As informações são recebidas do usuário e são tratadas de acordo com certos parâmetros internos ao Manipulador. Esses parâmetros, contidos na BCES depositada pelo processo solicitante, são definidos de forma que identifiquem a informação teclada como sendo uma entrada de dados ou um comando para a LICO. Quando uma BCES não é encontrada pelo Manipulador, o mesmo fica em estado de espera, mesmo que informações tenham sido tecladas.

2.3.6 - COMANDOS E/OU ENTRADAS DA LICO

Qualquer informação teclada é dirigida para a LICO, a menos que exista um pedido de entrada por teclado. Nesse caso, a informação deve ser prescedida pela tecla <CMD> que as dirige para a LICO. Essas informações podem ser um comando ou resposta a informações complementares solicitadas pela LICO. Com essa condição, o Manipulador deposita os dados num buffer onde , posteriormente, são tratados pelo módulo MESTRE. Além dos dados serem identificados como entrada ou comando, os mesmos são tratados de forma que sejam enviados para o módulo RECEPTOR, à nível de caracter ou à nível de registro. No caso da LICO, qualquer informação é recebida à nível de registro, sendo identificadas após o usuário pressionar a tecla <ENTRE>.

Quando o Manipulador recebe um "fim-de-comando" , o mesmo prepara os dados, acessando a linha da matriz em questão, e identificando as informações a enviar para o módulo MESTRE. Toda informação a tratar encontra-se numa linha da matriz, entre o caracter identificador do começo-de-comando e a coluna 80. No caso de não haver um começo-de-comando, o Manipulador trata a informação a partir da primeira posição da linha da matriz.

Como pode ser notado, o tamanho máximo da mensagem é de 80 caracteres, o que define o tamanho do buffer do módulo MESTRE. A mensagem enviada corresponde ao conteúdo da linha da matriz apontada pela posição corrente do cursor. Os comandos da LICO que controlam a impressão de dados, o <PL> e <CL>, são tratados pelo Manipulador de forma que não sejam enviados para o monitor. Quando o Manipulador recebe um <PL>, o impressor do sistema é interrompido através de um pedido de desativação feito ao núcleo (chamada a subrotina "espera-física"). No caso do recebimento do <CL>, um pedido de ativação é feito e o impressor acordado, dando continuidade à impressão (chamada a subrotina "ativa").

2.3.7 - COMANDOS E/OU ENTRADAS DE PROCESSOS

As informações, quando não prescedidas pela tecla <CMD>, são tratadas como "entrada" de algum processo pedinte. Isso implica que, além da LICO, existe um outro processo que recebe informações do Manipulador. Esse processo controla as entradas de dados de processos que têm o teclado como um arquivo de entrada. Os pedidos de entrada são depositados na fila desse processo que fica esperando informações do Manipulador, ativando um dos seus parâmetros. A partir desse ponto, todas as informações

tecladas pelo usuário são dirigidas para o buffer desse processo (gerente), a menos que sejam prescedidas por um <CMD>, o que as dirige para a LICO (Módulo MESTRE).

O Manipulador, ao receber informações do usuário, verifica o formato da mensagem a enviar, para que sejam recebidas pelo gerente de entrada à nível de caracter ou à nível de registro. No caso do usuário entrar com informações para programas, as mesmas são entregues no gerente de entrada que fará o tratamento de acordo com seu estado. Baseando-se nesse sistema, o usuário deve teclar as informações de maneira que as mesmas sejam recebidas pelo processo adequado.

2.3.8 - TRATAMENTO A MENSAGENS

Quando um caracter é recebido do usuário, o Manipulador consulta o parâmetro identificador do formato da mensagem para decidir se espera ou não um fim-de-mensagem. Após essa decisão o caracter é entregue ao gerente de entrada ou, após o recebimento do "fim-de-mensagem", o Manipulador prepara a linha da matriz em questão para posterior tratamento. Essa preparação implica na identificação do tamanho da mensagem a ser enviada, definida pelo caracter começo-de-mensagem e a coluna 80. Nesse momento, a linha está pronta para ser tratada de forma que a mensagem seja depositada no módulo apropriado.

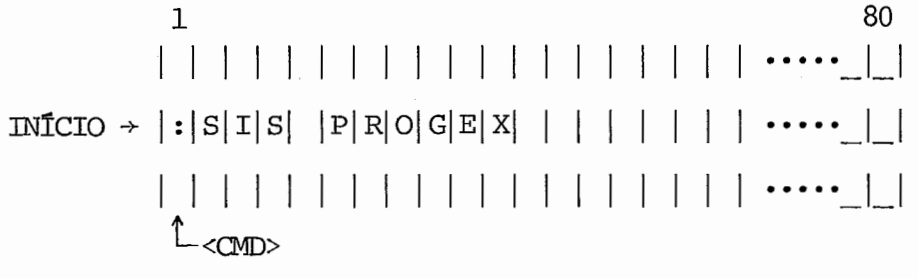


FIG. 5

2.4 - MÓDULO GERENTE

O módulo GERENTE é responsável pelo gerenciamento do teclado quando o mesmo é utilizado como um arquivo de entradapor um processo do usuário ou do sistema. Quando um usuário deseja utilizar o teclado como um arquivo de entrada, o mesmo deverá depositar um pedido de abertura na fila do módulo GERENTE. Daí por diante o arquivo fica bloqueado, podendo ser utilizado somente por esse usuário, até que o mesmo deposite um pedido de fechamento desse arquivo. Os pedidos depositados na fila são acessados pelo GERENTE através de subrotinas do núcleo, "retire" e "deposite", sendo os mesmos atendidos de acordo com sua ordem de chegada. Além dos pedidos de abertura e fechamento,os pedidos de leitura de dados são atendidos de forma que as informações a serem lidas sejam entregues pelo "Manipulador".

O módulo GERENTE é uma rotina cíclica que funciona como um "serviço" para o sistema. Isto é, o módulo GERENTE é identificado pelo sistema como sendo um serviço disponível , o qual está associado ao arquivo "tecla". Quando um usuário deseja utilizar este arquivo, o mesmo pede uma abertura onde é devolvida uma "BCES" preenchida contendo o número do serviço de teclado, no caso, o GERENTE. Logo após, a "BCES" é depositada na fila de requisições do GERENTE, onde é feita a abertura do arquivo TECLA.

No sentido geral, o GERENTE tem as mesmas funções que qualquer outro serviço do sistema, resumindo-se em atender aos pedidos pendentes na sua fila de requisições. Quando um pedido é feito, o GERENTE identifica o usuário requisitante, através da "BCES", e avisando ao Manipulador que o arquivo está aberto e qual o tipo de entrada que deve ser feita, podendo ser à

Nível de caracter ou à nível de registro (entre). A partir daí, sempre que uma informação for enviada para o GERENTE, o Manipulador as deposita num buffer pedindo, em seguida, a ativação do GERENTE. Por sua vez, o GERENTE acessa as informações no buffer entregando-as ao usuário por quem o arquivo foi aberto.

Algumas situações devem ser consideradas para que seja alcançado um perfeito funcionamento do GERENTE. Nota-se que o GERENTE pode ser ativado por um processo do usuário, através de um pedido, ou pelo Manipulador. Pode ocorrer o caso de uma leitura ser solicitada e não haver informações depositadas no buffer. Com isso o GERENTE seria desativado e ficaria esperando uma ativação do Manipulador. Outro caso seria uma ativação feita pelo Manipulador sem que houvesse nenhum pedido de leitura depositado. Nesse caso o gerente seria ativado pelo processo do usuário quando houvesse um pedido. Para que a execução do GERENTE independa do processo que o ative, define-se sua ativação somente no caso de havér um pedido de leitura e informações no buffer, depositadas pelo Manipulador. Ou seja, sempre que uma ativação for feita, seja por um processo do usuário ou pelo Manipulador, é testada a existência de pedido e de informação, caso contrário o GERENTE fica desativado. Com esta regra de ativação, o sistema fica confiável, uma vez que a informação de entrada só é entregue a um mesmo usuário, isto é , o que abriu o arquivo.

2.5 - MÓDULO MESTRE

O módulo MESTRE é responsável pela interpretação dos comandos e execução da tarefa solicitada pelo usuário. Os comandos teclados são depositados pelo Manipulador na fila de

requisições do módulo MESTRE, onde são atendidos segundo sua ordem de chegada.

Esse módulo é composto por três rotinas que são executadas de acordo com a fase de utilização de um usuário.São elas: A ROTINA DE ABERTURA, responsável pela inicialização do sistema; ROTINA DE TRATAMENTO, responsável pela execução das tarefas e a ROTINA DE FECHAMENTO, responsável pela finalização do sistema.

2.5.1 - ROTINA DE ABERTURA

A ROTINA DE ABERTURA, "sistem.abre.exe", é uma rotina cíclica e residente que é carregada juntamente com outros módulos do sistema operacional quando é dada a carga do sistema pelo usuário.

A rotina de abertura recebe o comando do usuário, e caso este não seja o comando "OI", é devolvida ao MESTRE uma mensagem para que seja teclado o comando adequado.

Quando o usuário tecla o comando "OI", a rotina pede uma senha, "senha do sistema", para verificar se é ou não autorizada sua entrada. Uma vez que o sistema só é aberto pela rotina de abertura, as rotinas de tratamento e fechamento ficam em disco, até que seja dada a abertura do sistema. Isto ocorre quando a senha correta é teclada pelo usuário e a rotina de abertura carrega em memória o restante do módulo MESTRE, habilitando o usuário a entrar com os comandos desejados.

Além da senha, esta rotina pede a entrada de informações indispensáveis para a utilização do usuário. São elas: O disco "default" a ser utilizado, sendo assumido o de "sistema" caso não haja informaçõa, e ainda o diretório a ser

utilizado. Essas informações são passadas para o diretor do disco, para posterior tratamento das mesmas.

A carga das rotinas de tratamento e fechamento é feita pelo módulo carregador através de um pedido ao monitor, onde as rotinas são carregadas sobre a rotina de abertura, já que a mesma só será utilizada na abertura da próxima sessão, sendo porém carregada no final da sessão atual pela rotina de fechamento. Antes desta rotina carregar o restante do módulo, são dadas informações gerais sobre o sistema, informações estas que estão contidas num arquivo de dados do SIRIUS. O restante do módulo é carregado e fica esperando uma chamada feita pelo monitor ao módulo MESTRE.

2.5.2 - ROTINA DE TRATAMENTO

A rotina de tratamento tem como função interpretar os comandos teclados pelo usuário e invocar a rotina adequada para a execução da tarefa solicitada. A rotina de tratamento é ativada pelo monitor o qual envia o primeiro comando da fila de atendimento. De acordo com a classificação do comando dentro da LICO, uma função é executada e a rotina retorna ao estado de espera de uma nova ativação feita pelo monitor.

Classificação dos comandos quanto à diretiva:

- COMANDO DIRETO

Um comando é dito "DIRETO" quando é um comando "execute", isto é, quando não é necessária uma interpretação, já que o mesmo se encontra na forma direta de pedir a execução de uma tarefa. Comandos que têm essa forma geram um pedido de carga e execução do módulo mencionado, após a palavra "execute". Esse pedido é depositado no módulo carregador (loader), onde é feita a avaliação da existência, ou não, do módulo executável mencionado no comando. No caso da inexistência do mesmo, o módulo carregador tomará as decisões necessárias para a compilação do arquivo de trabalho ou envio de informações ao usuário sobre a tarefa solicitada. A avaliação é feita através do recebimento de um "status" enviado pelo diretor de disco após o pedido de abertura do arquivo mencionado (veja LOADER).

- COMANDO INDIRETO

Um comando é dito "INDIRETO" quando não se encontra na forma direta. Neste caso é necessária uma interpretação para que seja definido o tipo do comando dentro de sua classificação. A interpretação de um comando é composta por duas fases: Na primeira é testada a validade do comando dentro da linguagem, onde é identificada sua existência ou não, dentro do grupo de comandos que compoem a linguagem; na segunda fase, é definida sua "derivada", ou seja, de acordo com a classe do comando (gerador ou sinônimo) é definida uma função a ser executada.

Para que a interpretação seja feita, define-se u

ma "tabela de comandos" na qual conste todos os comandos da linguagem, juntamente com seu tipo, derivada e rotina de formatação a ser chamada.

Na primeira fase, uma busca é feita na tabela de comandos sendo testada a existência do mesmo. Na segunda fase o tipo do comando é testado, podendo ser feita uma chamada à subrotina responsável pela formatação e depósito no serviço requisitado, ou um novo comando é interpretado.

TABELA DE COMANDOS

COMANDO	T	DERIVADA	E
AJUDE	8	? AJUDE.SIS	5
-	-	-	-
-	-	-	-
-	-	-	-

FIG. 6

Os comandos indiretos dividem-se em duas classes de comandos: COMANDOS SINÔNIMOS e COMANDOS GERADORES.

- COMANDOS SINÔNIMOS

Um comando é dito SINÔNIMO quando sua derivada é um novo comando a ser interpretado. Nesse caso a classe do comando é identificada pelo seu tipo, os parâmetros são montados e a derivada é novamente avaliada, até que resulte num comando "gerador".

- COMANDOS GERADORES

Um comando é GERADOR quando resulta num depósito de pedido, em qualquer módulo disponível no sistema. Uma vez que o tipo desses comandos é identificado, é feito um desvio para o endereço constante na tabela, onde são montados os parâmetros em função da derivada, e em seguida é feito um depósito no serviço responsável pela tarefa requisitada. Durante a interpretação existe a "montagem" de parâmetros, que é a atualização de identificadores constantes no campo "derivada" de cada comando.

- MONTAGEM

Os parâmetros de uma tarefa podem ser opcionais ou obrigatórios, dependendo da função do comando teclado. Para que o comando seja devidamente avaliado, definem-se dois tipos de caracteres constantes na derivada, que representam a natureza da informação a ser recebida pelo sistema. Os parâmetros obrigatórios são representados pelo caracter '|', e devem ter o mesmo número de ocorrências, tantas quantas apareçam no campo "derivada". Para cada ocorrência um '|' na derivada, deverá corresponder um parâmetro no comando teclado. Cada identificador(|), é substituído pelo parâmetro correspondente, resultando numa nova informação a ser tratada pelo módulo.

O segundo identificador, representado pelo caracter '?', implica na ocorrência "opcional" do parâmetro no comando teclado. Para cada identificador (?) é feita uma substituição pelo parâmetro correspondente constante no comando. Caso o correspondente não exista, o identi-

ficador é substituído pelo branco (' '), o qual terá efeito no tratamento da informação resultante.

Havendo qualquer inconsistência dos dados, um pedido é depositado no módulo impressor do SIRIUS, onde o usuário é informado da ocorrência do erro.

- FORMATAÇÃO E CHAMADA

Após a fase de montagem, a rotina volta a interpretar o comando resultante, no caso de sinônimos,ou faz a chamada de subrotinas contidas no núcleo. Essas subrotinas têm uma forma própria de receber as informações,fazendo com que antes da sua chamada seja feita sua formatação. Com isso, os seguintes passos são necessários para a execução desses comandos: Busca na tabela de comandos para testar a validade, montagem dos parâmetros,identificação do seu tipo e, finalmente, a formatação das mensagens e chamada da subrotina.

A formatação e pedido de execução estão contidos em procedimentos da própria rotina de tratamento, onde a identificação do início dos mesmos é dada pelo endereço contido no campo 'E' da tabela. Com isso, logo após a identificação é feito o desvio e os procedimentos executados.

- ARQUIVOS LICO

Como já descrito no ítem "tratamento de arquivos", existem arquivos compostos por um conjunto de comandos definidos previamente pelo usuário. A execução

desses comandos é feita através da interpretação de um comando gerador, o qual tem sua derivada, um pedido de carga e execução do módulo executável, SUBMET.EXE, res - ponsável pela leitura desses comandos.

Esse módulo tem como entrada o arquivo mencionado no comando e terá seu término quando encontrar o fi - nal do arquivo a ser lido. Para cada registro lido, isto é, comando, um pedido com reativação é depositado no módulo MESTRE, segundo o fluxo normal de pedidos e atendimento, semelhante ao Manipulador. No caso de ocorrer um novo comando de arquivo LICO, um novo módulo é carregado e as tarefas empilhadas, de forma que para cada final de arquivo LICO seja ativada a tarefa em espera da qual originou-se a execução do atual módulo.

2.5.3 - ROTINA DE FECHAMENTO

A rotina de fechamento é responsável pela fina - lização do sistema, ou seja, fecha a sessão de utilização de um usuário. A rotina de fechamento é um módulo executável, ativado pelo pedido de carga e execução, resultante do comando "TCHAU", teclado pelo usuário. Nesse momento a rotina de abertura é carregada sobre a rotina de tratamento, sendo enviadas para o usuário informações sobre a sessão, contidas num arquivo de dados do SIRIUS. A partir deste momento o sistema retorna ao estado inicial, onde só fica disponível ao usuário através do comando "OI". No momento que inicia a execução da rotina é verificada a existência do arquivo de trabalho, sendo este salvo , caso não tenha sido feito pelo usuário.

2.6 - IMPRESSOR DO SIRIUS

Como já descrito, o módulo "IMPRESSOR" é responsável pela impressão de mensagens resultantes da execução de tarefas. Essas informações são impressas através de pedidos de impressão de mensagens depositados nesse módulo. Os pedidos são atendidos na ordem de chegada e entregues ao impressor através de uma "BCES" (Bloco de Controle de E/S), onde o endereço dos dados e o tamanho da mensagem são acessados para tratamento. Quando um pedido é entregue ao impressor, o endereço é acessado e as linhas decompostas para impressão, as quais são identificadas pelo caracter '?'.

Tomemos como exemplo um pedido resultante da deleção de um arquivo onde o endereço passado contem a seguinte informação: 'ARQUIVO NÃO EXISTE?TENTE NOVAMENTE'. A mensagem seria impressa na próxima linha disponível da janela corrente, no seguinte formato:

```
|          -
|          -
|   ARQUIVO NÃO EXISTE
|   TENTE NOVAMENTE
|          -
|          -
```

FIG. 7

A impressão de qualquer pedido segue o esquema de semáforos descrito no ítem "COMPARTILHAMENTO DE VÍDEO".

2.7 - FUNÇÕES DOS COMANDOS

Para cada comando uma função é executada pela rotina de tratamento do módulo MESTRE. Pedidos são entregues ao monitor, os quais podem ser funções internas ao mesmo ou a carga de um módulo em código objeto, responsável pela execução. Os módulos executáveis são guardados em disco com a identificação 'nome.EXE' onde 'nome' é o nome do comando teclado pelo usuário. Quando o módulo carregador é requisitado, o módulo mencionado no comando ou na tabela é carregado e automaticamente executado.

A seguir são descritas as funções dos comandos juntamente com a definição do módulo executável, caso solicitado. A descrição do módulo executável é sempre feita dentro da descrição do comando gerador que o solicita.

Todas as tarefas enviarão informações para o usuário através do módulo impressor do SIRIUS.

2.7.1 - O comando AJUDE (sinônimo) resulta numa interpretação do comando LISTE. Caso um nome de comando seja mencionado, o mesmo é adcionado à extensão '.sis' e passado como parâmetro para ser listado. A definição do módulo executável responsável pela listagem é descrita juntamente com o comando gerador resultante.

2.7.2 - O comando ASSOCIE (imediato) resulta num pedido de execução da subrotina do núcleo 'associe', onde é feita uma associação do arquivo, com o nome lógico associado a um periférico. A associação é feita referente aos arquivos que constam na tarefa a ser solicitada pelo usuário. A tabela de nomes lógicos é atualizada, caso necessário.

2.7.3 - COMANDO DATA

O comando DATA (gerador) resulta num pedido de carga e execução do módulo 'data.exe', o qual pedirá a execução da subrotina do núcleo responsável pelo relógio do sistema e enviará ao usuário a data e a hora.

2.7.4 - COMANDO OI

O comando 'OI' (gerador) é especialmente tratado pela rotina de inicialização do módulo MESTRE. Quando o comando é reconhecido, a rotina deposita um pedido de carga e execução do restante do módulo executável responsável pela listagem de informações por terminal. O nome de um arquivo de dados do sistema é passado como parâmetro para ser listado e a senha do usuário é guardada numa área para posterior tratamento a nomes de arquivo, pelo diretor de disco.

2.7.5 - COMANDO TCHAU

Quando o comando TCHAU (gerador) é reconhecido, a rotina de tratamento deposita um pedido de carga e execução da rotina de fechamento do módulo MESTRE, TCHAU.EXE, responsável pela emissão de informações referentes à utilização feita pelo usuário. Juntamente com este pedido, outros dois pedidos são depositados para que a rotina de abertura seja recarregada e o arquivo de trabalho selecionado seja salvo, caso exista.

2.7.6 - COMANDO SUSPENDE

O comando SUSPENDE (imediato), é um comando que interrompe e termina a execução da tarefa referenciada pelo no-

me, no comando de entrada. Isto implica numa chamada à subroti na 'termine' do núcleo, que recebe o nome da tarefa menciona - da no comando e a retira da fila de execução suprimindo os re - sultados.

2.7.7 - COMANDO SISTEMA

O comando SISTEMA (gerador), resulta num pedido de carga e execução do módulo 'SISTEM.EXE' , responsável pela informação ao usuário do atual estado do sistema. Se uma tarefa é mencionada, uma busca é feita nas filas e o estado daquela tarefa é informado. Caso contrário, todas as filas do sistema são acessadas e informado o estado de cada tarefa nelas constantes.

2.7.8 - COMANDO EXECUTE

Se um nome de programa é mencionado, é suposta a existência de um módulo executável e um pedido de carga e execução daquele módulo é depositado no módulo carregador. Caso o nome seja omitido, um pedido é depositado sendo compilado o arquivo de trabalho previamente selecionado, caso necessário, e automaticamente executado. O passo compilação é feito, sendo carregado e executado o compilador identificado pela extensão do arquivo fonte. A decisão para compilação e/ou execução é feita pela rotina do módulo carregador.

2.7.9 - COMANDO SELECIONE

O comando 'selecione' (sinônimo), resulta numa nova interpretação do comando 'copie', sendo feita uma cópia do

arquivo FONTE mencionado para um arquivo de trabalho,'TRAB.TXT'. Com isto, é feito um pedido de carga e execução do módulo executável 'COPIE.EXE', responsável pela cópia de arquivos (veja comando COPIE). Quando um arquivo é selecionado, o mesmo fica disponível para o usuário utilizá-lo, através dos comandos que têm o arquivo de trabalho como parâmetro obrigatório ou opcional. Para isso, o nome do arquivo é guardado numa área auxiliar, para ser acessado como parâmetro pelos módulos que utilizam o arquivo de trabalho.

Além do nome do arquivo, o formato é passado como parâmetro para o módulo COPIE.EXE, para que seja feita a cópia, mesmo que o arquivo FONTE não tenha o mesmo formato do arquivo de trabalho, 80 bytes com tamanho fixo. O formato e o nome do arquivo FONTE ficam guardados numa área auxiliar para que o módulo 'SALVE.EXE' execute os procedimentos necessários para o salvamento do arquivo de trabalho. Caso o arquivo a ser selecionado não exista no diretório do sistema, o usuário receberá as mensagens de erro.

2.7.10 - COMANDO SINTAXE

O comando SINTAXE (gerador), resulta num pedido de carga e execução de um compilador, identificado pela extensão do arquivo FONTE previamente selecionado. O compilador é carregado e sua execução é feita sempre com o arquivo de trabalho, acarretando o envio de mensagens de erro, caso o arquivo FONTE não tenha sido selecionado.

2.7.11 - COMANDO DEPURE

O comando DEPURE (gerador), resulta num pedido de carga e execução do módulo 'DEPURE.EXE',responsável pela de

puração de programas previamente selecionados. O nome do arquivo de trabalho é acessado de forma semelhante ao comando SINTA - XE, onde o compilador é identificado.

2.7.12 - COMANDO APAGUE

O comando APAGUE (gerador), resulta num depósito de um pedido no módulo diretor de disco, com a opção 'delete', adcionado o nome do arquivo desejado. Caso o nome seja omitido, o módulo diretor acessa o nome do arquivo de trabalho selecionado, interrogando o usuário para a execução da tarefa.

2.7.13 - COMANDO EDITE

O comando EDITE (gerador), resulta num pedidode carga e execução do editor de textos, módulo 'EDITE.EXE', onde o arquivo de trabalho é acessado para a utilização. O comando EDITE, exige que um arquivo seja previamente selecionado, tomando este como parâmetro único e obrigatório.

2.7.14 - COMANDO DIRETÓRIO

O comando DIRETÓRIO (gerador), resulta num depósito de um pedido de carga e execução do módulo 'DIRETO.EXE',responsável por uma busca do(s) arquivo(s) irmão(s) mencionado(s) e informação da sua existência ou não através do impressor do SI - RIUS.

2.7.15 - COMANDO COPIE

O comando COPIE (gerador), é responsável pela cópia de arquivos e é usado quando resulta de um comando sinôni-

mo ou quando é solicitado diretamente pelo usuário. O comando 'COPIE' sempre reproduz um arquivo 'A' para um arquivo 'B', po - dendo estes arquivos serem palavras que definam periféricos ou arquivos previamente associados pelo usuário. Caso os arquivos não estejam associados, é assumido o disco como sendo o periféri co associado aos arquivos. Isto é feito durante a preparação dos arquivos para a cópia, ou seja, quando o comando é reconhecido e o módulo 'COPIE.EXE' é executado. As funções básicas do 'COPIE' resumem-se em pedir a abertura dos arquivos e em seguida execu - tar a cópia solicitada, o que resulta no começo de uma troca de informações entre os processos envolvidos. A princípio, os ar - quivos são verificados para definir-se em qual periférico se en- contram, definindo-se assim, em qual módulo serão feitos os pedi dos referentes àqueles arquivos. Uma busca é feita na tabela de nomes físicos, os quais podem ser de entrada: tecla, disco, K7 ; ou de saída: tela, disco, K7, impressora. Se um nome físico não é encontrado, uma tabela de nomes lógicos é acessada para verifi car se os arquivos foram associados pelo usuário. Se isto ocor- rer, o número do módulo é guardado e a tarefa iniciada; caso con trário, é suposta a existência dos arquivos no diretório, sendo depositados no diretor (disco) os pedidos de abertura e fechamen to de arquivos, assim como as operações de entrada e saída a exe cutar. Caso os arquivos não existam, o diretor será responsável pelo envio de um 'status', o que resultará em mensagem de erro e suspensão da execução da cópia.

Qualquer pedido que resulte na execução do módu lo 'COPIE.EXE' deve conter todas as especificações do arquivo a ser copiado.

2.7.16 - COMANDO MUDE

O comando MUDE (gerador), resulta num depósito de pedido no diretor do disco onde é passada a opção 'MUDE', adicionados os nomes constantes no comando. Com isso, o diretor pega o pedido e executa a troca do nome do arquivo solicitado.

2.7.17 - COMANDO SALVE

O comando SALVE (gerador), resulta num pedido de carga e execução do módulo 'SALVE.EXE'. Esse módulo é responsável pela inclusão do arquivo de trabalho no diretório, segundo as condições a seguir. Se o arquivo a salvar tem o mesmo nome que o arquivo fonte, isto é, um nome não é mencionado no comando, é depositado um pedido de deleção do arquivo fonte e em seguida um pedido de troca do nome do arquivo de trabalho pelo do fonte. Se o arquivo a salvar tem um novo nome, apenas é feito um pedido de troca do nome do arquivo de trabalho pelo nome mencionado no comando 'SALVE'. Em ambos os casos, o módulo objeto do arquivo de trabalho é trocado pelo nome 'arq.EXE', onde 'arq' é o nome do arquivo fonte sem a extensão.

Para que sejam feitos os procedimentos acima mencionados, o arquivo fonte deverá ter o mesmo formato do arquivo de trabalho, ou seja, 80 bytes com tamanho fixo.Este formato é identificado pelo módulo 'Salve.EXE' através do acesso a uma área auxiliar comum, onde é depositado o formato do arquivo fonte pelos procedimentos resultantes do comando 'SELECIONE'.Caso o formato não seja o mesmo, é feito um pedido de carga e execução do módulo 'COPIE.EXE', sendo passado como parâmetro o

formato e o nome do arquivo FONTE, seguido de um pedido de deleção do arquivo FONTE original. Caso um nome seja mencionado no comando, o novo arquivo salvo receberá este nome e o arquivo fonte original não é deletado.

2.7.18 - COMANDO LISTE

O comando LISTE é um comando sinônimo que resulta numa nova interpretação do comando COPIE. O comando COPIE resultante tem o formato 'COPIE ARQUIVO TELA:', onde ARQUIVO é o nome do arquivo mencionado no comando LISTE. Com isso, o comando COPIE pede a abertura do arquivo mencionado, através de um pedido depositado no diretor do disco e uma abertura do periférico 'TELA' ao módulo impressor do sistema, dando início a execução da listagem.

2.7.19 - COMANDO CRIE

O comando CRIE é um comando sinônimo que resulta numa nova interpretação do comando COPIE. O formato final da interpretação é 'COPIE TECLA: ARQUIVO', onde ARQUIVO é o nome do novo arquivo em disco ou em qualquer outro periférico anteriormente associado pelo usuário. Com isso, um pedido de abertura é solicitado ao módulo responsável pelo periférico em questão, e o gerente de teclado é responsável pelo recebimento e entrega da informação ao processo COPIE.

2.7.20 - COMANDO SUBMETA

O comando SUBMETA (gerador), resulta num pedido de carga e execução do módulo 'SUBMET.EXE', responsável pela

leitura dos comandos contidos num arquivo LICO, assim como pela entrega dos mesmos ao módulo MESTRE. Para cada comando encon - trado, um pedido com reativação é feito e, logo após o término da tarefa referente àquele comando, o módulo é novamente ativado prosseguindo com os pedidos para cada comando encontrado. Caso um novo comando 'SUB' seja encontrado, um novo módulo 'SUBMET. EXE' é carregado e o fluxo de depósitos de pedidos prossegue da mesma forma do módulo anterior.

2.7.21 - COMANDOS <PL> E <CL>

Os comandos <PL> e <CL> (geradores), são especialmente tratados pelo Manipulador e resultam numa interferência do Manipulador no módulo impressor do sistema.

Quando o <PL> é reconhecido, uma chamada é feita à subrotina do núcleo 'ESPERA-FÍSICA', fazendo com que a execução do impressor seja interrompida, parando o envio de informações para a tela. Ao contrário do <PL>, o <CL> resulta numa chamada à subrotina do núcleo 'ATIVE', que faz a ativação do módulo impressor, anteriormente desativado por um <PL>, dando continuidade ao envio de informações para a tela.

CAPÍTULO III

3. DEFINIÇÃO DOS MÓDULOS

3.1 - MANIPULADOR

O MANIPULADOR, rotina residente, é ativado através de interrupção quando o usuário pressiona uma tecla. As funções básicas do Manipulador resumem-se em informar ao usuário, a través do periférico 'TELA', as informações que serão posteriormente tratadas pelo sistema, e servir de veículo entre o usuário e os processos responsáveis pela execução de procedimentos, re - sultantes de qualquer informação enviada pelo usuário.

3.1.1 - INFORMAÇÕES DE ENTRADA

A única informação tida como entrada do Manipulador é o caracter que, independente do seu tipo, será depositado num endereço e recebido pelo Manipulador para ser tratado.

Um caracter pode ser de 4 tipos:

- Caracter de posicionamento
- Caracter de tela (LT,IL,NL)
- Caracter funcional (CMD,FUN,INS,DEL,RES,ENTRE)
- Caracter simples (A-Z,especial,0-9,pontuação)

Para cada caracter é dado um tratamento de acordo com o seu tipo e atual estado do Manipulador. No caso de recebimento dos comandos de controle, <PL> e <CL>, o próprio Manipulador define o processo a ser invocado.

3.1.2 - DEFINIÇÃO DOS PROCEDIMENTOS

Para cada tipo de caracter existe um procedimento a ser executado e, de acordo com a informação recebida, o Manipulador está pronto para entregá-la a um outro processo ou retornar ao estado de espera de um outro caracter.

Sempre que um caracter é recebido, uma decisão é tomada em relação à posição atual do cursor. Isto é, sendo (x,y,z) os fatores que representam a posição do cursor, os mesmos podem ser alterados resultando num deslocamento do cursor.

3.1.3 - CARACTERES DE POSICIONAMENTO

Quando um caracter de posicionamento é recebido, apenas os fatores (x,y,z) são alterados, atualizando a posição do cursor, mas sem resultar na impressão de um caracter.

Para que sejam definidos os procedimentos para cada caracter de posicionamento,é necessária a determinação dos movimentos do cursor dentro da tela, o que é mostrado na figura abaixo.

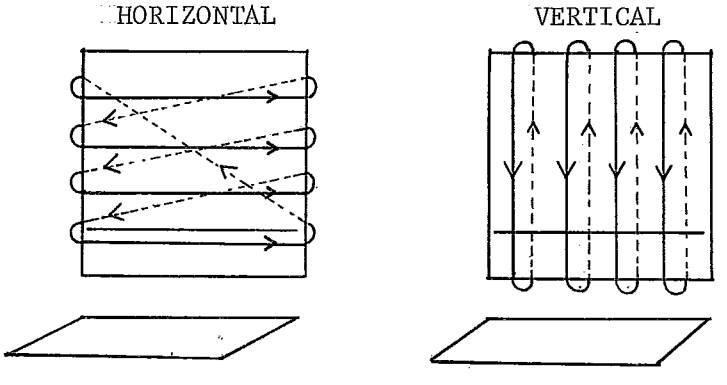


FIG. 8

Baseando-se nos movimentos mostrados pela figu - ra, são definidos os movimentos para os seguintes caracteres:

→ - A linha do cursor não é alterada e a coluna in - crementada de um, causando uma remoção para a direita. Caso a posição ultrapasse a coluna 80 , o cursor é posicionado na primeira posição da próxima linha. Nota-se que a linha não pode ser a 27 e, caso a atual seja a 30, a nova linha a ser considerada é a primeira da janela superior.

← - Semelhante ao caso anterior, somente a coluna é alterada com o decremento, causando uma remoção do cursor para a esquerda. Caso ultrapasse a coluna um, o cursor é posicionado na coluna 80 da linha anterior. Se a linha a ser tratada for a linha 1, a nova linha a ser considerada é a 30 , obedecendo ao esquema da figura 12.

↓ - Neste caso, a coluna permanece inalterada e a linha é incrementada, sendo considerada a linha um da mesma coluna, caso a linha em questão seja a 30.

↑ - Ao contrário do caso anterior, a linha é decre - mentada, permanecendo na mesma coluna. Se com o decremento a linha um for ultrapassada, a 30 é a nova linha a ser tratada.

<TAB> Quando este caracter é recebido, o cursor é posicionado na próxima posição tabulada. Para que isto ocorra, o Manipulador acessa um conjunto de

tabulações contido num vetor, e pega a próxima posição tabulada, atualizando a posição do cursor. O vetor de tabulações contem a linha e coluna de todas as posições a tabular, da matriz de projeção. Essas posições são definidas por processos que utilizam o teclado como 'ENTRADA' e que passam o vetor de tabulações como parâmetro, durante a abertura dos periféricos TELA/TECLADO.

<NL> Causa o remanejamento do cursor para a primeira posição (coluna) da linha seguinte. Caso a linha atual seja a 30, o cursor é posicionado na primeira linha da janela superior.

Caso um desses caracteres de posicionamento ultrapassem os limites vertical ou horizontal da janela superior, um 'STATUS' é preparado e enviado para o processo que tenha depositado um pedido de entrada, à nível de caracter, no gerente do teclado. O campo 'Z' é alterado à medida que o cursor muda de janela.

3.1.4 - CARACTERES DE TELA

Esses caracteres causam a alteração de informações na tela seguida de uma alteração da posição do cursor. Em seguida são mostrados os caracteres com seus respectivos procedimentos.

<LT> Esse caracter causa a limpeza de toda a janela em que o cursor se encontra. Caso o cursor esteja na janela superior, a mesma é totalmente -

limpa e o cursor fica disponível na linha de entrada da janela inferior. Caso o cursor esteja na janela inferior, esta é limpa e o cursor fica na linha 28, coluna um.

<LL> Esse caracter causa a limpeza da linha corrente, a partir da coluna em que se encontre o cursor.

3.1.5 - CARACTERES FUNCIONAIS

Os caracteres funcionais estabelece um novo estado a partir do qual as informações recebidas são tratadas de acordo com a função definida pela natureza do caracter. São eles:

<CMD> Este caracter tem como função distinguir o módulo que receberá do Manipulador a informação teclada.

<FUN> Este caracter tem como função distinguir a natureza da informação recebida, podendo defini- la como sendo um comando para um outro módulo. Com isso, processos que têm o teclado como entrada (ex: EDITOR), podem receber informações do tipo 'comando', sendo estas diferenciadas das demais quando prescedidas pelo caracter <FUN>.

<INS> Quando este caracter é recebido, um novo estado é estabelecido, ficando o teclado em modo de 'inserção', para receber as próximas informa - ções. Com isso, qualquer caracter simples (definido adiante), ao ser recebido é inserido na

posição onde se encontra o cursor, sendo os demais remanejados para a direita, ficando o cursor posicionado logo após o caracter inserido.Nota-se que mesmo em modo de inserção, o tamanho máximo da mensagem é de 80 caracteres, ficando o teclado bloqueado caso seja feita uma tentativa.

<RES> Este caracter 'restaura' o estado do teclado que foi alterado através de um erro por parte do usuário, ou através do recebimento do caracter <INS>. Com isso,em qualquer novo estado em que o teclado se encontre, o usuário poderia desativá-lo, pressionando a tecla <RES>. O mesmo ocorre para o estado de inserção, onde o Manipulador passa a ecoar normalmente os caracteres recebidos.

<ENTRE> O caracter <ENTRE>, quando recebido, determina que as informações dadas como entrada, devem ser passadas para o módulo em questão. Este caracter é análogo a um 'final de registro', definindo um estado para os processos que têm sua entrada a nível de caracter.

3.1.6 - CARACTERES SIMPLES

Esses caracteres são informações que não causam efeitos funcionais sobre o Manipulador, e que constituem mensagens a serem tratadas pelo sistema. Os caracteres são os 'espe

ciais': ! , / , ? , $, < , > , = , & , | , % , - , + , * , (,) ; os de pontuação, o alfabeto, e os números de 0 a 9.

3.1.7 - ALGORITMO

Esse algoritmo contém procedimentos referentes a todos os caracteres teclados pelo usuário, assim como as decisões supostas para a formatação de mensagens. Durante sua execução, endereços de desvios são alterados para que instruções sejam inoperadas, evitando o teste de informações desnecessárias. As seguintes condições são consideradas:

1 - A entrada de um comando, quando iniciada por um <CMD>, implica na entrega do mesmo ao monitor do MESTRE independendo da existência de pedidos feitos ao GERENTE. Qualquer caracter simples dá início à montagem de um comando.

2 - A entrada de um comando quando não iniciada por um <CMD>, implica na sua entrega ao monitor do MESTRE, a menos que exista um pedido de leitura depositado no monitor do GERENTE de teclado. Nesse caso a informação é entregue ao GERENTE sendo o mesmo ativado pelo MANIPULADOR.

3 - Os caracteres de posicionamento do cursor são tratados de forma que sejam testados os semáforos referentes a janela em questão. O teste do semáforo só é feito antes do início da montagem de um comando.

Para que não sejam feitos esses testes ou execu-

ções desnecessárias as variáveis abaixo representam endereços de desvios, os quais definem estados do Manipulador.

E1 - Esta variável compõe o primeiro passo do algoritmo o qual recebe endereços de desvios, dependendo das seguintes condições:

a. Ocorreu um erro por parte do usuário, onde 'E1' recebe o passo (endereço) onde se encontra a rotina que só aceita o caracter <RES>.

b. A mensagem já começou e o 'E1' recebe o passo logo após o teste dos semáforos.

E2 - Esta variável antecede o passo onde é feita a ativação dos semáforos, recebendo o passo imediatamente após a ativação para que a mesma não seja feita desnecessariamente.

E3 - Esta variável antecede o passo onde é feita a 'inserção' e inicialmente recebe o passo onde cada caracter é ecoado, sem o Manipulador estar em modo de inserção. Quando o usuário tecla o caracter <INS>, 'E3' recebe o passo onde a inserção é tratada, sendo restaurado quando o Manipulador recebe um '<RES>' ou um '<ENTRE>'.

Além dos endereços acima descritos, as seguintes variáveis com suas respectivas funções controlam o funcionamento do algoritmo:

X - Apontador da coluna corrente

Y - Apontador da linha corrente

```
Z   -  Indicador da janela corrente.
TM  -  Indicador da mensagem a nível de caracter
EXISTE_PEDIDO - BOOLEANA
ULD -  Última linha de entrada disponível,  antes
       do cursor ser direcionado para outra.

/*  INICIALIZAÇÃO DAS VARIÁVEIS  */

X = 1 ; Y = 28 ;
Z = 3 ; TM = 80 ;
ULD = 28 SALVATM = 0 ;
E1 = 2 ; E2 = 14 ; E3 = 23 ;

   /*  INÍCIO DO ALGORITMO */

1.  VA P/ E1 ;
2.  FAÇA
    E1 = 3 ;
    SE Z = 3
    ENTÃO SE S3 = 1 /* TESTA SUPERIOR */
             ENTÃO PARE :
             SENÃO VÁ P/ 3 ;
    SENÃO SE S2 = 1 /* TESTA INFERIOR */
             ENTÃO PARE ;
             SENÃO VÁ P/ 3 ;
    FIMFA

      /* TESTA CARACTER */

3.  SE CARACTER = 'DEL' ENTÃO VÁ P/ 68 ;
3.1 SE CARACTER = 'PL'  ENTÃO VÁ P/ 74 ;
3.2 SE CARACTER = 'CL'  ENTÃO VÁ P/ 76 ;
```

```
 4.  SE CARACTER = 'LT'  ENTÃO VA P/ 34 ;
 5.  SE CARACTER = 'LL'  ENTÃO VA P/ 42 ;
 6.  SE CARACTER = ' → ' ENTÃO VA P/ 44 ;
 7.  SE CARACTER = ' ← ' ENTÃO VA P/ 51 ;
 8.  SE CARACTER = ' ↑ ' ENTÃO VA P/ 55 ;
 9.  SE CARACTER = ' ↓ ' ENTÃO VA P/ 48 ;
10.  SE CARACTER = 'INS' ENTÃO VA P/ 58 ;
11.  SE CARACTER = 'RES' ENTÃO VA P/ 63 ;
12.  SE CARACTER = 'NL'  ENTÃO VA P/ 66 ;
13.  VA P/ E2 ; /*  INIBIDOR DA ATIVAÇÃO DO SEMÁFORO */
14.  INIBE-INTERRUPÇÃO ; ORCC ÕINIBI)
     SE Z = 3
        ENTÃO S3 = 1 ;
        SENÃO S2 = 1 ;
     LIBERA-INTERRUPÇÃO ; (ANDCC ÕLIBEI)
15.  SE TM = 1 ENTÃO E2 = 16 ;
                    SENÃO FAÇA
                             E2 = 18 ;
                             VA P/ 18 ;
                          FIMFA
16.  BUFFER.GERENTE = CARACTER ;
17.   SE ESPERA.MANIPU = TRUE /* TESTA FLAG DE ATIV. */
        ENTÃO PEDE.ATIVAÇÃO.GERENTE ;
18.  SE CARACTER = <ENTRE>
        ENTÃO
          FAÇA
            I = 80 ;
            ENQUANTO MATRIZ(Y,I) = ' ' & I  > 1
              FAÇA I = I - 1 ;
            MAX = I ;
```

```
                ENQUANTO MATRIZ(Y,I)   = ':' & I >  1
                   FAÇA I = I - 1 ;
                SE EXISTE-PEDIDO /* TM = 2 */
                   ENTÃO
                     SE MATRIZ(Y,1)    =  <CMD>
                        ENTÃO
                          FAÇA
                            BUFFER.GERENTE=MATRIZ(Y,I A MAX);
                            PEDE.ATIVAÇÃO.GERENTE ;
                            VA P/ 26 ;
                          FIMFA
                BUFFER.MONITOR.MESTRE = MATRIZ(Y,I A MAX) ;
                VA P/ 26
               FIMFA
19.  VA P/ E3 ; /*  INIBIDOR DO TRATAMENTO A INSERÇÃO */
20.  I = 80 ;
21.  ENQUANTO MATRIZ(Y,I) = ' ' FAÇA I = I - 1 ;
22.  SE I = 80
        ENTÃO FAÇA
                MATRIZ(27,10) = 'LINHA-CHEFIA' ;
                CHAVE = 2 ; /* ATIVA CONDIÇÃO DE ERRO */
                E1 = 62 ; /* SO ACEITA 'RES' */
                PARE ; /* ESPERA INTERRUPÇÃO */
              FIMFA
      SENÃO FAÇA /* REMANEJA 1 POS. P/ DIREITA */
                ENQUANTO I > X
                        FAÇA
                          I = I - 1 ;
                          MATRIZ(Y,I+1) = MATRIZ(Y,I) ;
                        FIMFA
```

```
23.  MATRIZ(Y,X) = CARACTER ; /* ECOA */
25.  VA P/44 ; /* VERIFICA LIMITE DA LINHA */

          /* ATUALIZA ENDEREÇOS E LIBERA CURSOR */

26.  E1 = 2 ;
27.  E2 = 14 ;
28.  X = 1 ;
29.  Y = Y + 1 ;
30.  SE Z = 3
        ENTÃO FAÇA
                SE Y > 30
                   ENTÃO VA P/ 35 ; /* LIMPA JANELA INF. */
                ULD = Y ; /* PEGA ÚLTIMA LIN. DISP. */
              FIMFA
        SENÃO S2 = 0
31.  POSICIONA-CURSOR ;
32.  SE CHAVE   =  0
        ENTÃO FAÇA /* RESTAURA DIVISÓRIA */
                MATRIZ(27) = '----------------------' ;
                CHAVE = 0
              FIMFA
33.  PARE ; /* ESPERA INTERRUPÇÃO */

          /* TRATAMENTO DOS CARACTERES DE CONTROLE */

          /* LIMPA TELA (LT) */

34.  SE Z = 2
        ENTÃO FAÇA /* LIMPA SUPERIOR */
                FAÇA I DE 1 A 27 ;
```

```
                    FAÇA J DE 1 A 80 ;
                        MATRIZ(I,J) = ' ' ;
                    FIMFA
                    X = 1 ;
                    S2 = 0 ;
                    Y = 1 ;
                    POSICIONA-CURSOR ;
                    PARE ; /* ESPERA INTERRUPÇÃO */
                    FIMFA
35.  FAÇA I DE 28 A 30 ;
       FAÇA J DE 1 A 80 ;
           MATRIZ(I,J) = ' ' ;
       FIMFA
     FIMFA
36.  Y = 28 ;
37.  ULD = 28 ;
38.  X = 1 ;
39.  S3 = 0 ; /* DESATIVA S3 */
40.  POSICIONA-CURSOR ;
41.  ESPERA-INTERRUPÇÃO ;

          /* LIMPA LINHA (LL) */

42.  FAÇA J DE X A 80 ;
          MATRIZ(Y,J) = ' ' ;
     FIMFA
43.  PARE ; /* ESPERA INTERRUPÇÃO */

          /* TRATA CARACTER ' → ' */
44.  X = X + 1 ;
45.  SE X <= 80 ENTÃO VA P/ 71 ; /* POSICIONA */
```

```
46.  X = 1 ;
47.  PREPARA-STATUS ; /* PREPARA ÁREA QUE VAI P/ MONITOR */

          /* TRATA CARACTER '↓' */

48.  Y = Y + 1 ;
49.  SE Y = 27
        ENTÃO FAÇA
                PREPARA-STATUS ;
                SE S3 = 1
                   ENTÃO Y = 26 ;
                   SENÃO FAÇA
                           Y = 28 ;
                           Z =  3 ;
                         FIMFA

              FIMFA
        SENÃO SE Y > 30
                 ENTÃO SE S2 = 1
                          ENTÃO Y = 30 ;
                          SENÃO Y = 1 ;
50.  VA P/ 71 ; /* ENTREGA STATUS A NÍVEL DE CARACTER */

          /* TRATA CARACTER ' ← ' */

51.  X = - 1 ;
52.  SE X > 0 ENTÃO VA P/ 71 ;
53.  X = 80 ;
54.  PREPARA-STATUS ;

          /* TRATA CARACTER ' ↑ ' */

55.  Y = Y - 1 ;
```

```
56.  SE Y = 27
        ENTÃO FAÇA
                PREPARA-STATUS ;
                SE S2 = 1
                   ENTÃO Y = 28 ;
                   SENÃO FAÇA
                           Y = 26 ;
                           Z = 2  ;
                         FIMFA
              FIMFA
        SENÃO SE Y < 1
        ENTÃO FAÇA
                PREPARA-STATUS ;
                SE S3 = 1
                   ENTÃO Y = 1 ;
                   SENÃO FAÇA
                           Y = 30 ;
                           Z =  3 ;
                         FIMFA
              FIMFA
57.  VA P/ 71 ; /* ENTREGA STATUS A NÍVEL DE CARAC. */

          /* TRATA CARACTER 'INS' */

58.  CHAVE = 1 ; /* ATIVA COND. INSERÇÃO */
59.  E3 = 20 ; /* ATUAL.  ENDER.  DA INSERÇÃO */
60.  MATRIZ(27,60) = 'INSERE' ;
61.  PARE ; /* ESPERA INTERRUPÇÃO */

          /* ROTINA QUE SÓ ACEITA 'RES' */
```

```
62. SE CARACTER    =  'RES'
       ENTÃO PARE ;
63. SE CHAVE = 1  /* TRATA O 'RES' */
       ENTÃO FAÇA
             MATRIZ(27,60) = '--------------------' ;
             E3 = 23 ;
          FIMFA
       SENÃO SE CHAVE = 2
             ENTÃO FAÇA
             MATRIZ(27,10) = '--------------------' ;
             E3 = 23 ;
          FIMFA
64. CHAVE = 0 ; E1 = 3 ;
65. PARE ;

       /* TRATA CARACTER 'NOVA LINHA ' */

66. X = 1 ;
67. VÁ P/ 48 ;  /* TRATA CARACTER ↓ */

          /* TRATA CARACTER 'DEL' */

68. FAÇA I DE X A 79 ;
       MATRIZ(Y,I) = MATRIZ(Y,I+1) ;
    FIMFA
69. MATRIZ(Y,80) = ' ' ;
70. PARE ;  /* ESPERA INTERRUPÇÃO */

    /* ENTREGA STATUS A NÍVEL DE CARACTER */

71. SE TM = 1
       ENTÃO BUFFER-GERENTE = CARACTER + STATUS ;
72. POSICIONA-CURSOR
```

```
73. PARE ;  /* ESPERA INTERRUPÇÃO */

/* TRATA <PL> */
74. PEDE.PARADA.IMPRESSOR.SISTEMA ;
75. PARE ;  /* ESPERA INTERRUPÇÃO */

/* TRATA <CL> */
76. PEDE.CONTINUAÇÃO.IMPRESSOR.SISTEMA ;
77. PARE ;  /* ESPERA INTERRUPÇÃO */
```

3.2 - MÓDULO GERENTE

O módulo GERENTE, como já descrito, é responsável pelo gerenciamento do teclado. É através desse módulo que é feita a entrega de informações de entrada de um usuário, tendo o mesmo requisitado o 'teclado' como um arquivo de entrada.

3.2.1 - DEFINIÇÃO DAS INFORMAÇÕES

Quando o GERENTE é ativado, o mesmo acessa sua fila de requisições onde identifica uma 'BCES'. Contidas na 'BCES' estão todas as informações necessárias para execução de procedimentos do GERENTE, podendo essa ser uma abertura, leitura ou fechamento de arquivo. Desta forma um pedido sempre é identificado por uma 'BCES' que, de acordo com o preenchimento de seus campos, define a operação a ser feita. Além da 'BCES' como sendo uma informação a ser tratada, existe uma área de dados onde serão depositadas, pelo Manipulador, as informações tecladas. Com isso a 'BCES' e a área de dados definem as ferramentas utilizadas para que o ciclo entrega/recebimento seja concluído.

3.2.2 - DEFINIÇÃO DOS PROCEDIMENTOS

Quando o GERENTE é ativado o mesmo testa a existência de um pedido na sua fila de requisições. Caso o pedido seja de abertura, a identificação do usuário é transferida para uma área auxiliar, sendo a mesma testada sempre que uma leitura é feita. Com isso somente o usuário que abriu o arquivo poderá fazer pedidos de leitura seguidos da entrega de dados. Quando um pedido é de leitura, é testada a validade do usuário seguida do teste da existência de informação na área de dados. Caso haja informação, a mesma é transferida para a área de leitura definida pela 'BCES', e em seguida, o pedido finalizado até a próxima ocorrência. Caso não haja informação, o gerente fica esperando uma nova ativação feita pelo Manipulador ou pelo processo requisitante. Por fim quando um pedido é de fechamento, a área auxiliar é restaurada, liberando assim o arquivo para qualquer outro usuário. No caso de ocorrência de pedidos de abertura ou fechamento, o Manipulador é avisado para efeito de preenchimento ou não da área de dados.

3.2.3 - ALGORITMO

As seguintes variáveis são utilizadas:

ÁREA - Área de memória.

BCES - BCES do pedido depositado:

EXISTE-INFO - Booleana (tem ou não info no buffer)

TM - Identifica o tipo de leitura
  0 - Não existe
  1 - A nível de caracter
  2 - A nível de registro

USU - Ident. do usuário que abriu o arquivo

```
BUFFER - Área de depósito do Manipulador
PEDIDO - Cod. da BCES (tipo de operação)

/* GERENTE */

1. CALL RETIRA ;
2. SE USU = IDENT.USU.BCES
      ENTÃO VA P/ 4 ;
3. SE USU = 0
      ENTÃO SE PEDIDO = ABERTURA
            ENTÃO FAÇA /* ABRE ARQUIVO */
                  TM = TIPO.LEITURA.BCES ;
                  USU = IDENT.USU.BCES ;
                  FINALIZA ;
                  VA P/ 1 ;
               FIMFA
            SENÃO FAÇA /*  TEM QUE ABRIR ARQUIVO  */
                  PREPARA_STATUS ;
                  FINALIZA ;
                  VA P/ 1 ;
               FIMFA
      SENÃO FAÇA  /* USUÁRIO NÃO AUTORIZADO  */
            PREPARA_STATUS ;
            FINALIZA ;
            VA P/ 1;
         FIMFA

/*  ARQUIVO ABERTO  */

4. SE PEDIDO = ABERTURA
      ENTÃO FAÇA /* ARQUIVO JÁ ESTÁ ABERTO */
```

```
                PREPARA_STATUS ;
                FINALIZA ;
                VA P/ 1
            FIMFA
5. SE PEDIDO = FECHAMENTO
      ENTÃO FAÇA  /* FECHA ARQUIVO */
            TM = 0 ;
            USU = 0 ;
            FINALIZA ;
            VA P/ 1 ;
          FIMFA

/*  PEDIDO DE LEITURA  */

6. SE NÃO 'TEM_INFO'
      ENTÃO FAÇA /* ESPERA MANIP. */
            INIBE_INTERRUPÇÃO ;
            ESPERA.MANIP = TRUE ;
            LIBERA_INTERRUPÇÃO ;
            CALL SUSPENDE ;
          FIMFA
7. INIBE_INTERRUPÇÃO ;
      ESPERA.MANIP = FALSE ;
   LIBERA_INTERRUPÇÃO ;
8. INIBE_INTERRUPÇÃO  ;
      ÁREA(END.BCES) = BUFFER ;
      TEM_INFO = FALSE ;
   LIBERA_INTERRUPÇÃO ;
9. FINALIZA ;
10. VA P/ 1 ;
```

3.3 - MÓDULO MESTRE

O módulo MESTRE, como já descrito, é responsá - vel pela interpretação e ativação do serviço resultante do co - mando teclado pelo usuário. O módulo MESTRE é composto por 3 outros módulos que, segundo o fluxo hierárquico da utilizaçãode um usuário, define os estados em que o sistema pode se encon - trar, visando sua utilização para o usuário. Com isso, o siste ma fica disponível para um usuário através da rotina de 'abertu ra', que o avalia e colhe as informações inerentes à sua utili- zação ; o sistema fica disponível para execução de tarefas a - través da rotina de 'tratamento'; e finalmente é finalizado pe la rotina de 'fechamento', o qual fica bloqueado até uma nova utilização.

3.3.1 - ROTINA DE ABERTURA

A rotina de abertura é a primeira parte do módu lo MESTRE a ser carregada quando o sistema é inicializado. A execução da rotina de abertura consiste no recebimento de infor mações a nível de caracter, na avaliação das mesmas segundo as especificações do sistema, na entrega dessas informações aos processos que as utilizarão e, por fim, na carga do restante do módulo MESTRE.

3.3.1.1 - INFORMAÇÕES DE ENTRADA

As informações recebidas pelo sistema são a nível de caracter e compoem mensagens que definem estados da rotina. Inicialmente a rotina só aceita a mensagem 'OI', causando a impressão de mensagens de er-

ro, através do impressor do SIRIUS, caso esta mensagem não seja recebida.

Como o recebimento é a nível de caracter, a própria rotina tem o controle do tamanho da mensagem a receber, não sendo necessária a entrada de um 'fim-de-mensagem', a menos que o usuário deseje cancelar todas as informações tidas como entradas até aquele ponto, retornando ao estado inicial.

3.3.1.2 - DEFINIÇÃO DOS PROCEDIMENTOS

Como a rotina de abertura tem o controle do tamanho da mensagem, após cada mensagem ser completada, a rotina informa a próxima mensagem a ser teclada pelo usuário. O esquema funciona da seguinte forma:

1. Somente os caracteres 'O' e 'I' são aceitos, sendo emitidas mensagens de erro, caso um outro seja teclado.

2. Logo após o recebimento do caracter 'I', a rotina começa a informar qual a mensagem a receber, e fica recebendo os caracteres, até que uma mensagem seja concluída. Esses procedimentos de informar e receber se repetem até que todas as informações sejam recebidas.

3. O recebimento de um <ENTRE> em qualquer fase da rotina causará o reinício da mesma.

As informações a serem recebidas estão de acordo com a utilização do usuário e nesta versão são as seguintes: senha do sistema, sigla do usuário e o disco a ser utilizado. Para que possam ser incluídas ou excluídas informações que definem uma utilização do usuário, a rotina de abertura utiliza uma 'tabela de utilização', que consta de todas as informações a receber com suas respectivas mensagens a emitir. Com isso, qualquer inclusão ou exclusão de informações, uma alteração na tabela é necessária seguida da inclusão de procedimentos ao algoritmo. Essa inclusão de procedimentos no algoritmo não altera a estrutura do mesmo, uma vez que os tratamentos dados às mensagens estão divididos em módulos independentes.

As seguintes informações compoem a tabela de utilização: endereço do próximo passo a ser executado (EPP), mensagem a emitir (TABMENS), tamanho da mensagem a receber (TAMREC), e mensagem recebida (TABREC). Segue abaixo a descrição de todos os campos com suas respectivas utilizações.

Endereço do próximo passo a ser executado:

Este campo contém o número do próximo passo a ser executado após o recebimento da informação ser completado. Com isso, após seu recebimento, o algoritmo desvia para 'EPP', onde se encontram os procedimentos que definem a crítica das informações recebidas.

Mensagem a emitir.

Este campo contém a mensagem a ser emitida para o usuário, antes do mesmo entrar com as informações. Tomemos como exemplo a 'senha do usuário', onde este campo contém: 'DIGITE SENHA COM 4 POSIÇÕES'

Tamanho da mensagem a receber.

Este campo contém o número de caracteres a serem recebidos, que definem a mensagem referente a última informação dada pelo sistema ao usuário. Este campo também define o final do ciclo 'envio e recebimento', quando o mesmo contiver o valor zero, determinando o final da tabela de utilização.

Mensagem recebida.

Este é o campo onde serão depositados os caracteres recebidos referentes a última mensagem. Após o recebimento da última informação, todos esses campos são entregues aos processos que utilizam as informações tecladas.

A tabela abaixo é utilizada nessa primeira versão do algoritmo.

TABELA DE MENSAGENS

EPP	TABMENS	TAMREC	TABREC
21	DIGITE SENHA (4 POS)	4	
22	DIGITE SIGLA (3 POS)	3	
23	DIGITE DISCO	6	
26	*** SESSÃO ABERTA **	0	

Figura 9

Após cada recebimento de informação, a mesma é criticada, sendo informada ao usuário a exis - tência de algum erro, ou ficando o mesmo habilitado a entrar com o restante das informações. No final de to das as informações serem recebidas, a rotina interroga o usuário, a fim de verificar se as informações tecla- das estão realmente de acordo com sua utilização. Nes se caso, qualquer informação a ser alterada e que não esteja correta, o usuário terá que entrar com todas as informações novamente.

A crítica das informações consiste : no 'teste da senha', para verificar se a mesma é igual a permitida pelo sistema; na 'validade da sigla', atra vés de uma pesquisa no diretório do disco; e na exis - tência do volume do disco, informado pelo usuário. Ca- so qualquer um desses testes não seja válido, o usuá - rio será informado através de mensagens de erro.

As seguintes variáveis são utilizadas no algoritmo:

IND - Índice da tabela de utilização, incrementado após cada mensagem recebida.

PPC - Contém o próximo passo a ser executado, caso a mensagem recebida esteja correta.

PPE - Contém o próximo passo a ser executado, caso a mensagem recebida esteja errada. O conteúdo de 'PPC' e 'PPE' é alterado de acordo com a fase da rotina onde a mesma pode estar em fase de recebi mento ou de verificação.

PP - Contém o próximo passo a ser executado, após o recebimento de uma mensagem.

3.3.1.3 - ALGORITMO

```
   /* INICIALIZAÇÃO DAS VARIÁVEIS */

1. IND = 0 ; TABREC(4) = 0 ;
2. PPC = 8 ;
3. PPE = 11 ;

   /* ENVIO E RECEBIMENTO */

4. LÊ CARACTER ;  /* FAZ PEDIDO A NÍVEL DE CARACTER */ ;
5. SE CARACTER   ≠ 0
      ENTÃO FAÇA
            PEDE-IMPRESSÃO 'PARA UTILIZÁ-LO TECLE OI' ;
            VA P/ 4 ;
          FIMFA ;
6. SE CARACTER   ≠ I
      ENTÃO FAÇA
            PEDE-IMPRESSÃO 'PARA UTILIZÁ-LO TECLE OI' ;
            VA P/ 4 ;
          FIMFA ;
7. SE CARACTER = <ENTRE>
      ENTÃO VA P/ 24 ;
8. IND = IND + 1 ;
9. PP  = EPP(IND) ;
10. SE TAMREC(IND) = 0
      ENTÃO VA P/ 15 ; /* TERMINOU MAS VOLTA A INTERROGAR
          FIMFA
```

```
11. PEDE-IMPRESSÃO TABMENS(IND) ;
12. CONTAM = 0 ; /* CONTADOR DOS CARACTERES RECEBIDOS */
13. ENQUANTO CONTAM < TAMREC(IND)
      FAÇA
         LE CARACTER ; /* FAZ PEDIDO */
         CONTAM = CONTAM + 1 ;
         SE CARACTER = <ENTRE>
            ENTÃO VA P/ 24 ;
         TABREC(IND,CONTAM) = CARACTER ;
      FIMFA
14. VA P/ PP ; /* PROX. PASSO SEGUND MENS. RECEBIDA */

    /* VERIFICA INFORMAÇÃO TECLADA */

15. PPE = 24 ; /* EMITE ERRO E RECOMEÇA */
16. PPC = 26 ; /* CARREGA TRATAMENTO */
17. PEDE-IMPRESSÃO
       'SE QUISER ALTERAR ALGUMA INFORMAÇÃO,
        TECLE O NÚMERO SEGUNDO TABELA ABAIXO:
             SENHA - 1
             SIGLA - 2
             DISCO - 3
             NENHUMA - 4' ;
18. LÊ CARACTER ;
19. IND = CARACTER ;
20. SE CARACTER = 4
       ENTÃO FAÇA
             PP = 26 ;
             VA P/ 11 ;
          FIMFA
       SENÃO VA P/ 9 ;
```

```
    /* TESTA SENHA */

21. SE TABREC(IND)   ≠ SENHA-DO-SISTEMA
       ENTÃO FAÇA
             PEDE-IMPRESSÃO 'SENHA ERRADA' ;
             VA P/ PPE ;
          FIMFA
       SENÃO VA P / PPC ;

    /* TESTA SIGLA */

22. SE TABREC(IND) = 'NÃO-EXISTE'
       ENTÃO FAÇA
             PEDE-IMPRESSÃO 'SIGLA ERRADA'
             VA P/ PPE ;
          FIMFA
       SENÃO VA P/ PPC ;

    /* TESTA DISCO */

23. SE TABREC(IND) = 'NÃO-EXISTE'
       ENTÃO FAÇA
             PEDE-IMPRESSÃO 'DISCO NÃO EXISTE' ;
             VA P/ PPE ;
          FIMFA
       SENÃO VA P/ PPC ;

    /* ROTINA DE ERRO */

24. PEDE-IMPRESSÃO
             'CERTIFIQUE-SE DAS INFORMAÇÕES E
              TENTE TUDO NOVAMENTE
              PARA UTILIZÁ-LO TECLE OI' ;
```

```
25. VA P/ 1 ;

    /* FINALIZA ABERTURA E CARREGA TRATAMENTO */

26. PARÂMETRO_DIRETOR_DISCO = TABREC ;
27. PEDE-CARGA-ROTINA-DE-TRATAMENTO ;
28. PARE ;  /* SÓ É CARREGADA PELA ROTINA DE FECHAMENTO */
```

3.3.2 - A ROTINA DE TRATAMENTO

A rotina de tratamento, ativada através de um depósito feito pelo monitor no módulo MESTRE, é responsável pela interpretação do comando teclado e ativação do processo que executa a tarefa solicitada pelo usuário. Para que um comando seja devidamente aceito e um processo invocado, são necessários certos procedimentos contidos nesta rotina, onde as informações recebidas são avaliadas e submetidas a tratamentos, para que sejam devidamente recebidas pelas rotinas a serem invocadas.

3.3.2.1 - INFORMAÇÕES DE ENTRADA

As informações de entrada da rotina de tratamento, ou seja, o comando teclado pelo usuário, estão contidas num buffer de 80 bytes, as quais foram depositadas pelo monitor no módulo MESTRE. O comando encontra-se da mesma forma em que foi teclado, exceto o caracter <CMD> que serviu apenas para decisão de entrega para o Manipulador. Qualquer tipo de informação pode estar contida no buffer, sendo que a interpretação da mesma dirá se existe ou não compatibilidade com as informações a serem aceitas pelo sistema.

3.3.2.2 - DEFINIÇÃO DOS PROCEDIMENTOS.

Quando o buffer é preenchido e entregue à rotina de tratamento, é iniciada a execução de passos representados pelo diagrama a seguir:

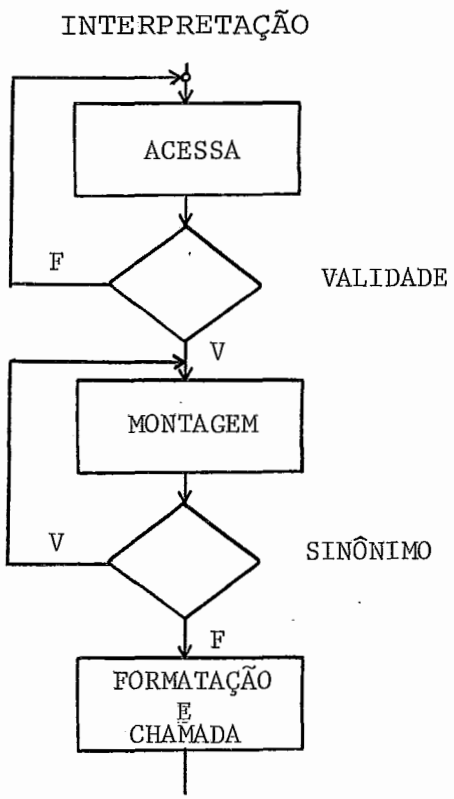


FIG. 10

A primeira fase da interpretação resume-se em acessar os três primeiros caracteres do comando e fazer uma busca sequencial na tabela de comandos - para identificar sua existência ou não. Caso o comando não seja encontrado, a subrotina do núcleo, 'DEPOSITE', é chamada para que seja feito um pedido de impressão ao impressor do SIRIUS, a fim de informar ao usuário sua inexistência. Quando um comando é encontrado, a segunda fase da interpretação é iniciada onde é feita a montagem dos parâmetros em função dos identificadores constantes no campo 'DERIVADA', da tabela de comandos. Uma

área auxiliar do mesmo tamanho da derivada é utilizada, para que as informações sejam transferidas da derivada ou do buffer, de acordo com o posicionamento dos identificadores.

Essa área auxiliar (AUX), conterá a forma final do comando, após a montagem de parâmetros . Para cada montagem de um comando 'sinônimo', a mesma é transferida para o buffer de entrada, sendo iniciada uma nova interpretação.

Quando um 'EXE' é encontrado, o nome 'SISTEM.TRAB.EXE' é passado para o carregador, a menos que um nome seja mencionado. Nesse caso, o mesmo é adicionado à extensão '.EXE' e passado para o módulo carregador.

Após a fase de montagem, a rotina desvia para o endereço contido no campo 'E' da tabela, onde são feitos os procedimentos de formatação e chamada da subrotina do núcleo, responsável pelo tratamento da forma final do comando, após sua formatação.

A chamada é feita e a rotina de tratamento fica em estado de espera, aguardando uma nova ativação feita pelo monitor.

A seguinte tabela de comandos é utilizada nessa versão:

TABELA DE COMANDOS

COM	TP	DERIVADA	E
AJU	10	SIS:SIS.!.SIS	-
APA	00	0 ? SIS:SIS.TRAB.TXT	54
ASS	00	! ! !	37
COP	00	SIS:SIS.COPIE.EXE ! ! ?	29
CRI	04	TECLA: !	-
DAT	00	SIS:SIS.DATA.EXE	29
DEP	00	SIS:SIS.DEPURE.EXE	29
DIR	00	SIS:SIS.DIRETO.EXE ?	29
EDI	00	SIS:SIS.EDITE.EXE	29
LIS	04	! TELA:	-
MUD	00	2 ! !	54
SAL	00	SIS:SIS.SALVE.EXE	29
SEL	04	SIS:SIS.TRAB.TXT	-
SIN	00	SIS:SIS.SINTAX.EXE	29
SIS	00	SIS:SIS.SISTEM.EXE ?	29
SUB	00	SIS:SIS.SUBMET.EXE !.LIC	29
SUS	00	!	35
TCH	00	SIS:SIS.TCHAU.EXE	29

- DEFINIÇÃO DOS CAMPOS

. COMANDO - Os três primeiros caracteres do comando a ser interpretado.
(3 bytes)

. T - Tipo do comando, podendo assumir os seguin-

tes valores:

= 0 - Comando gerador

≠ 0 - Comando sinônimo, onde seu conteúdo aponta para o próximo comando a ser interpretado.

(1 byte)

. DERIVADA - Define a forma final do comando recebido.

Os seguintes identificadores são utilizados:

! - Parâmetro obrigatório

? - Parâmetro opcional

(45 bytes)

. E - Número do passo do algoritmo (endereço na rotina), onde será feita a formatação e chamadas as subrotinas do núcleo, localizadas nos seguintes passos:

29 - Deposite (loader)

35 - Termine

37 - Associe

54 - Especial (segmes)

(1 byte)

- MONTAGEM

Após o teste de validade do comando é iniciada a montagem dos parâmetros, onde a área de saída do algoritmo é preenchida por informações da derivada e do buffer de entrada. A transferência dos parâmetros segue uma forma padrão para qualquer comando encontrado na

tabela, baseando-se nas seguintes condições:

1 - A área auxiliar utilizada, tem o mesmo tamanho do buffer de entrada (80 bytes), sobre o qual poderá ser transferida, no caso de comandos sinônimos.

2 - Quando um '!' é encontrado na derivada, a próxima 'string' contida no buffer de entrada é transferida para a área auxiliar. Caso não exista essa 'string', significa que um parâmetro obrigatório não foi encontrado no buffer, o que é informado ao usuário através do impressor do SIRIUS, e a rotina interrompida. Após a transferência de um parâmetro obrigatório, a rotina volta a pesquisar uma nova ocorrência de identificador na derivada.

3 - Quando um '?' é encontrado, implica que pode ou não haver uma 'string' no buffer de entrada. Se existir, a mesma é transferida para a área auxiliar e a montagem é encerrada. Se não existir, o restante da derivada é transferido para a área auxiliar, encerrando a montagem.

4 - Sempre que um '?' ocorre na derivada, o parâmetro correspondente no buffer de entrada, deverá ser a última informação a ocorrer dentro do mesmo.

5 - Todos os campos da derivada estão separados por apenas um espaço em branco.

6 - Para a formatação de 'strings' é utilizada uma matriz

(3,47), onde cada linha representa uma 'string' a ser passada. O formato da 'string' a passar, é o mesmo definido na 'ESTRUTURA DOS DADOS' do núcleo, com os seguintes campos:

Tamanho máximo - (1 byte)

Tamanho ocupado - (1 byte)

Dados - (45 bytes)

7 - Para a passagem de informações para o módulo carregador, é utilizada uma área cujo conteúdo é formado pelos seguintes dados:

ÁREA(1,*) - Nome do módulo a carregar

ÁREA(2,*) - Endereço de parâmetros a passar para o módulo a ser carregado.

- VARIÁVEIS UTILIZADAS

As seguintes variáveis com seus respectivos significados, são utilizadas no algoritmo:

BUFFER - Área onde se encontra o comando teclado.

COMTEC - Contém os 3 primeiros caracteres do comando teclado.

FIMCOM - Aponta para o final do comando teclado.

AUX - Área auxiliar.

IND - Índice do comando encontrado na tabela.

STRING(i,j) - String a ser formatada e enviada.

TAMA - Tamanho do nome encontrado na aux.

CPARM - Contador de parâmetros a serem passados.

ÁREA(1,*) - Nome do módulo a carregar.

ÁREA(2,*) - Endereço dos parâmetros a passar.

BCES - Bloco de controle de e/s

DADOS - Recebe endereço dos dados a passar para 'DEPOSITE'.

SERV - Número do serviço.

A - Número do carregador.

B - Número do disco (segmes)

C - Número do impressor do SIRIUS.

3.3.2.3 - ALGORITMO

```
/* ROTINA DE TRATAMENTO */

1. J = 1 ;
2. ENQUANTO BUFFER(J) = ' ' & J < 80
            FAÇA J = J + 1 ;  /* ENCONTROU COMANDO */
3. SE J = 80 ENTÃO PARE ;  /* NÃO TEM COMANDO */
4. COMTEC = BUFFER(J) ;  /* PEGA AS 3 PRIM. POSIÇÕES */
5. ENQUANTO BUFFER(J)   =  ' '
            FAÇA J = J + 1 ;  /* ENCONTRA FINAL DO COMANDO */
6. FIMCOM = J ;  /* GUARDA FINAL DO COMANDO */
7. FAÇA /* ECOA COMANDO TECLADO */
      AUX = COMTEC ;
      SERV.BCES = C ;
      END.DADOS.BCES = END(AUX) ;
   FIMFA
8. CHAMA ESCREVA(END(BCES)) ;  /* PEDE IMPRESSÃO */
9. SE COMTEC   ≠ 'EXE' ENTÃO VA P/ 13 ;
```

```
10. ENQUANTO BUFFER(J) = ' ' & J <= 80
            FAÇA J = J + 1 ; /* PROCURA NOME DO COMANDO */
11. SE J = 80
       ENTÃO FAÇA /* NÃO ENCONTROU O NOME */
            ÁREA(1,*) = 'SISTEM.TRAB.EXE'; /* CARREGAR */
            ÁREA(2,*) * ' ' ; /* END. PARÂMETROS */
            DADOS = END(ÁREA) ;
            CHAMA DEPOSITE(A,END(DADOS)) ;
          FIMFA
       SENÃO FAÇA
            L = 0 ;
            ENQUANTO BUFFER(J)  = '' &J <=80
               FAÇA
                  ÁREA(1,L) = BUFFER(J) ;
                  J = J + 1 ;
                  L = L + 1 ;
                FIMFA
               ÁREA(1,L+1) = '.EXE' ;
               ÁREA(2,*) = ' ' ; /* END. PARÂMETROS */
               DADOS = END(AREA) ;
               CHAMA DEPOSITE(A,END(DADOS)) ;
             FIMFA
12. PARE ; /* ESPERA ATIVAÇÃO */

    /* TRATA COMANDO INDIRETO */

       /* BUSCA SEQUENCIAL */

13. COMANDO(19) = COMTEC ;
14. IND = 1 ;
15. ENQUANTO COMANDO(IND)   ≠ COMTEC
```

```
        FAÇA IND = IND + 1 ;
16. SE IND = 19
        ENTÃO FAÇA /* PEDE IMPRESSÃO */
                AUX = 'COMANDO INEXISTENTE' ;
                SERV.BCES = C ;
                END.DADOS.BCES = END(AUX) ;
                CHAMA ESCREVA(END(BCES)) ;
                PARE ;
              FIMFA

    /* MONTAGEM */

17. J = 1 ; /* INDICE QUE PERCORRE A DERIVADA */
18. K = 0 ; /* INDICE QUE PERCORRE A AUXILIAR */
19. I = FIMCOM ; /* PEGA FINAL DO COMANDO */
20. SE DERIV(IND,J) = ' '
        ENÃO FAÇA /* PROCURA OPCIONAL */
                ENQUANTO BUFFER(I) = ' ' & I < 80
                   FAÇA I = I + 1 ;
                SE I = 80
                   ENTÃO FAÇA /* NÃO TEM NO BUFFER */
                      J = J + 1 ;
                      ENQUANTO J ← 24
                         FAÇA
                            K = K + 1 ;
                            AUX(K) = DERIV(IND,J) ;
                            J = J + 1 ;
                          FIMFA
                       FIMFA
                  SENÃO FAÇA /* TEM INFORMAÇÃO */
```

```
                ENQUANTO BUFFER(I)     ≠ ' '
                   FAÇA
                      K = K + 1 ;
                      AUX(K) = BUFFER(I)
                      I = I + 1 ;
                   FIMFA
                FIMFA
          FIMFA
    SENÃO SE DERIV(IND,J) = '|'
          ENTÃO FAÇA /* PROCURA OBRIGATÓRIA */
             ENQUANTO BUFFER(I) = ' ' & I <= 80
                FAÇA I = I + 1 ;
             SE I = 80
                ENTÃO FAÇA /* NÃO TEM INFO. */
                      AUX = 'FALTA PARÂMETRO' ;
                      SERV.BCES = C ;
                      END.DADOS.BCES = END(AUX) ;
                      CHAMA ESCREVA (END(BCES)) ;
                      PARE ;
                   FIMFA
                SENÃO FAÇA /* TEM INFO. */
                      ENQUANTO BUFFER(I)   ≠ ' '
                         FAÇA
                            K = K + 1 ;
                            AUX(K) = BUFFER(I) ;
                            I = I + 1 ;
                         FIMFA
                      FIMFA
             FIMFA
          SENÃO FAÇA /* COLOCA CARACTER NA AUX */
```

```
                    K = K + 1 ;
                    AUX(K) = DERIV(IND,J) ;
                  FIMFA
21. SE J >= 45
       ENTÃO VÁ P/24 ;  /* ACABOU MONTAGEM */
22. J = J + 1 ;  /* APONTA P/PROX. CARAC. NA DERIV. */
23. VÁ P/20 ;  /* VOLTA A INTERPRETAR */

    /* IDENTIFICA O TIPO */

24. SE TIPO(IND) = 0 /* COMANDO GERADOR */
       ENTÃO VA P/E(IND);  /* FORMATAÇÃO E CHAMADA */

    /* COMANDO SINÔNIMO */

25. FIMCOM = 1;  /* ATUALIZA INÍCIO DO BUFFER */
26. BUFFER = AUX;  /* COLOCA AUX. NO BUFFER */
27. IND = TIPO(IND);  /* APONTA P/SINÔNIMO */
28. VÁ P/17;  /* RECOMEÇA A MONTAGEM */

    /* FORMATAÇÃO E MONTAGEM */
      /* DEPOSITE (LOADER) /*

29. I = 1;
30. ENQUANTO AUX(I)   ≠ ' '  /* PEGA NOME DO MÓDULO */
       FAÇA
          ÁREA(1,I) = AUX(I);
          I = I + 1;
        FIMFA
31. ÁREA(2,*) = END(AUX(I+1);  /* PEGA END. DOS PARAM */
32. DADOS = END(ÁREA);
```

```
33. CHAMA DEPOSITE(A,END(DADOS));
34. PARE;  /* ESPERA ATIVAÇÃO */

    /* TERMINE */

35. CHAMA TERMINE(AUX);
36. PARE;  /* ESPERA ATIVAÇÃO */

    /*ASSOCIE */

37. I = 1 ;
38. L = 1 ;
39. CPARM = 0;  /* ZERA CONTADOR DE PARAM. */
40. TAMA = 0;  /* ZERA TAMANHO DA STRING */
41. J = 1 ;
42. STRING(I,J) = 26;
43. J = 2;
44. ENQUANTO AUX(L)   ≠ ' '
          FAÇA
             TAMA = TAMA + 1 ;
             STRING(I,J+TAMA) = AUX(L) ;
             L = L + 1 ;
           FIMFA
45. STRING(I,2) = TAMA;  /* PEGA TAMANHO DA STRING */
46. CPARM = CPARM + 1 ;
47. SE CPARM = 3
       ENTÃO FAÇA
             CHAMA ASSOCIE(STRING(*,*)) ;
             PARE;  /* ESPERA ATIVAÇÃO */
           FIMFA
48. ENQUANTO AUX(L) = ' '
```

```
        FAÇA L = L + 1;  /* PROCURA PROX. PARAM. */
49. I = I + 1 ;
50. TAMA = 0
51. VÃ P/41;  /* VOLTA A MONTAR PROX. STRING */

    /* SGMES */

52. COD.OP.BCES = AUX(1);
53. END.DADOS.BCES = END(AUX(3);
54. NUM.SERV.BCES = B;  /* NÚMERO DO SGMES */
55. CHAMA ESPECIAL(END(BCES));
56. PARE;  /* ESPERA ATIVAÇÃO */
```

3.3.3 - ROTINA DE FECHAMENTO

A rotina de fechamento do módulo MESTRE é carregada e executada quando um comando 'TCHAU' é teclado pelo usuário. Essa rotina encontra-se no diretório do sistema, identificada pelo nome 'SIS:SIS.TCHAU.EXE', e tem como principal função a carga da rotina de abertura, anteriormente apagada pela rotitina de tratamento quando foi dada sua carga.

3.3.3.1 - DEFINIÇÃO DOS PROCEDIMENTOS

Quando o comando 'TCHAU' é encontrado, é feita uma chamada à subrotina 'DEPOSITE' do núcleo, onde é passado o número do módulo carregador e o nome do arquivo 'SIS:SIS.TCHAU.EXE'. Esse módulo faz novas chamadas à subrotina 'DEPOSITE', onde pede a carga e execução do módulo 'SIS:SIS.ABRE.EXE', rotina de abertura, seguido de outro pedido para o módulo -

'SIS:SIS.COPIE.EXE', com os parâmetros 'SIS:SIS.CONTAB. SIS.TELA:'. A execução do 'COPIE' implica na listagem do arquivo mencionado, onde estão contidas informações sobre a utilização do usuário.

3.3.3.2 - ALGORITMO

1. AUX = 'SIS:SIS.ABRE.EXE'
2. CHAMA DEPOSITE (A,END(AUX));
3. AUX = 'SIS:SIS.COPIE.EXE SIS:SIS.CONTAB.EXE TELA:' ;
4. CHAMA DEPOSITE (A,END(AUX));
5. PARE

3.4 - MÓDULO SECUNDÁRIO

Como já descrito, a 'RESULTANTE' da interpretação de um comando é a chamada de subrotinas do núcleo, que executam procedimentos de acordo com suas funções internas. As funções de uma subrotina nem sempre estão diretamente ligadas a tarefa solicitada pelo usuário ou por um processo qualquer. Com isso , se a subrotina chamada contiver procedimentos que executem diretamente a tarefa solicitada (SUS,ASS), o nível de chamadas termina nesse ponto, uma vez que sua execução foi terminada.

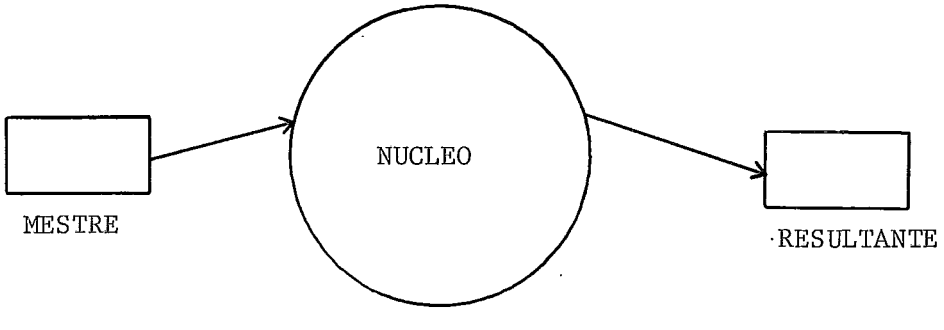


FIG. 11

Todavia, nem sempre o significado do comando teclado tem procedimentos correspondentes dentro de subrotinas do núcleo. Nesse caso, a subrotina 'DEPOSITE' é chamada, onde é solicitada a carga e execução de um novo módulo. Esse novo módulo por sua vez, poderá ser um 'módulo independente' (edi - tor,compilador,etc), ou um módulo que contem procedimentos referentes à solicitação feita pelo usuário (módulo executável).

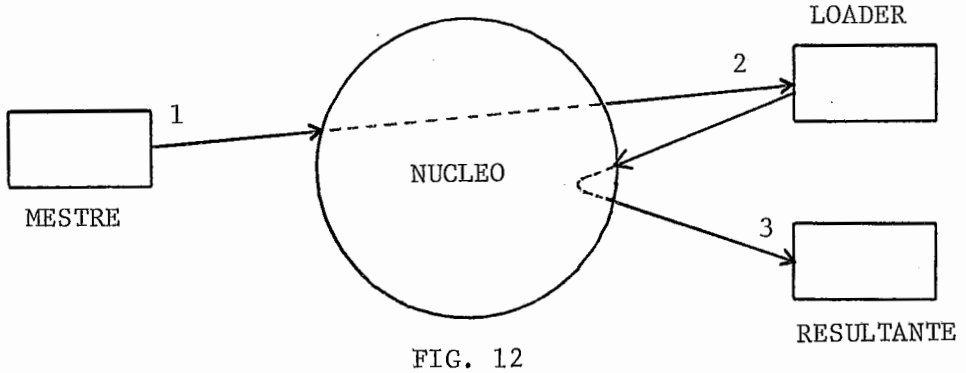


FIG. 12

O módulo secundário é composto por um conjunto de módulos executáveis, os quais contêm procedimentos referen - tes ao comando teclado, e que não estão contidos em subrotinas do núcleo. Com isso, os módulos executáveis podem conter mais de uma diretiva para o núcleo que, unidas às suas instruções , compõem os procedimentos necessários para a execução da tarefa desejada.

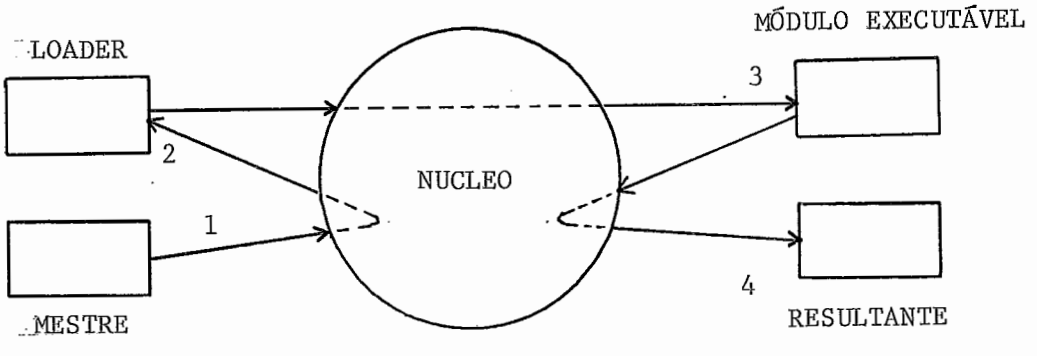


FIG. 13

Nessa versão do SIRIUS, os seguintes módulos são descritos: COPIE, DATA, SALVE, SUBMETA, SISTEMA e DIRETÓ - RIO.

3.4.1 - MÓDULO DATA

O módulo 'DATA' tem como objetivo informar ao usuário a data do dia e a hora, segundo o relógio do sistema.Para isso, diretivas são dadas ao núcleo, onde são invocadas as subrotinas responsáveis pela entrega dessas informações.

3.4.1.1 - FUNCIONAMENTO

São três as subrotinas chamadas: DATA HORA e TICCAR. A subrotina 'DATA' fornece a data corrente, no seguinte formato: DD/MM/AA. A cadeia fornecida é acessada e transferida para a área de impressão , posteriormente enviada para o usuário.

A subrotina 'HORA' fornece a hora corrente em formato binário, cuja unidade está em tic's. A próxima subrotina a ser chamada é a 'TICCAR' que acessa a hora devolvida pela subrotina 'hora' e converte para a forma 'HH:MM:SS', a qual também é transferida para a área de impressão. Após o preenchimento da área de impressão, uma 'BCES' é preenchida, e um pedido é feito ao impressor do SIRIUS, onde é passado o endereço da BCES, o que resultará na impressão dos dados no seguinte formato:

DATA - DD/MM/AA

HORA - HH:MM:SS

3.4.1.2 - ALGORITMO

Para a formatação das informações a serem impressas, o módulo utiliza uma área de impressão para onde são transferidas as informações recebidas . Além dessa área também são utilizadas outras áreas para que sejam passadas e recebidas informações das subrotinas chamadas.

As seguintes variáveis são utilizadas:

ARIMP - Conjunto de 31 elementos, inicializado com os seguintes valores:

DATA - XXXXXXXX/HORA - XXXXXXXX

AREC - Cadeia de recebimento das informações

HT - Palavra dupla onde é recebida a hora em tic's.

```
/* MÓDULO DATA */

1. CHAMA DATA(AREC);
2. FAÇA I DE 1 A 8;
      ARIMP(I+7) = AREC(I);
   FIMFA
3. AREC = ' ' ;
4. CHAMA HORA(END(HT));
5. CHAMA TICCAR(END(HT),END(AREC));
6. FAÇA I DE 1 A 8;
      ARIMP(I+23) = AREC(I);
   FIMFA
7. PREPARA BCES;  /* COLOCA NUM. DO IMP. */
8. CHAMA ESCREVE(END(BCES));
9. CHAMA TERMINE(DATA);  /* DESATIVA-SE */
```

3.4.2 - MÓDULO COPIE

O módulo 'COPIE' tem como objetivo reproduzir as informações contidas num determinado arquivo, num outro arquivo definido pelo usuário. Tanto o arquivo original quanto o arquivo destino, podem estar associados a qualquer periférico desde que estejam de acordo com as características do sistema . O módulo executável 'COPIE' é carregado e executado quando um pedido é depositado no módulo MESTRE, podendo este ser resultado de um comando 'COPIE' teclado pelo usuário, resultado de qualquer outro comando sinônimo ao COPIE, ou depósito feito por qualquer outro processo do sistema.

3.4.2.1 - DEFINIÇÃO DOS PROCEDIMENTOS

O módulo 'COPIE' tem como procedimentos básicos, a leitura de informações do arquivo de origem e a escrita das mesmas no arquivo destino. Para que esses procedimentos tenham fundamentos genéricos, ou seja, qualquer processo do sistema possa utilizá-los, é necessário o tratamento de informações contidas no pedido depositado, as quais definem as especificações do arquivo origem e destino. Esses procedimentos dividem-se em duas fases onde são identificados os arquivos origem e destino. Na primeira fase é feito um pedido de abertura do arquivo origem, onde é devolvido um 'status' informando a existência ou não desse arquivo. Caso o arquivo não exista, o módulo é interrompido e fica esperando uma nova ativação; caso contrário, a 'BCES' devolvida é acessada onde é identificado o formato do arqui-

vo e o número do módulo onde serão depositados os pe - didos de leitura.

Na segunda fase, um novo pedido de abertura é feito, sendo que em relação ao arquivo destino. O teste de sua existência é feito através do 'status' devolvido, sendo definido seu formato de acordo com o parâmetro de entrada (opcional) ou igual ao do arquivo de origem caso o mesmo não tenha sido men - cionado. Logo após, o número do serviço onde serão depositados os pedidos de escrita, é definido, e as operações de leitura e escrita são iniciadas.

Os pedidos de abertura feitos pelo módulo COPIE, são feits através de chamadas à subrotina 'abra-arquivo' do núcleo. Para isso, uma 'BCES' é preparada para cada pedido de abertura e a subrotina chamada, sendo passada essa BCES. Qualquer inconsis - tência na abertura desses arquivos, o 'SGMES' (sistema de gerenciamento e manipulação de e/s) informa ao processo de onde originou o pedido de execução do CO - PIE. No caso do MANIPULADOR, o impressor do SIRIUS é invocado.

3.4.2.2 - INFORMAÇÕES DE ENTRADA

As informações tidas como entrada do módulo 'COPIE' estão contidas numa área de dados passada pelo módulo carregador, logo após o módulo COPIE ser carregado. A área de dados é passada para o COPIE a - través do seu endereço, cujo conteúdo contém as seguintes informações:

- Nome do arquivo origem
- Nome do arquivo destino
- Formato do arquivo destino (opcional)

Com a passagem do 'endereço' da área de dados pelo carregador, o conteúdo da mesma fica transparente a esse módulo, sendo acessada somente pelo CO - PIE.

Na execução dos procedimentos de leitura e escrita, os dados de entrada e saída encontram - se numa área de trabalho reservada para a transferência de informações da 'BCES' de entrada e para a 'BCES' de saída.

3.4.2.3 - TRATAMENTO DAS INFORMAÇÕES

Inicialmente as informações contidas na área de dados são decompostas, onde são identificados os nomes do arquivos e o formato do arquivo destino. Para isso, um conjunto de dois elementos é utilizado, Nome(2,24), onde seus elementos receberão os nomes dos arquivos encontrados na área recebida. Após a decomposição da área de dados, é iniciada a abertura dos arquivos para que sejam definidas as especificações dos arquivos origem e destino. O formato do arquivo destino é tratado de forma que sejam definidos os procedimentos para registros de tamanho fixo ou variável, segundo as seguintes regras:

FTN - Para todo registro cujo tamanho seja maior que'N' só serão consideradas as 'N' primeiras posições ,

sendo o restante desprezado.

FN - Para todo registro cujo tamanho seja maior que 'N', será gerado um novo registro com o restante das informações.

V - Registro de tamanho Variável

Quando uma leitura é feita, é devolvida pelo núcleo a BCES de entrada, e o registro lido é colocado na área de trabalho (ARTRAB), apontada pela BCES. Para os arquivos que têm tamanho fixo ou igual ao do arquivo de entrada, apenas é feita uma chamada à subrotina ESCREVA do núcleo, onde o formato de saída já está atualizado, e a área de dados (ARTRAB) é apontada pela BCES de saída - (BCES.s). Para os arquivos que têm tamanho variável, mesmo que seja igual ao do arquivo origem por omissão do formato, é atualizado o tamanho do registro de saída para cada registro de entrada lido. Os arquivos que têm formato fixo, considerando todas as informações de um registro de entrada maior que seu tamanho, são tratados de maneira que, para cada 'bloco' de informações do registro lido cujo tamanho seja maior que seu formato, seja gravado um novo registro de saída. Para isso, para cada gravação de registro novo, o apontador da área de dados na BCES de saída é incrementado do tamanho do registro de saída, e uma nova chamada à subrotina ESCREVA é feita, até que todo registro de entrada seja gravado. Caso o arquivo a

ser gravado seja o arquivo de trabalho, o nome e o formato do arquivo origem são guardados,o qual será futuramente acessado pelo módulo 'SALVE'.

3.4.2.4 - ALGORITMO

As seguintes variáveis são utilizadas:

BCES.E	- BCES de entrada
BCES.S	- BCES de saída
ARTRAB	- Área de trabalho
NOME(2,24)	- Cadeias de nomes
FORM(8)	- Formato recebido
BUFFER	- Buffer de entrada
TAMREG	- Tamanho do registro
FORMREG	- Formato do registro
DESVI1	- Endereço de desvio
DESVI2	- Endereço de desvio
NOMEFONTE	- Guarda nome do original
FORMFONTE	- Guarda formato do original
TAMFONTE	- Guarda tamanho do original

```
/* MÓDULO COPIE */

1. I = 1; DESVI1 = 14; DESVI2 = 19 ;
2. FAÇA J = 1 A 2 /* PEGA OS NOMES */
      K = 2 ;
      ENQUANTO BUFFER(I)   ≠  ' '
            FAÇA
               K = K + 1 ;
               NOME(J,K) = BUFFER(I) ;
```

```
                    I = I + 1 ;
                FIMFA
        NOME(J,2) = K ;  /* COLOCA TAMANHO NA CADEIA */
        I = I + 1 ;
    FIMFA
3. ENQUANTO BUFFER(I) = ' ' & I <= 80 /* PROCURA FORM */
        FAÇA I = I + 1 ;
4. SE I <= 80
    ENTÃO FORM(*) = BUFFER(I) ; /* PEGA FORMATO */
5. NOME(1,1) = 24 ; /* PREPARA CADEIAS */
6. NOME(2,1) = 24 ;
7. END.ARTRAB.BCES.E = END(ARTRAB) ;
8. CHAMA ABRA-ARQUIVO(NOME(1,*),END(BCES.E)) ;
9. SE STATUS.BCES.E = 'INVALIDO'
    ENTÃO CHAMA TERMINE ; /* DESATIVA-SE */
10. SE FORM(*) = ' '
    ENTÃO FAÇA
        FORMREG.BCES.S = FORMREG.BCES.E ;
        SE FORMREG.BCES.E = 'V'
            ENTÃO DESV1 = 18 ;
            SENÃO TAMREG.BCES.S = TAMREG.BCES.E ;
        FIMFA
    SENÃO SE FORM(1) = 'F'
        ENTÃO FAÇA
            FORMREG.BCES.S = 'F' ;
            SE FORM(2) = 'T'
                ENTÃO FAÇA I = 3 ;
                SENÃO FAÇA
                    DESVI2 = 21 ;
                    I = 2 ;
```

```
                    TAMREG.BCES.S = FORM(I) ;
                FIMFA
            SENÃO FAÇA
                    FORMREG.BCES.S = 'V' ;
                    DESVI1 = 18 ;
                FIMFA
11. END.ARTRAB.BCES.S = END(ARTRAB) ;
12. CHAMA ABRA-ARQUIVO(NOME(2,*),END(BCES.S)) ;
13. SE STATUS.BCES.S = 'INVALIDO'
       ENTÃO CHAMA TERMINE ; /* DESATIVA-SE */

    /* LEITURA E ESCRITA */

14. ARTRAB = ' ' ;
15. CHAMA LEIA(END(BCES.E)) ;
16. SE STATUS.BCES.E = 'FIM-DE-ARQUIVO'
       ENTÃO VA P/ 23 ;
17. VA P/ DESVI1 ;
18. TAMREG.BCES.S = TAMREG.BCES.E ;
19. CHAMA ESCREVA(END(BCES.S)) ;
20. VA P/DESV12 ;
21. TAMREG.BCES.E = TAMREG.BCES.E - TAMREG.BCES.S ;
22. SE TAMREG.BCES.E > 0
       ENTÃO FAÇA
             END.ARTRAB.BCES.S = END.ARTRAB.BCES.S +
                                 TAMREG.BCES.S ;
             VA P/ 19 ;
          FIMFA
       SENÃO FAÇA
             END.ARTRAB.BCES.S = END(ARTRAB) ;
```

```
            VA P/ 14 ;
          FIMFA

    /* FIM DE ARQUIVO */

23. CHAMA FECHA-ARQUIVO(END(BCES.E)) ;
24. CHAMA FECHA-ARQUIVO(END(BCES.S)) ;
25. SE NOME.BCES.S = 'TRAB'
       ENTÃO FAÇA /* GUARDA ORIG. P/ SALVE */
             NOMEFONTE = NOME(1) ;
             FORMFONTE = FORMREG.BCES.E ;
             TAMFONTE = TAMREG.BCES.E ;
          FIMFA

    /* PREPARA IMPRESSÃO */

26. ARTRAB = 'CÓPIA TERMINADA' ;
27. NUM.SERV.BCES.S = NUM.IMPR.SIRIUS ;
28. CHAMA ESCREVA(END(BCES.S)) ;
29. PARE ;
```

3.4.3 - MÓDULO SALVE

O módulo SALVE tem como objetivo o salvamento do arquivo de trabalho, TRAB.TXT, anteriormente selecionado pelo usuário. O salvamento desse arquivo implica na sua cópia para um outro arquivo, cujas especificações são as mesmas do arquivo fonte que originou o arquivo de trabalho, através do comando selecione. Esses dados previamente guardados pelo módulo 'COPIE', são acessados pelo módulo SALVE, e o salvamento iniciado onde o arquivo-origem é o arquivo de trabalho e o destino é um arqui

vo cujas informações foram salvas do arquivo de trabalho.

3.4.3.1 - DEFINIÇÃO DOS PROCEDIMENTOS

A idéia básica do módulo SALVE é que as informações contidas no arquivo de trabalho sejam devolvidas ao arquivo que as originou. As seguintes considerações são feitas para o módulo SALVE: o formato dos arquivos e a criação de um novo arquivo a partir do arquivo de trabalho.

- Formatos do fonte e do trabalho iguais.

Quando o formato do arquivo fonte for igual ao do arquivo de trabalho, apenas a troca de nomes é necessária. Isto é, se um nome de arquivo for mencionado no comando de entrada, o nome do arquivo de trabalho é trocado por esse nome, onde é feito um pedido de troca ao MANIPULADOR de disco (SGMES). Se um nome não for mencionado, é feito um pedido de deleção do arquivo fonte selecionado e, em seguida, é feito um pedido, onde o nome do arquivo de trabalho é trocado pelo nome do arquivo fonte.

- Formatos do fonte e do trabalho iguais.

Neste caso, um pedido de cópia de arquivo é efetuado, onde o arquivo origem é o arquivo de trabalho e o arquivo destino é o mencionado (opcio - nalmente) no comando, cujo formato é o mesmo do arquivo fonte. Caso um arquivo não seja mencionado no comando de entrada, é feito um pedido de deleção do ar-

quivo fonte e, em seguida, a cópia é efetuada, onde o arquivo-destino terá o mesmo nome e formato do ar - quivo fonte.

3.4.3.2 - ALGORITMO

As seguintes variáveis são utiliza - das:

AUX	-	Área auxiliar.
BUFFER	-	Contém ou não o nome do arquivo.
BCES.E	-	BCES do arquivo de trabalho.
BCES.S	-	BCES do arquivo salvo.
NOMENOVO	-	Nome do arquivo salvo.
ÁREA(1,*)	-	Módulo a ser carregado.
ÁREA(2,*)	-	End. dos param. do módulo a ser carreg.
DADOS	-	End. dos param. do carregador.

```
/* MÓDULO SALVE */

1. AUX = 'SIS:SIS.TRAB.TXT' ;
2. CHAMA ABRA-ARQUIVO(AUX.END(BCES.E) ;  /* ABRE TRAB. */
3. SE BUFFER = ' '
      ENTÃO FAÇA
            COD.OP.BCES.S = 0 ;  /* CÓDIGO DELEÇÃO */
            NOME.ARQ.BCES.S = NOMEFONTE ;
            NUM.SERV.BCES.S = B ;  /* SGMES */
            CHAMA ESPECIAL(END(BCES.S)) ;  /* PEDE DELEÇÃO */
            NOMENOVO = NOME FONTE ;
      SENÃO NOMENOVO = BUFFER ;
4. SE FORMREG.BCES.E ≠ FORMFONTE
```

```
        ENTÃO VA P/ 9 ;
        SENÃO SE TAMREG.BCES.E ≠ TAMFONTE
                ENTÃO VA P/ 9 ;

    /* FORMATOS IGUAIS */

 5. COD.OP.BCES.S = 2 ;  /* CÓDIGO TROCA */
 6. END.DADOS.BCES.S = END(NOMENOVO) ;
 7. CHAMA ESPECIAL(END(BCES.E)) ;  /* PEDE TROCA */
 8. CHAMA TERMINE ;  /* DESATIVA-SE */

    /* FORMATOS DIFERENTES */

 9. ÁREA(1,*) = 'SIS:SIS.COPIE.EXE' ;
10. AUX(*) = 'SIS:SIS.TRAB.TXT' ;
11. AUX(18) = NOMENOVO ;
12. AUX(42) = FORMFONTE ;
13. AUX(43) = TAMFONTE ;
14. ÁREA(2,*) = END(AUX) ;
15. DADOS = END(ÁREA(*,*)) ;
16. CHAMA DEPOSITE(AEND(DADOS)) ;  /* PEDE EXE. COPIE */
17. CHAMA FECHA-ARQUIVO(END(BCES.E) ; /* FECHA TRAB */
18. CHAMA TERMINE ;  /* DESATIVA-SE */
```

3.4.4 - MÓDULO SUBMETA

O módulo 'SUBMETA' tem como objetivo executar os comandos contidos num 'arquivo lico'. Esses comandos são executados um por um, os quais têm sua execução com espera, o que implica na execução de um comando somente após o término da execução do comando anterior. O módulo 'SUBMETA' é carregado e

executado quando um comando 'SUBMETA' é teclado pelo usuário , ou quando, dentro de um arquivo LICO, um novo comando 'SUBMETA' é encontrado, implicando na carga e execução de um novo módulo.

3.4.4.1 - DEFINIÇÃO DOS PROCEDIMENTOS

Quando um comando 'SUBMETA' é tecla - do, o módulo MESTRE pede a carga e execução do módulo 'SIS:SIS.SUBMET.EXE', passando como parâmetro o nome do arquivo LICO a ser lido e o número do 'NO' onde seus comandos serão executados (opcional).

Inicialmente, o módulo testa a exis - tência do número do 'NO' no comando, para decidir se a execução dos comandos será no próprio computador, ou se pedirá uma transmissão para que os comandos do arquivo LICO sejam executados num outro computador. Se um 'NO' for mencionado, implica que o mesmo comando teclado pe- lo usuário deverá ser transmitido para o 'NO' em ques - tão, onde será entregue ao módulo MESTRE daquele 'NO' , e reiniciada sua execução. Se um 'NO' não for menciona do, segue a execução normal do módulo que inicialmente pede a abertura do arquivo LICO a ser lido.Após a aber- tura são lidos os registros cujo conteúdo é um comando a ser interpretado, sendo feito um pedido com espera ao módulo MESTRE para que esse comando seja interpretado e executado. Esses passos de 'leitura' e 'pedidos com es pera' são feitos até que todo arquivo LICO seja lido.No ta-se que quando um módulo MESTRE recebe um comando 'SUB META' cuja entrada foi feita num outro computador, to - das as referências são feitas para o computador 'ori -

gem', cujos 'defaults' foram passados junto com o co - mando. Dessa forma, somente a execução é feita nesse computador, sendo realizadas o restante das operações de abertura, fechamento, etc., no computador origem.

3.4.4.2 - ALGORITMO

As seguintes variáveis são utiliza - das:

BCES - BCES do arquivo LICO.

BUFFER - comando recebido.

ARQ - recebe o nome do arquivo.

NO - recebe o número do NÓ.

M - número do módulo MESTRE.

AUX - área auxiliar

```
/* MÓDULO SUBMETA */

1. I = 1.
2. ENQUANTO BUFFER(I) ≠ ' '
      FAÇA
         ARQ(I) = BUFFER(I) ;  /* PEGA NOME DO ARQUIVO */
         I = I + 1 ;
       FIMFA
3. ENQUANTO BUFFER(I) = ' ' & I <= 80
       FAÇA I = I + 1 ;
4. SE I = 80
      ENTÃO VA P/10 ;
5. NO = BUFFER(I) ;
6. AUX = 'SUB' ;
```

```
7. AUX(5) = ARQ ;
8. CHAMA TRANSMITA(NÓ,END(AUX)) ;  /* PEDE TRANSMISSÃO */
9. CHAMA TERMINE ;  /* DESATIVA-SE */

   /* LEITURA/PEDIDO */

10. CHAMA ABRA-ARQUIVO(ARQ,END(BCES)) ;  /* PEDE ABERTURA */
11. CHAMA LEIA(END(BCES)) ;  /* PEDE LEITURA */
12. ENQUANTO STATUS ≠ 'FIM-DE-ARQUIVO'
       FAÇA
          AUX = DADOS.BCES ;
          CHAMA DEPOSITE(M,END(AUX)) ;  /* ENTR. COM ESPERA */
          CHAMA LEIA(END(BCES)) ;  /* PEDE LEITURA */
        FIMFA
13. CHAMA TERMINE ;  /* DESATIVA-SE */
```

3.4.5 - MÓDULO DIRETÓRIO

O módulo DIRETÓRIO, 'DIRETO.EXE', tem como objetivo informar ao usuário a existência de um ou mais arquivos no seu diretório, além de informar todo seu conteúdo. Para que isso seja feito, esse módulo, ao receber as informações do usuário, dá um certo tratamento, para que sejam definidos os procedimentos que enviarão informações ao usuário a nível de arquivos, arquivos irmãos ou a nível de diretório.

3.4.5.1 - DEFINIÇÃO DOS PROCEDIMENTOS

A estrutura organizacional dos arquivos em disco foi definida de forma que existem níveis hierárquicos para que um determinado arquivo seja acessado.

No primeiro nível encontra-se um arquivo 'MESTRE' cujo conteúdo são todos os diretórios existentes naquele volume. No segundo nível encontram-se os arquivos 'DIRETÓRIOS' cujo conteúdo são todos os arquivos associados a um usuário. Por fim, no terceiro e último nível encontram-se os 'ARQUIVOS' cujo conteúdo pode ser um programa FONTE, OBJETO, ARQUIVOS DE DADOS , etc. Para cada arquivo contido nessa estrutura existe uma identificação para o sistema, independendo do nível em que ele se encontre. Com isso, um arquivo do tipo 'MESTRE', 'DIRETÓRIO' ou de dados, pode ser lido sendo acessado o seu conteúdo o qual está diretamente ligado com a função do arquivo dentro do seu nível.

Baseado nessa estrutura, o módulo 'DIRETO.EXE' trata as informações recebidas de forma que identifique o nível da estrutura a ser acessado e defina as operações referentes a tarefa solicitada.

Existem 4 tipos de informações que podem ser enviadas pelo usuário: nenhuma informação, arquivos irmãos por extensão, arquivos irmãos pelo nome e simplesmente um determinado arquivo. Uma outra forma seria 'TODOS OS DIRETÓRIOS EXISTENTES', onde somente o primeiro nível da estrutura seria acessado.

Quando nenhuma informação é recebida, o arquivo 'DIRETÓRIO' é acessado, sendo identificado pelo diretório (default) do usuário. Sua identificação é dada por "MESTRE.(DIRETÓRIO).SIS".

Quando arquivos 'IRMÃOS POR EXTENSÃO' são mencionados, o arquivo diretório é aberto e todos os

registros lidos sendo informados todos os arquivos com aquela extensão. O mesmo ocorre para arquivos 'IRMÃOS PELO NOME', sendo listados todos os arquivos com o nome mencionado. No caso da informação de entrada ser sim - plesmente um nome de arquivo, um pedido de abertura é feito, onde a devolução do 'STATUS' da operação identifica sua existência ou não, poupando assim o acesso e busca no arquivo 'DIRETÓRIO' em questão. Após a devolução do 'STATUS', o usuário é informado e, caso o arquivo exista, é feito um novo pedido para que o mesmo seja fechado.

3.4.5.2 - ALGORITMO

As seguintes variáveis são utilizadas:

CADEIA - Cadeia contendo o nome do arquivo.
BUFFER - Área de entrada do comando.
BCES - BCES a considerar.
*.REG - Campos do registro lido.
E1 - Endereço de desvio.
EXTARQ - Salva extensão do arquivo.
NOMARQ - Salva nome do arquivo.

```
/* MÓDULO DIRETÓRIO */

1. E1 = 3 ;
2. I = 80 ;
3. ENQUANTO BUFFER(I) = ' ' & I >= 1
      FAÇA I = I - 1 ;
4. SE I = 1
      ENTÃO FAÇA
```

```
            CADEIA(2) = 3 ;
            CADEIA(3) = '*.*' ;
            VA P/12 ;
          FIMFA
 5. J = I ;  /* SALVA I */
 6. K = 1 ;
 7. ENQUANTO BUFFER(K) = ' ' & K < J
       FAÇA K = K + 1 ;
 8. SE K = J
       ENTÃO FAÇA
             PEDE-IMPRESSÃO('NOME INVALIDO') ;
             CHAMA TERMINA ;
           FIMFA
 9. CADEIA(2) = J - K + 1 ;
10. L = 3 ;
11. ENQUANTO K <= J
       FAÇA
          CADEIA(L) = BUFFER(K) ;
          I = I + 1 ;
          K = K + 1 ;
        FIMFA
12. CHAMA SEPARARQ(END(BCES),END(CADEIA));  /* SEPARA NOME */
13. SE EXT.BCES = '*'
       ENTÃO SE NOME.BCES = '*'
             ENTÃO VÁ P/14 ;  /* IGUAL AO VAZIO */
             SENÃO FAÇA
                   NOMARQ = NOME.BCES ;
                   E1 = 22 ;  /* SÓ TESTA NOME */
                FIMFA
             SENÃO SE NOME.BCES = '*'
```

```
                    ENTÃO FAÇA
                          EXTARQ = EXT.BCES ;
                          E1 = 20 ;  /* SÓ TESTA EXTENSÃO */
                       FIMFA
                    SENÃO FAÇA
                          CHAMA ABRE-ARQUIVO(END(BCES),END(CADEIA) ;
                          CHAMA FECHA-ARQUIVO ;
                          CHAMA TERMINE ;
                       FIMFA
14. CADEIA = NO.BCES || ':' || VOLUME.BCES || '.'
             || DIRETÓRIO.BCES || '.MESTRE.SIS' ;
15. CADEIA(2) = 26 ;
16. CHAMA  ABRA-ARQUIVO(END(BCES),END(CADEIA)) ;

    /*  LEITURA E LISTAGEM */

17. CHAMA LEIA(END(BCES)) ;
18. SE 'FIM-DE-ARQUIVO'
       ENTÃO CHAMA TERMINE ;
19. VÁ P/ E1 ;
20. SE EXTARQ ≠ EXT.REG
       ENTÃO VÁ P/ 17 ;
21. VÁ P/ 23 ;
22. SE NOMARQ ≠ NOME.REG
       ENTÃO VÁ P/ 17 ;
23. PEDE-IMPRESSÃO( REG ) ;
24. VÁ P/ 17 ;
```

3.4.6 - MÓDULO SISTEMA

Quando uma tarefa é inicializada, a mesma fica

esperando sua execução de acordo com o estado do sistema. Durante sua execução a tarefa pode se encontrar em várias filas, as quais definem a natureza do estado de uma tarefa. O objetivo desse módulo é informar ao usuário o estado de uma ou mais tarefas dentro do sistema.

3.4.6.1 - DEFINIÇÃO DOS PROCEDIMENTOS

Quando esse módulo é carregado o mesmo recebe os parâmetros do comando num BUFFER, podendo ser o nome de uma tarefa ou nenhuma informação, o que implica que o estado de todas as tarefas deve ser informado.

Se uma tarefa for recebida, a subrotina do núcleo, 'estado', é chamada e o módulo fica esperando as informações que serão enviadas para o usuário. Caso o BUFFER de recebimento esteja vazio, é colocado o valor '-1' na variável PARÂMETRO DA SUBROTINA e a chamada é feita para que sejam devolvidas informações sobre o primeiro processo do sistema. Sempre que a subrotina é chamada, são devolvidas informações sobre a tarefa (no caso, o primeiro) seguida da identificação do próximo processo, a qual está contida no segundo parâmetro da subrotina. Com isso o módulo envia para o usuário o estado do primeiro processo e, em seguida, faz chamadas sucessivas para que sejam informados o estado dos próximos processos do sistema. Desta forma, o estado de todos os processos é informado até que a subrotina devolva a identificação do final deles.

3.4.6.2 - ALGORITMO

As seguintes variáveis são utiliza - das:

BUFFER - Contém o nome da tarefa.

P1,P2,P3 - Parâmetros da subrotina 'estado'.

ARECEB - Área de recebimento de informações.

```
   /* MÓDULO SISTEMA */

 1. P3 = END(ARECEB) ;
 2. I = 1 ;
 3. ENQUANTO BUFFER(I) = ' ' & I <= 80
          ENTÃO FAÇA I = I + 1 ;
 4. SE I > 80
       ENTÃO VÁ P/ 10 ;
 5. P1 = 0 ;
 6. P2 = END(BUFFER(I)) ;
 7. CHAMA ESTADO(P1,P2,P3) ;
 8. PEDE-IMPRESSÃO ;
 9. CHAMA TERMINE ;  /* DESATIVA-SE */

   /* INFORMA TODAS */

10. P1 = -1 ;
11. CHAMA ESTADO(P1,P2,P3) ;
12. BUFFER = ARECEB ;
13. P2 = END(BUFFER) ;
14. P1 = 0 ;
15. ENQUANTO COD.RETORNO ≠ 'FIM'
       FAÇA
```

```
            ARECEB = ' ' ;  /* LIMPA ÁREA */
            CHAMA ESTADO(P1,P2,P3) ;
            PEDE-IMPRESSÃO ;
            BUFFER = MEMÓRIA(P2) ;  /* PEGA MEM APON. P/ P2 */
         FIMFA
16. CHAMA TERMINE;  /* DESATIVA-SE */
```

∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿

CAPÍTULO IV

4. ANÁLISE ESTRUTURAL

Durante o desenvolvimento desse trabalho foi feito um levantamento das especificações de micros nacionais , sendo obtidos resultados que definem um conjunto de 'RECURSOS MÍNIMOS' disponíveis. Isto é, um conjunto de recursos que representa o mínimo que pode ser oferecido a um usuário em termos de software e hardware.

O seguinte quadro, contendo alguns micros nacionais, mostra alguns dos recursos oferecidos|[11]|:

SISTEMA	CPU	MEM	SIST. OPER.	LINGUAGENS
BRASCOM BR 1000	Z80A	64 A 128K	BR 1000 COMPAT. CP/M	BASIC, COBOL FORTRAN, PL/I
COBRA C 305	Z80A	64 K	SOM/E COMPAT. CP/M	COBOL,LTD,LPS FORTRAN, BASIC
ITAUTEC I 7010	8085	64 A 128K	SIM/M	BASIC
LABO 8221	Z80A	128K	SOL COMPAT. CP/M	BASIC
POLIMAX 201 DP	Z80A	64 K	SOP COMPAT. CP/M	BASIC, COBOL FORTRAN, PL/I
PROLOGICA SIST. 700	Z80A	64 K	DOS 700 COMPAT. CP/M	COBOL, BASIC FORTRAN,FATUROL
SID 3000	8085	64 K	DOS 3000 COMPAT. CP/M	COBOL, BASIC FORTRAN
EDISA ED 281	Z80A	64 A 208K	EDOS COMPAT. CP/M	COBOL, BASIC FORTRAN
SHUMEC M 100/85	8085	64 K	CP/M 2.2	COBOL, BASIC
POLIMAX MAXXI	6502	16 A 48 K	SO COMPAT. APPLE DOS	BASIC
DISMAC 8000	Z80	16 A 48 K	DOS COMPAT. TRS DOS	BASIC
KEMITRON NAJA	Z80	16 A 48 K	KDOS COMPAT. TRS DOS	BASIC
UNITRON AP-II	6502	48 K	UDOS COMPAT. APPLE DOS	BASIC

Nota-se que, em termos de hardware, não existe uma grande variedade de recursos oferecidos definindo, assim, uma configuração mínima, comum a todos eles. Com isso, um microcomputador cuja arquitetura se enquadre no atual mercado (CPU, Memória, Periféricos), faz com que o mesmo seja caracterizado primordialmente pelo software oferecido para sua área de aplicação |[11]|.

No que diz respeito ao software, deste também pode-se colher dados que resultem num mínimo oferecido. Das linguagens que aparecem no quadro, COBOL e BASIC mostram claramente sua presença em quase todos, restando o "SISTEMA OPERACIONAL" como o ítem fundamental de avaliação de um microcomputador, segundo o quadro apresentado.

Um SISTEMA OPERACIONAL oferecido por um micro é avaliado pelas facilidades que o mesmo fornece para a manipulação dos recursos disponíveis|[11]|.

Valendo-se do fato de que "um sistema só é bom quando é bom para o usuário", proporcionando-lhe a edição de textos, a manipulação de arquivos, o gerenciamento de tarefas, a disponibilidade do software básico, etc., o "SIRIUS" foi desenvolvido para que, além dessas ferramentas, que são oferecidas pelo software do projeto em desenvolvimento, o usuário disponha de um ambiente propício para interagir com o sistema|[15]|. Com o objetivo de fazer uma melhor exposição do que é oferecido, classificaremos o usuário em dois tipos, de acordo com a natureza dos dados que o mesmo manipula dentro do sistema. São eles: o usuário de desenvolvimento de software aplicativo e o usuário de desenvolvimento de software básico. Dentro do universo abrangido pelo "SIRIUS", a seguir são mostradas as ferramentas específicas para cada um desses dois tipos de usuário.

4.1 - DESENVOLVIMENTO DE SOFTWARE APLICATIVO

Para um usuário que desenvolve software aplicativo, o mesmo deverá dispor de ferramentas que englobem os seguintes pontos: edição de arquivos, disponibilidade de software básico, manipulação de arquivos e execução de programas|[13]|. Ob -

servando passo a passo uma sessão típica de um usuário, podemos ordenar os recursos oferecidos pelo SIRIUS, à medida que sua utilização se torne necessária. Assim, os 'MEIOS DE UTILIZAÇÃO' ficam no primeiro nível, seguido da 'LINGUAGEM DE COMANDOS' e, por fim, a filosofia de 'ARQUIVOS DE TRABALHO'.

- MEIOS DE UTILIZAÇÃO

No que diz respeito aos meios de utilização, daremos enfoque principal à divisão do vídeo em duas janelas, possibilitando ao usuário tratar com informações de diferentes natureza no mesmo vídeo. Isto é, na janela inferior encontram-se as informações tecladas pelo usuário, tidas como entradas pelo sistema. Na janela superior encontram-se as informações enviadas pelo sistema ao usuário, podendo estas serem comandos recebidos pelo sistema cujo tratamento resultou numa interpretação, informações tidas como mensagens do sistema, e informações de processos que utilizam o vídeo como um arquivo de saída. Nota-se que a utilização do vídeo pode ser feita por mais de um processo, onde as informações são enviadas para a janela que se encontre disponível naquele instante. Uma situação fácil de ocorrer seria a execução simultânea da listagens de um arquivo na janela superior e a entrada de comandos na janela inferior. Neste caso, as mensagens do sistema tidas como respostas aos comandos teclados seriam dirigidas para a janela inferior, quando um comando não estivesse sendo teclado.

- LINGUAGEM DE COMANDOS

Como já descrito no CAPÍTULO I, a linguagem de comandos é composta por comandos de arquivo, comandos de progra

ma e comandos de sistema. Baseado numa configuração mínima de software, comentada no início desse capítulo, esse sistema dispõe de comandos que proporcionam ao usuário a edição de arquivos, o software básico existente, a manipulação de arquivos|[1]| e a execução de programas. Nota-se que um sistema cujo conjunto de comandos satisfaça aos ítens acima mencionados, torna- se compatível com o quadro anteriormente mostrado, figurando junto com o SOM|[3]| e o CP/M oferecido pela maioria dos micros|[8, 12]|. Com isso, pode-se incluir esse sistema dentre aqueles que oferecem um conjunto de diretivas, proporcionando ao usuário a manipulação de arquivos e a execução de tarefas, contidas nos comandos de arquivo e de programas. Através dos comandos de sistema, o usuário tem acesso às informações gerais do sistema, além da suspensão de tarefas dentro do ambiente de multiprogramação oferecido pelo projeto. A disponibilidade do software básico dá-se através dos comandos definidos, onde são utilizadas as linguagens oferecidas pelo projeto juntamente com o editor de textos.

- ARQUIVOS DE TRABALHO

Esses arquivos foram introduzidos nesse sistema para atender às seguintes necessidades:

1. Otimizar o software/hardware, para responder aos casos normais, mesmo que isto cause prejuízo aos não corriqueiros.

2. Tirar o máximo da necessidade de conhecimento de computadores por parte do usuário, tornando transparente ao mesmo as funções de processamento interno.

3. Minimizar o software, fazendo com que as rotinas do sistema utilizem um arquivo padrão, onde serão feitas as transações.

4. Maior nível de segurança, pois o arquivo original não é alterado diretamente e, sim, o de trabalho.

Quando um usuário desejar alterar um programa , o arquivo FONTE deverá ser selecionado, cuja forma está definida na linguagem de comandos. Uma vez que o arquivo esteja selecionado, o próximo passo poderá ser a EDIÇÃO, onde não se dá o nome do arquivo.

Seguindo a EDIÇÃO, o próximo passo é a COMPILAÇÃO, que chamaremos de SINTAXE. Esse passo pode ser omitido , tornando-se transparente ao usuário a geração de um código intermediário, para posterior execução. Após a SINTAXE, segue-se a execução que, se solicitada pelo usuário sem o nome do arquivo, o arquivo de trabalho OBJETO é executado. Caso o arquivo OBJETO não exista, automaticamente é feita a SINTAXE e depois a EXECUÇÃO, o que define a transparência da compilação para o usuário.

- OPERAÇÃO NORMAL

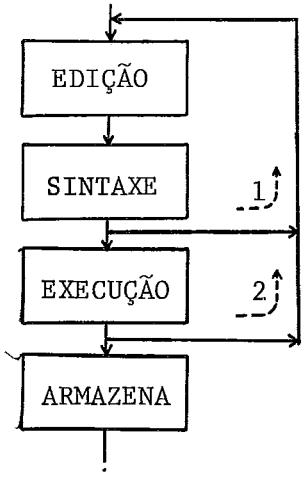


FIG. 14

Suponhamos que um usuário não necessite saber do passo COMPILAÇÃO. Desta forma, o usuário somente editará e executará a tarefa. Se após uma compilação com erro, o executor não executar o programa e der uma mensagem de erro, a seqüência de comandos teclados pelo usuário seria como mostra a figura a - baixo:

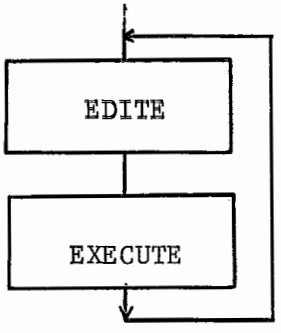


FIG. 15

Nota-se que esta seqüência de comandos se repete até que não haja mais erros de sintaxe, ficando o passo SINTAXE transparente, mas dando suas mensagens para o usuário. Quando um programa é executado, os erros de execução podem ser tirados via DEPURAÇÃO. Após o programa estar totalmente correto ou, no final de uma sessão do usuário, o mesmo poderá salvar seus arquivos, onde são recuperados o FONTE e o OBJETO, caso existam.

4.2 - DESENVOLVIMENTO DE SOFTWARE BÁSICO

Para um usuário que desenvolve SOFTWARE BÁSICO , isto é, desenvolve software em benefício do sistema, é necessário que o mesmo disponha de um ambiente que ofereça a possibilidade de alterações e inclusões de novas ferramentas. Dentro do ambiente abrangido pelo SIRIUS, temos como principal enfoque a inclusão de um novo comando na linguagem gerando, assim, a necessidade do do-

mínio do seu funcionamento. Com isso, são analisados dois principais pontos, cujo desenvolvimento baseou-se na facilidade de domínio do seu funcionamento. São eles: A modularidade do sistema e o tratamento dado aos comandos.

- MODULARIDADE

A definição de uma estrutura modular teve como principal objetivo a descentralização das funções desse sistema juntamente com a independência de execução das tarefas (paralelismo) |[5]|. Ou seja, desde o recebimento de informações do usuário até a devolução de uma resposta, a informação segue uma trajetória, de forma que a mesma é entregue a um módulo cujas funções representam uma uma nova etapa do seu processamento. Tomemos como exemplo o envio de uma diretiva pelo usuário, sendo esta recebida pelo MANIPULADOR, entregue ao módulo MESTRE e, posteriormente, uma resposta é enviada pelo IMPRESSOR. Podemos observar que as funções de recebimento, tratamento e resposta,são executadas dentro de cada módulo a que as mesmas pertencem. Desta forma, a alteração de uma dessas funções resultaria apenas na alteração de um determinado módulo.

Além da descentralização das funções, a modularidade proporciona a independência de tarefas, onde a entrega de informações a um determinado módulo não implica, obrigatoriamente, na espera dos resultados do seu tratamento. Ou seja, um módulo poderá enviar informações a um outro e prosseguir com seu funcionamento normal, a menos que a entrega seja feita com "ESPERA", utilizando-se opções oferecidas pelas funções do "NÚCLEO" |[2]|. Com essa independência de tarefas, após um comando ser recebido pelo Manipulador, este é entregue ao módulo MESTRE e o

usuário é liberado para teclar um novo comando.

- TRATAMENTO DOS COMANDOS

Como já descrito, a definição das funções do SIRIUS baseou-se em vários pontos, dentre eles a configuração do hardware. A rotina de tratamento do módulo MESTRE, rotina que interpreta os comandos, foi definida com o objetivo de ter uma utilização viável em qualquer configuração do sistema, principalmente podendo ser utilizada em toda a gama de memória, oferecida pelas prováveis expansões. Com isso, os seguintes fatores foram pontos-chaves para sua definição: Pouca utilização de memória, e o conjunto de funções que resultaram no grupo de comandos que compõem a linguagem. Esses dois fatores tornaram a "definição formal de uma linguagem" inviável, uma vez que o reduzido número de comandos, junto a possíveis alterações, facilitaria a utilização de uma tabela de comandos.

A tabela de comandos foi definida de forma que seu conteúdo dispusesse de informações suficientes para que os três passos que compõem a interpretação fossem executados. São eles: A avaliação do comando, a montagem dos parâmetros e a formatação e chamadas de subrotinas do núcleo. A estrutura da tabela, junto com a quantidade de comandos, possibilitaram a utilização de um algoritmo de busca seqüêncial, no passo AVALIAÇÃO, aplicado para esse tipo de tabela, dispensando a utilização de algoritmos mais complexos, o que acarretaria numa maior utilização de memória. Nota-se que a inclusão de um novo comando acarretará na alteração da tabela, onde as informações atualizadas estão diretamente ligadas à natureza do novo comando, podendo este ser sinônimo ou gerador.

Em resumo, todas as fases do desenvolvimento desse sistema, teve como principal objetivo o "USUÁRIO", para que o mesmo, em seus diversos níveis de utilização, tivesse acesso às ferramentas utilizadas e disponíveis pelo sistema.

∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿∿

5. C O N C L U S Ã O

Diante das exposições feitas no capítulo anterior, "ANÁLISE ESTRUTURAL", onde são colocadas as ferramentas oferecidas pelo SIRIUS para cada ítem de avaliação, nota-se seu enquadramento em todo um ambiente atualmente representado pelos recursos dos diversos microcomputadores existentes.

É claro que o referido "enquadramento" diz respeito apenas ao universo abrangido pelo SIRIUS, onde uma avaliação do projeto como um todo, descrito na introdução deste trabalho , seria prematura trazendo sérias desvantagens, como por exemplo o grande número de programas aplicativos existentes no atual mercado.

Como última etapa de desenvolvimento deste trabalho, o SIRIUS foi implementado num microcomputador, TRS 80 da RADIO SHACK, cujas especificações de memória e CPU se assemelham ao projeto em questão.

A seguir são feitas algumas sugestões para a continuação do desenvolvimento do SIRIUS, resultando numa nova versão para o projeto em desenvolvimento.

- ARQUIVOS DE TRABALHO LOCAIS À TAREFA

  Neste caso poderia existir mais de um arquivo de trabalho referente a um mesmo usuário. Com isso mais de uma tarefa de compilação, depuração, etc., poderia ser executada,

sendo o arquivo de trabalho identificado com referência à tarefa executada.

- ANÁLISE SINTÁTICA NO MANIPULADOR

Neste caso informações seriam analisadas pelo manipulador evitando o envio de informações, possivelmente incorretas, para serem interpretadas pelo MESTRE, acarretando evidente envio de mensagens de erro. As informações a serem testadas seriam do tipo - tamanho do nome do arquivo, nome de arquivo alfanumérico, nome de comando alfabético, etc.

- PASSAGEM DE PARÂMETROS PARA ARQUIVOS DE COMANDOS

Para cada submissão de arquivo, uma cópia do módulo executável 'SIS.SUBMET.EXE' é carregado e executado. Seguindo esta sugestão, os parâmetros seriam recebidos por esse módulo e seriam montados de acordo com informações contidas no arquivo de comandos em questão.

Por fim, todos os recursos oferecidos pelo SIRIUS foram definidos de acordo com a estrutura do projeto como um todo, acarretando que qualquer sugestão dada daqui por diante, figure como "mais um recurso" a ser oferecido pelo mesmo.

REFERÊNCIAS BIBLIOGRÁFICAS

1. Amaral, Henrique M. C. do - Subsistema de Gerência e Manipulação de E/S em Veículos de Acesso Direto para um Microcomputador. Tese de Mestrado, COPPE-UFRJ, Abr/82.

2. Breitiger, José Lavaquial - Um Núcleo Multiprogramável para Microcomputadores. Tese de Mestrado, COPPE-UFRJ,1983. (A ser defendida).

3. COBRA - Manual de Referência do Sistema Operacional Monoprogramável - SOM - COBRA 300/TD. Maio/1980.

4. Hansen, Brinch - Operating Systems Principles, Prentice Hall, 1973.

5. Hansen, Brinch - The Architecture of Concurrent Programs, Prentice Hall, 1978.

6. Holt, H. C. e Outros - Structured Concurrent Programming With Operations System Application. Massachusetts.Addison Wesley Pub. Co. - 1978.

7. Howard, John H. - Proving Monitors. Comm. of ACM 19(5), May/76.

8. Kirner, Claudio - Computação Distribuída: Proposta de um Sistema com Facilidades de Manipulação de Arquivos Dispersos numa Rede. Exame de qualificação para doutorado em Engenharia de sistemas (não publicado). COPPE-UFRJ.

9. Lampson, Butter W. e Redell, David D. - Experience with Processes and Monitors in MESA. Comm. of ACM 23(2). Fev/80.

10. Laventhal, Lance A. - 6502 Assembly Language Programming Osborn/McGrawhill, 1979.

11. Micro Mundo - Jornal do Usuário de Microcomputadores , out/82.

12. Nasajon, Cláudio - Uma Introdução ao CP/M - Micro Sistemas, set/82, 18-20.

13. Redell, David D. - PILOT: An Operating System for a Personal Computer. Comm of ACM 23(2), Fev/1980.

14. Ritchie, Denis M. e Thompson, Ken - The Unix Time-Sharing System. Comm. of ACM 17(7). Jul/74

15. Rotkowski, Chris - HASCI: An Introduction to the Human Aplications Standard Computer Interface. BYTE, out/82.

16. Seanlon, Leo J. - 6502 Software Design. Howard W. Sams Co. Inc., 1980.

17. Souza, Paulo Cesar - Um Modelo Sistêmico para Análise de Microcomputadores. Tese de Mestrado, IME, RJ, 1983 (a ser publicado).

18. Warren, Carl D. - The MC6809 Cook Book. TAB Books INC. 1980.

19. Wolochow, Peter I. - Microcomputer Operating System Trends Digital Design, 11(12):51-56, Dec/81.