UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIA E TECNOLOGIA

CURSO DE MESTRADO EM INFORMÁTICA

Um Modelo para Implementação de um Serviço e Protocolo de Transferência de Arquivos em Rede

Clizenit Pinheiro Assis de Lima

Campina Grande, PB

Maio/1990

Um Modelo para Implementação de um Serviço e Protocolo de Transferência de Arquivos em Rede

Clizenit Pinheiro Assis de Lima

Dissertação apresentada ao Curso de Mestrado em Informática da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do grau de Mestre.

Joberto S. B. Martins

Orientador

Campina Grande, PB

Maio/1990.

L732m Lima, Clizenit Pinheiro Assis
Um modelo para implementacao de um servico e protocolo de transferencia de arquivos em rede / Clizenit Pinheiro Assis de Lima. - Campina Grande, 1990.
210 f. : il.

Dissertacao (Mestrado em em Informatica) - Universidade Federal de Campina Grande, Centro de Ciencia e Tecnologia

1. Transparencia de Arquivos em Rede 2. Protocolo de Tranferencia de Arquivo 3. Dissertacao I. Martins, Joberto Sergio Barbosa, Dr. II. Universidade Federal de Campina Grande - Campina Grande(PB) III. Título

CDU 004.772(043)

UM MODELO PARA IMPLEMENTAÇAO DE UM SERVIÇO E PROTOCOLO
DE TRANSFERENCIA DE ARQUIVOS EM REDE

CLIZENIT PINHEIRO ASSIS DE LIMA

DISSERTAÇAO APROVADA EM 04.05.1990

JOBERTO SERGIO BARBOSA MARTINS
Orientador

DECIO FONSECA.
Componente da Banca

WANDERLEY LOPES DE SOUZA
Componente da Banca

ANTONIO CONCEIÇAO SOARES
Componente da Banca

Campina Grande, 04 de maio de 1990

À

Carlos, Rafael e Mateus

pela paciência, carinho

e compreensão.

Agradeço à Caixa Econômica Federal a oportunidade proporcionada para a realização do mestrado e o suporte oferecido para execução deste trabalho. Quero destacar a colaboração dos colegas Júlio Ernesto, Roberto Flávio, Paulo Roberto, Flávia Augusta e Eunice Mesquita.

Sou muito grata ao professor Joberto S. B. Martins pela orientação dedicada que prestou durante a elaboração deste trabalho.

Agradeço também a toda equipe da empresa INFOCON pela presteza e atenção dispensadas sempre que necessitava da sua colaboração.

Um agradecimento especial a todos os colegas da UFPB/Campina Grande pela amizade compartilhada. Em particular, um grato abraço a Rosangela Miná, José Neuman, Joaquim Celestino e Bruno.

Este trabalho só foi possível graças à ajuda valiosa de Carlos, meu marido. O meu agradecimento sincero pelo seu esforço, paciência e constante apoio durante todo o mestrado.

Este trabalho modela e implementa a infra estrutura necessária (processos, estruturas de dados, rotinas básicas,...) para suportar o serviço e protocolo de transferência, acesso e gerenciamento de arquivo.

O modelo de implementação é desenvolvido para o padrão FTAM ("File Transfer, Access and Management") no ambiente do sistema operacional UNIX. Os processos envolvidos, as estruturas de dados e as rotinas básicas são apresentadas juntamente com uma discussão sobre a flexibilidade e otimização decorrentes das opções adotadas. Uma comparação com um modelo alternativo de implementação é igualmente apresentada.

In this work is presented a model and the implementation of the basic structure (process, data structures and routines) for supporting the service and protocol for file transfer, access and management.

The model is developed for the FTAM standard in a Unix enviroment. The process, data structures and basic routines are presented together with a discussion about the flexibility and optimization of the options adopted. A comparison with an alternative implementation model is also presented.

ÍNDICE

Capítulo 1

INTRODUÇÃO

Os sistemas distribuídos têm crescido em importância e utilização sob a influência de vários fatores, dentre os quais a evolução de tecnologias computacionais de hardware e software e a utilização de mecanismos de cooperação entre processos. A evolução tecnológica vem propiciando uma multiplicidade de recursos a custos acessíveis, enquanto que os mecanismos de cooperação entre processos têm sido usados para aumentar a eficiência, a confiabilidade e a disponibilidade de sistemas [1].

De uma maneira geral, essa evolução leva a uma maior demanda de compartilhamento da informação pelos usuários nos sistemas de comunicação. Logo, aplicativos do tipo transferência de mensagens, compartilhamento de arquivos, distribuição de tarefas são ferramentas importantes para os usuários.

Dentre essas ferramentas destacam-se os serviços de transferência, acesso e gerenciamento de arquivos, motivo do trabalho em questão. Tais serviços definem como deve ser a transferência, o acesso e a gerência de arquivos armazenados ou trocados pelos sistemas de uma rede de computadores; permitem adicionar ou remover dados de uma facilidade de arquivamento e descrever os dados armazenados sem a necessidade de informações sobre detalhes de implementação e manutenção do sistema de arquivos [2].

O principal problema que surge, quando se decide implementar os serviços de transferência, acesso e gerenciamento de arquivos numa rede interligando sistemas computacionais heterogêneos, está relacionado com as características de cada sistema de arquivo. Observa-se que, de um sistema operacional para outro, existem diferenças entre os modos em que os dados de um arquivo são armazenados, bem como entre as maneiras de identificá-lo e entre os tipos de operações que podem ser efetuadas sobre o mesmo [3].

Além dos problemas relacionados com o sistema de arquivo, outras incompatibilidades podem aparecer devido à grande variedade de sistemas computacionais com capacidades de armazenamento e processamento diferenciadas.

Por tudo isso, acredita-se que, para evitar a proliferação de soluções particulares dos serviços de transferência, acesso e gerenciamento de arquivos, somente uma padronização garantirá uma estabilidade para os usuários.

Por ser o Modelo de Referência para Interconexão de Sistemas Abertos (RM-OSI/ISO) [4] uma tendência internacional, endossada a nível nacional pela Política Nacional de Informática, optou-se por implementar um serviço/protocolo de transferência de arquivo segundo a padronização FTAM ("File Transfer Access and Management) [5], [6], [7], [8] do referido modelo.

Por outro lado, sendo o sistema operacional UNIX [9] um padrão internacional numa vasta gama de equipamentos [10] e um sistema que oferece um poderoso ambiente para desenvolvimento de software [11], resolveu-se implementar o FTAM numa máquina rodando um sistema operacional compatível com o UNIX - System V. Esse sistema oferece recursos de comunicação interprocessos tais como filas de mensagens e memória compartilhada, que foram utilizadas na implementação. Oferece ainda facilidades de gerência de memória e escalonamento de processos, importantes na implementação do FTAM.

O sistema é implementado na linguagem "C" [12], não somente pela sua estreita relação com o sistema UNIX, mas principalmente por ser uma linguagem que permite implementações portáveis e que oferece recursos poderosos necessários a esta aplicação.

Neste escopo, o objetivo deste trabalho é modelar e implementar a infra-estrutura necessária para suportar um pacote FTAM completo, sem as "simplificações" que são algumas vezes feitas para acomodar a ausência de recursos de determinados sistemas operacionais.

Esse ambiente é estruturado de tal forma que as classes funcionais do FTAM podem ser implementadas gradativamente, de forma relativamente simples e sem modificações nas classes já desenvolvidas e instaladas.

Os capítulos deste trabalho estão organizados da seguinte forma:

. CAPíTULO 2: apresenta o serviço e o protocolo de transferência, acesso e gerenciamento de arquivo especificado pela ISO - o FTAM

. CAPíTULO 3: descreve os recursos de criação de processos e comunicação interprocessos do sistema operacional UNIX, utilizados na implementação.

. CAPíTULO 4: apresenta o modelo funcional da implementação, descreve os processos envolvidos, as estruturas de dados e tece considerações sobre: alocação de processos, comunicação interprocessos, interfaces com camadas adjacentes, sincronização e flexibilidade da implantação.

. CAPíTULO 5: faz um estudo comparativo entre o modelo de implementação apresentado no capítulo 4 e um modelo alternativo de implementação de um sistema de comunicação OSI.

. CAPíTULO 6: descreve os testes de operacionalidade da implementação.

. CAPíTULO 7: faz uma análise do modelo e da implementação, tecendo considerações sobre as características positivas e as desvantagens encontradas.

No anexo A descreve-se um exemplo da implementação do estabelecimento de conexão para o protocolo FTAM.

Nos anexos B e C apresenta-se, a título de ilustração, parte do código fonte dos processos envolvidos na implementação, bem como as estruturas de dados das tabelas utilizadas, das primitivas e UDPs do protocolo FTAM.

Capítulo 2

SERVIÇO E PROTOCOLO PARA TRANSFERÊNCIA, ACESSO E GERENCIAMENTO DE ARQUIVO DA ISO

2.1 INTRODUÇÃO

O objetivo deste capítulo é caracterizar os serviços de transferência, acesso e gerenciamento de arquivo especificados pela ISO.

Nele, faz-se primeiramente uma abordagem da camada de aplicação para posicionar os serviços de transferência, acesso e gerenciamento de arquivos dentro da arquitetura do Modelo de Referência da ISO. Em seguida, apresenta-se o Sistema de Arquivo Virtual que define todos os conceitos e aspectos para a descrição dos arquivos objetos dos serviços FTAM ("File Transfer, Access and Management").

As características dos serviços de transferência de arquivos são mostradas no ítem 2.3 deste capítulo e, finalmente, define-se o protocolo que suporta tais serviços.

2.2 A CAMADA DE APLICAÇÃO E O FTAM

O FTAM ("File Transfer, Access and Management") é um padrão ISO que define os serviços e o protocolo de transferência, acesso e gerenciamento de arquivo.

Na arquitetura do Modelo de Referência para Interconexão de Sistemas Abertos da ISO (RM-OSI/ISO), o serviço de FTAM está encaixado na camada de aplicação. Por isso, um breve relato dessa camada faz-se necessário para que se possa entender o contexto onde estão inseridos os serviços e o protocolo de transferência de arquivo.

O objetivo da camada de aplicação é permitir que processos de aplicação distribuída, que residem em sistemas distintos, se comuniquem, utilizando-se dos recursos oferecidos pelo ambiente OSI.

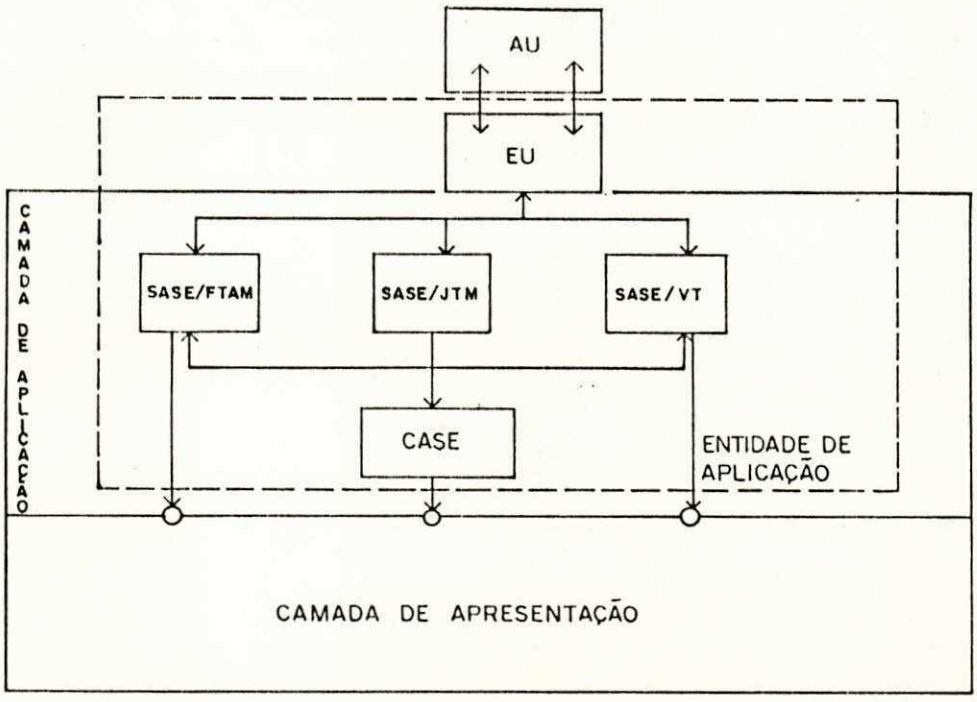
Em cada sistema aberto, a camada de aplicação é constituída por entidades de aplicação. Cada entidade de aplicação é constituída por elementos funcionalmente distintos. Podem-se identificar dois tipos de elementos : os ELEMENTOS DE SERVIÇO DE APLICAÇÃO e o ELEMENTO DE USUÁRIO. Uma entidade de aplicação é composta de um único elemento de usuário associado a um conjunto de elementos de aplicação.

Os Elementos de Serviço de Aplicação estão classificados em:

. Elementos de Serviço de Aplicação Comum ("Common Aplication Service Elements - CASE"): oferecem serviços básicos que independem da natureza da aplicação. Como exemplo, tem-se: estabelecimento e liberação de uma associação entre entidades de aplicação.

. Elementos de Serviço de Aplicação Específico ("Specific Application Service Elements - SASE"): oferecem serviços exclusivos para cada aplicação particular. Como exemplos, tem-se: transferência, acesso e gerenciamento de arquivos; terminal virtual e submissão remota de tarefas.

A figura 2.1 apresenta a estruturação da Camada de Aplicação do RM-OSI/ISO.



AU = Aplicação do Usuário

EU = Elemento de Usuário

SASE = ("Specific Application Service Element")

CASE = ("Common Application Service Element")

FTAM = ("File Transfer Access and Management")

JTM = ("Job Transfer and Manipulation")

VT = ("Virtual Terminal")

Fig. 2.1 : Estruturação da Camada de Aplicação

O Elemento de Usuário (EU) corresponde à parte do processo de aplicação que faz o mapeamento necessário entre as primitivas de serviço que devem ser fornecidas ao usuário da aplicação e as primitivas que são definidas para uso dentro da entidade de aplicação. Portanto, o Elemento de Usuário faz o mapeamento entre as primitivas de serviço relacionadas com o ambiente OSI e as primitivas correspondentes relacionadas com o ambiente real.

A figura 2.2 mostra o esquema de mapeamento de primitivas feito pelo Elemento de Usuário.

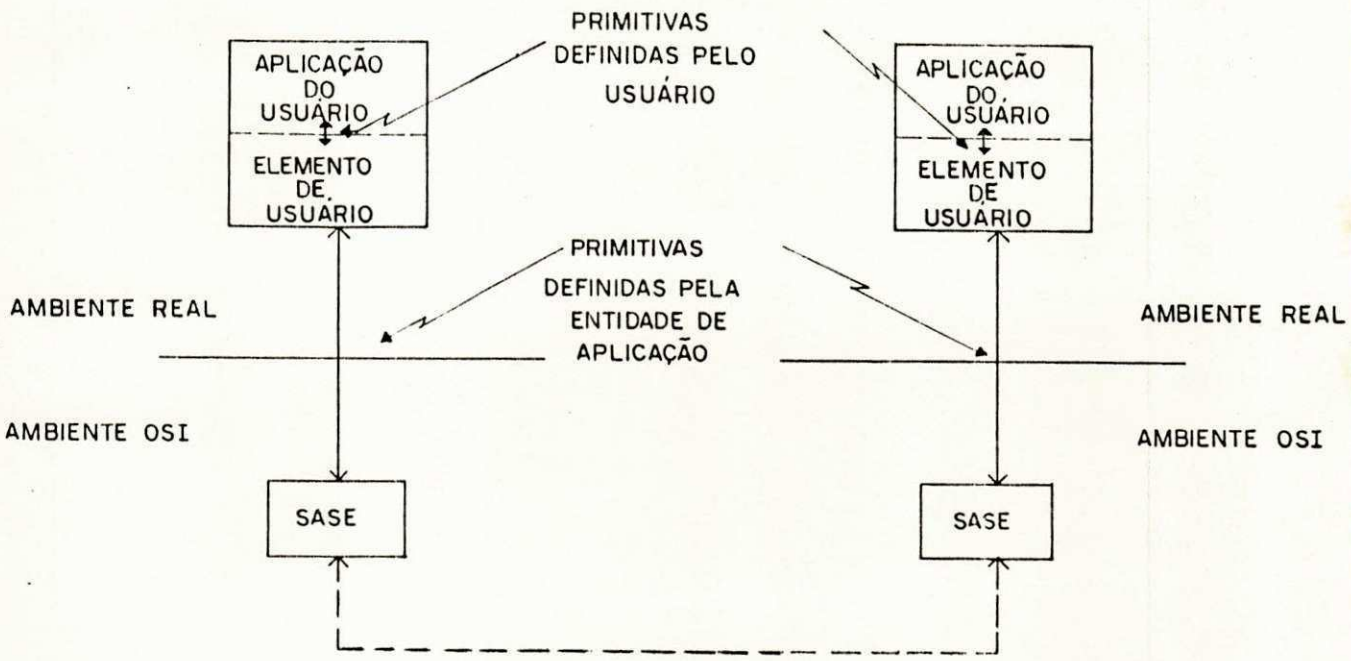


Fig.: 2.2 Mapeamento de Funções do Elemento de Usuário

A especificação do elemento de serviço para transferência e manipulação de arquivo que descreve-se a seguir, está baseada nas normas da ISO - DIS 8571/1/2/3/4 (FTAM).

2.3 ELEMENTO DE SERVIÇO DE APLICAÇÃO ESPECÍFICO PARA TRANSFERÊNCIA, ACESSO E GERENCIAMENTO DE ARQUIVO: SASE/FTAM

O objetivo principal dos serviços FTAM é permitir que sistemas heterogêneos realizem operações em arquivos de forma eficaz [13].

Para atingir tal objetivo, o serviço FTAM faz uso de um "arquivo virtual" cujo formato é independente da máquina hospedeira. Na transferência de um arquivo entre dois sistemas, o sistema fonte primeiro mapeia os dados arquivados para o formato do arquivo virtual; o sistema destino mapeia, então, o arquivo virtual recebido para o seu formato local.

A transformação de arquivos não é da alçada do serviço FTAM, mas sim dos seus sistemas usuários que devem implementar funções de mapeamento que absorvam as diferenças de representação e especificação de arquivos entre o sistema de arquivo virtual e o real. A figura 2.3 mostra o mapeamento entre o sistema de arquivo real e o ambiente OSI.

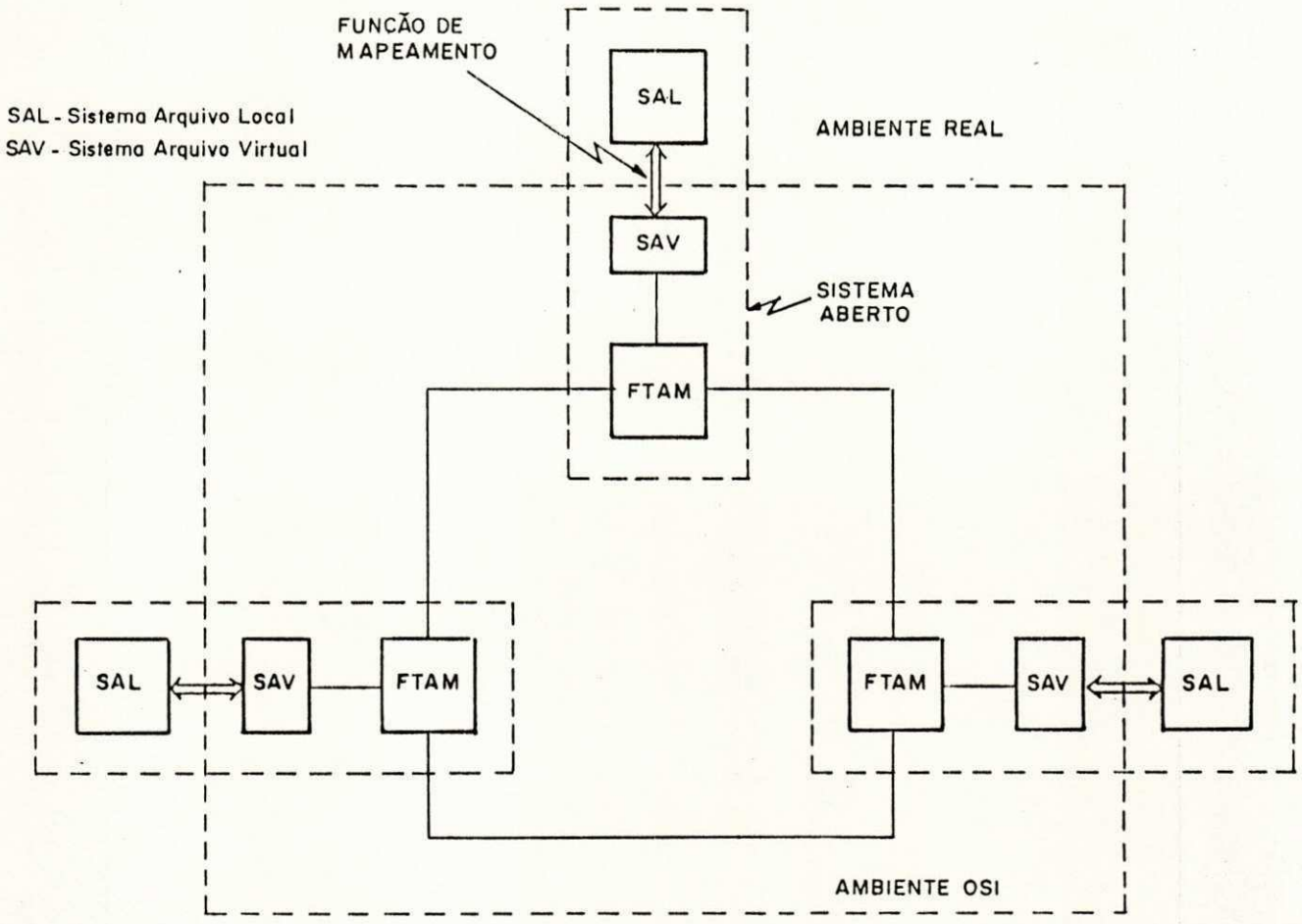


Fig. 2.3: Arquitetura de Sistemas Abertos com Serviços de Transferências, Acesso e Gerenciamento de Arquivos

O formato do arquivo virtual é definido pelo serviço FTAM através de um modelo comum de arquivos chamado Sistema de Arquivos Virtual (SAV).

2.3.1 SISTEMA DE ARQUIVOS VIRTUAL - SAV

A utilização de um Sistema de Arquivos Virtual, como um modelo padrão para descrever arquivos, permite a interconexão de sistemas de diferentes níveis de complexidade e que possuem diferentes maneiras de armazenar e acessar os dados de um arquivo qualquer [6]. O SAV define tanto a estrutura e os atributos associados aos arquivos, como também as operações válidas nos mesmos.

2.3.1.1 ESTRUTURA DE ARQUIVO

A estrutura de um arquivo define a organização das unidades de dados nele contidas e a relação existente entre elas. Neste modelo, os arquivos são considerados como tendo uma estrutura hierárquica.

Esta estrutura em árvore consiste de um nó raiz com nodos internos e folhas conectadas por arcos direcionados. Um nodo pertence a um único nível. Cada nodo dá acesso a sua subárvore, que é conhecida como FADU (File Access Data Unit). O conteúdo de um arquivo no SAV está armazenado em uma ou mais DUs ("Data Units"). O acesso a cada nodo dentro da FADU é feito na ordem: R, A, B, C, D, E, F. A figura 2.4 ilustra esses conceitos.

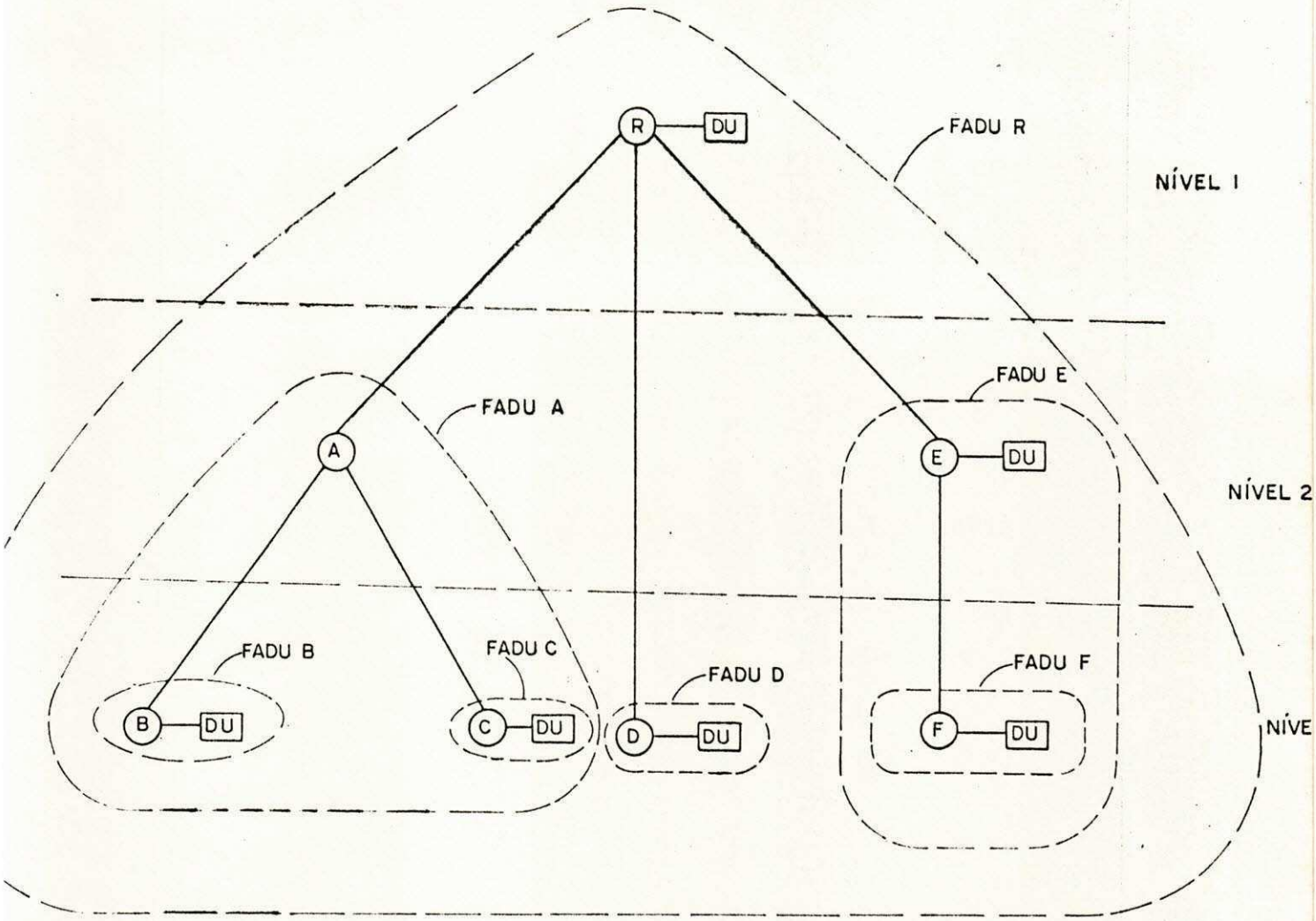


Fig. 2.4 : Estrutura de Acesso a Arquivo

2.3.1.2 ATRIBUTOS

Os atributos são as propriedades associadas ao arquivo e prestam-se para a sua gerência. São definidas duas classes de atributos:

- atributos de arquivo ou globais e
- atributos de atividade.

a) Atributos de Arquivo ou Globais:

Esses atributos traduzem as características que identificam o arquivo em questão e, num dado instante, possuem o mesmo valor para todo o usuário que deseja acessar este arquivo. Como exemplos, tem-se: o nome do arquivo, sua data de criação, a identidade do criador, etc....

b) Atributos de Atividade:

Esses atributos descrevem a relação existente entre o usuário e o arquivo em questão, definindo o estado das atividades em progresso. Como exemplos, tem-se: ação requisitada corrente, senhas de acesso corrente e identidade do iniciador corrente.

Com a finalidade de especificar os níveis que uma implementação pode suportar, esses atributos estão agrupados em três subconjuntos:

- Subconjunto Núcleo: corresponde ao conjunto de atributos que deve obrigatoriamente ser suportado pelo sistema de arquivos.

- Subconjunto de Armazenamento: agrupa atributos que estão relacionados com o controle do armazenamento físico do arquivo.

- Subconjunto de Segurança: corresponde às propriedades associadas ao controle de acesso aos arquivos. Esse subconjunto só pode ser suportado se o subconjunto de armazenamento tiver sido especificado.

A tabela 2.1 mostra as Classes e Subconjuntos dos atributos definidos para o Sistema de Arquivos Virtual.

SUBCONJUNTOS / CLASSES	NÚCLEO	ARMAZENAMENTO	SEGURANÇA
Atributos de Arquivo ou Globais	. Nome do Arquivo . Tipo de Dados . Data da última modificação . Data da última leitura . Data da última modificação de atributo . Identificação do Criador . Identificação último modificador . Identificação último leitor . Identificação último modificador de atributo . Disponibilidade do arquivo . Tipos de Operações . Tamanho do Arquivo . Tam. futuro do arq.	. Tamanho do Arquivo . Data da sua criação . Qualificações Legais	. Controle de Acesso . Algoritmo de Criptografia
Atributos de Atividade	. Tipo de Dados ativos . Pedido de Acesso . Localização corrente . Modo de Processamento . Entidades de Aplicação Correntes	. Conta Corrente . Contexto de Acesso . Controle de Concorrência . Senhas de Acesso	. Qualificacao legal ativa . Identificação do iniciador

Tab. 2.1: Classes e Subconjuntos dos Atributos definidos para o Sistema de Arquivo Virtual (SAV) da ISO.

2.3.1.3 OPERAÇÕES NO ARQUIVO

O modelo de Sistema de Arquivos Virtual define duas classes de operações sobre arquivos: operações num arquivo por inteiro e operações no conteúdo do arquivo. As tabelas 2.2 e 2.3 mostram essas operações com as respectivas funções.

OPERAÇÕES EM ARQUIVO INTEIRO	FUNÇÃO
Criação de Arquivo	Cria novo arquivo e define seus atributos.
Seleção de Arquivo	Verifica se o arquivo especificado existe e estabelece relacionamento entre iniciador e o arquivo selecionado.
Mudança de Atributos	Modifica os valores dos atributos de um determinado arquivo.
Leitura de Atributos	Obtém os valores dos atributos de um determinado arquivo.
Abertura de Arquivo	Abre arquivo para que os dados do mesmo possam ser acessados.
Fechamento de Arquivo	Fecha arquivo e seus dados não podem mais ser acessados.
Remoção de Arquivo	Remove o arquivo do SAV.
"Desseleção" de Arquivo	Desfaz o relacionamento existente entre o iniciador e o arquivo selecionado.

Tab. 2.2: Operações que podem ser efetuadas no arquivo inteiro

OPERAÇÕES NO CONTEÚDO DO ARQUIVO	FUNÇÃO
Busca	Localiza no arquivo uma FADU que passa a ser a FADU corrente.
Leitura	Lê a FADU corrente.
Inserção	Cria nova FADU e a insere no arquivo
Substituição	Substitui o conteúdo da FADU corrente. Após esta operação, a FADU corrente é a seguinte.
Extensão	Adiciona dados no fim da DU associada ao nó raiz da FADU corrente.
Remoção	Remove a FADU corrente. A próxima FADU passa a ser a FADU corrente

Tab. 2.3: Operações que podem ser efetuadas no conteúdo do arquivo

2.3.2 SERVIÇOS DE TRANSFERÊNCIA, ACESSO E GERENCIAMENTO DE ARQUIVOS

Na descrição dos serviços FTAM, a entidade que inicia e controla a comunicação é chamada **Entidade Iniciadora** e a entidade de aplicação parceira, à qual está associado o Sistema de Arquivo Virtual remoto é chamada **Entidade Respondedora**.

O serviço de arquivo e o protocolo suportando este serviço criam, numa série de etapas, um ambiente de trabalho no qual as atividades solicitadas pelo iniciador podem ser realizadas. O período de tempo no qual uma parte do estado comum mantido pelos usuários do serviço, permanece válido é chamado regime. O período durante o qual um conjunto de mensagens tem um propósito particular, tais como estabelecer, usar ou liberar um regime, é chamado fase.

Os seguintes regimes são definidos:

a) Regime de Conexão FTAM: período que existe enquanto uma conexão de aplicação é usada para o protocolo FTAM;

b) Regime de Seleção de Arquivo: período durante o qual um arquivo particular é associado à conexão estabelecida no regime anterior;

c) Regime de Abertura de Arquivo: período onde os dados do arquivo selecionado podem ser inspecionados ou manipulados;

d) Regime de Transferência de Dados: período onde acontece a transferência de arquivos propriamente dita.

A conclusão de um regime implica no término de todos os outros regimes que o sucederam. Para cada regime especificado, foram definidas fases que ocorrem durante o uso do serviço de arquivo. O aninhamento dos regimes e suas respectivas fases são mostrados na figura 2.5.

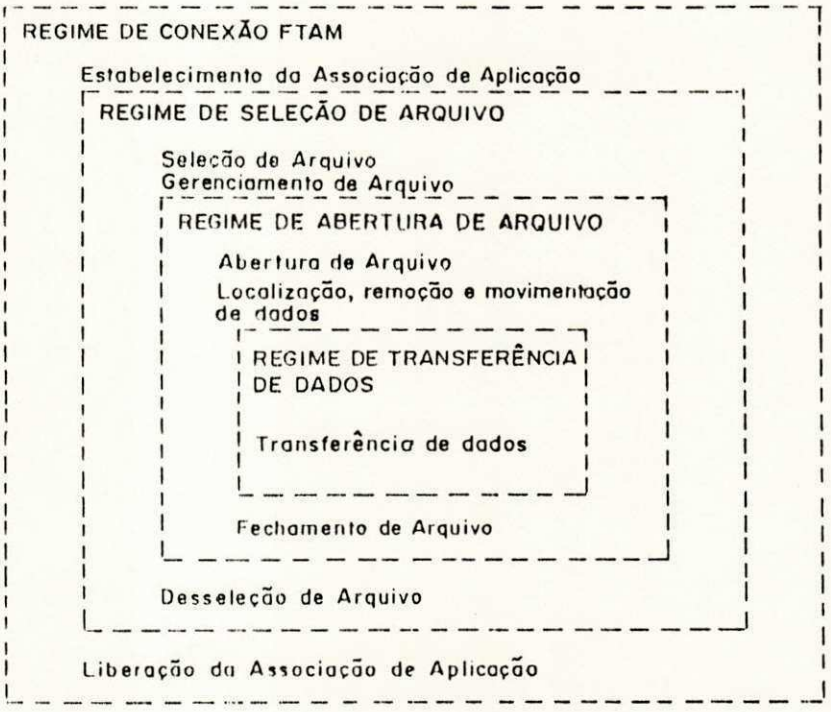


Fig. 2.5 - Regimes e Fases do Serviço de Arquivo

Associado com cada regime está definido um conjunto de primitivas conforme mostrado na figura 2.6 e no diagrama simplificado de estado/evento da figura 2.7.

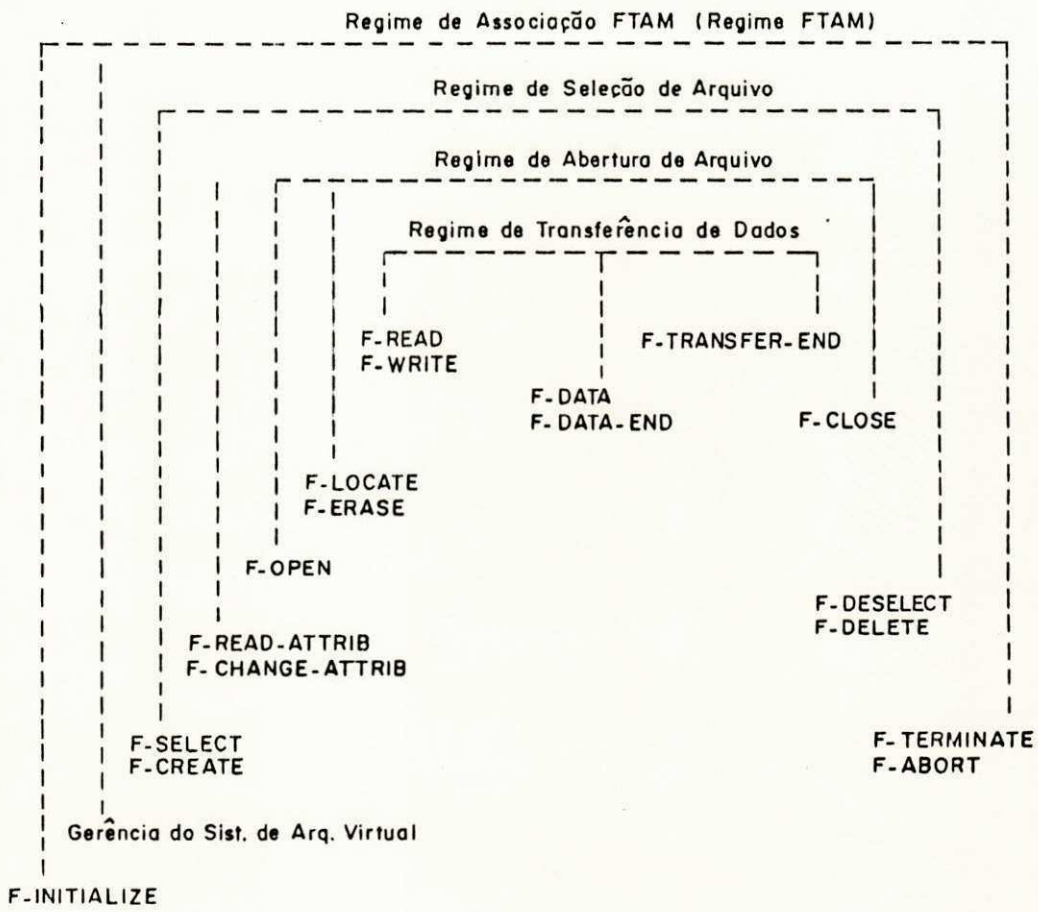


Fig. : 2.6 - Regimes do Serviço de Arquivo e Primitivas de Serviço envolvidas

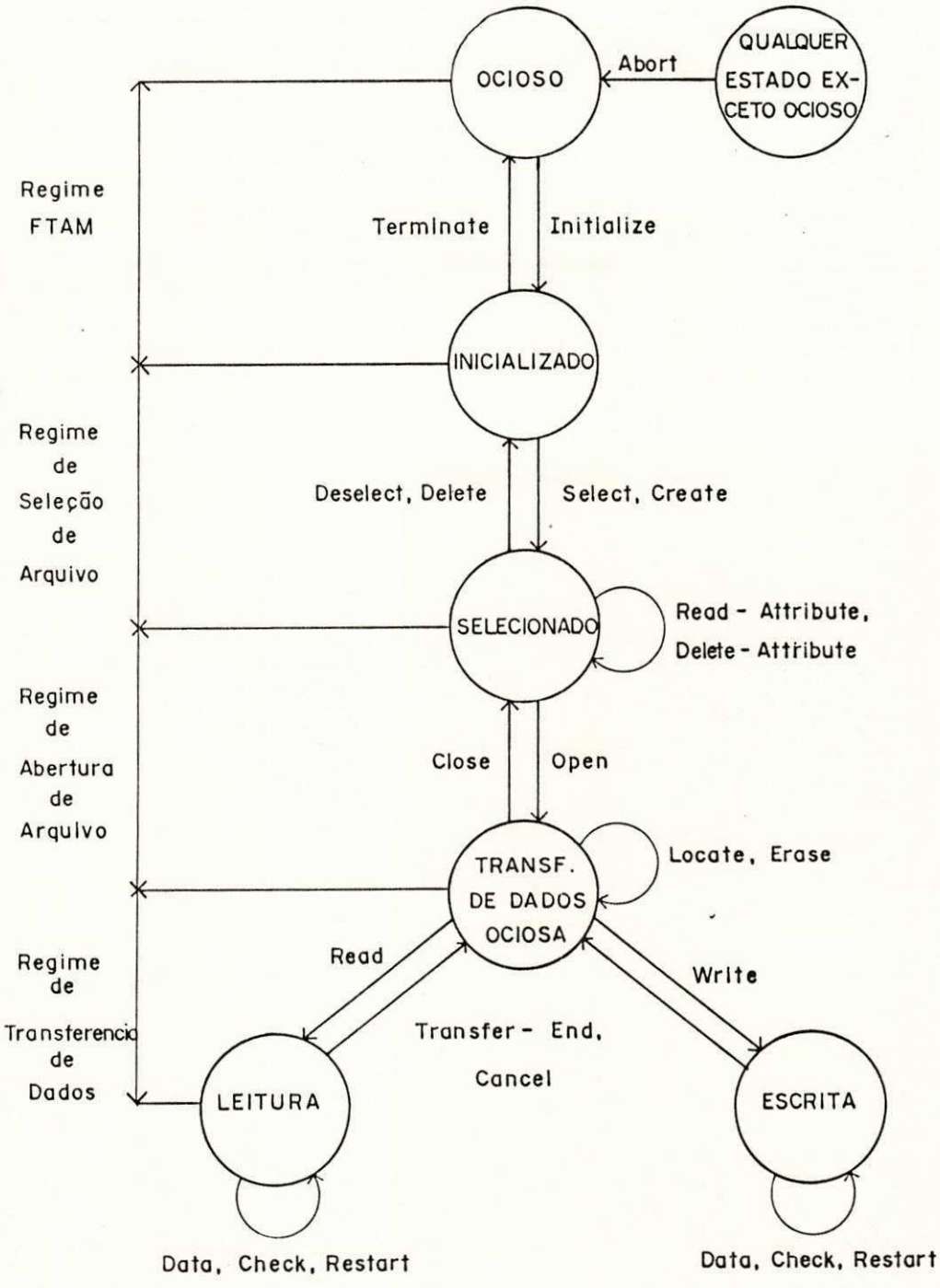


Fig. 2.7 : Diagrama Simplificado de Estados/Eventos e Regimes Associados

O serviço de arquivo é muito rico em funções, uma ampla faixa de tipos de aplicações pode ser suportada. Se fosse permitida uma escolha arbitrária das funções a serem realizadas, haveria, no outro lado, uma pequena chance de uma escolha similar ser feita e, consequentemente, uma pequena probabilidade de comunicação.

Para solucionar esse problema, o serviço FTAM define dois tipos de seleção funcional. No mais básico, as funções definidas são agrupadas em UNIDADES FUNCIONAIS. Uma implementação deve suportar uma unidade funcional completamente ou não suportá-la; isto reduz o número de escolhas a serem feitas. São definidos mecanismos que permitem negociação de unidades funcionais quando o regime FTAM é inicializado, e isto faz com que duas entidades comunicantes cheguem a um entendimento comum do conjunto de unidades funcionais disponíveis.

Isto em si reduz a variedade, contudo, ainda deixa considerável liberdade na escolha a ser feita. Uma convergência subsequente é alcançada pela definição de CLASSES DE SERVIÇOS. Uma classe de serviço consiste na combinação de um grupo de unidades funcionais para um determinado propósito.

Deste modo, em cada inicialização de uma associação de aplicação é necessário negociar a Classe de Serviço, as Unidades Funcionais opcionais e também o Nível de Serviço, que pode ser Confiável ou Corrigível pelo Usuário.

No serviço de arquivo confiável o usuário especifica seu padrão de qualidade de serviço, mas não se preocupa com a recuperação de erros, delegando tal atividade para o provedor do serviço de arquivo. O serviço de arquivo corrigível pelo usuário inclui primitivas que fornecem aos usuários do serviço, facilidades para recuperação de erros e gerência da transferência, tendo estes então, uma flexibilidade na escolha de estilos de gerência de erros, dependendo da necessidade da aplicação.

Foram definidas cinco Classes de Serviço:

T = Classe de Transferência de Arquivo

A = Classe de Acesso a Arquivo

G = Classe de Gerenciamento de Arquivo

TG = Classe de Tranferência e Gerenciamento de Arquivo

I = Classe Irrestrita (definida pelo usuário)

A tabela 2.4 mostra os serviços definidos para cada unidade funcional e quais unidades são mandatórias (m) ou opcionais (o) dentro de cada Classe de Serviço. Onde:

* = pelo menos uma das unidades de escrita ou leitura deve ser implementada.

branco = unidade funcional não permitida.

UNIDADES FUNCIONAIS	SERVIÇOS	CLASSES DE SERVIÇO T	A	G	TG	I
NÚCLEO	estabelecimento do regime FTAM término ordenado ou abrupto do regime FTAM seleção e desseleção de arquivo	m	m	m	m	m
LEITURA	leitura de massa de dados transf.de unid. de dados fim de transf. dos dados fechamento e abertura de arquivo	*	m		*	o
ESCRITA	escrita de massa de dados transf.de unid. de dados fim de transf. dos dados cancelamento de transferência de dados fechamento e abertura de arquivo	*	m		*	o
ACESSO A ARQUIVO	localizar e apagar FADU	m				o
GERÊNCIA DE ARQUIVO LIMITADA	criar arquivo deletar arquivo ler atributo	o	o	m	o	o
GERÊNCIA DE ARQUIVO AMPLIADA	alterar atributos (requer a anterior)	o	o	m	o	o
AGRUPAMENTO	início/fim de agrupamento	m	o	m	m	o
RECUPERAÇÃO	recuperação de regime ponto de verificação cancel.de transf. de dados	o	o	o	o	o
REINÍCIO DE TRANSF. DE DADOS	recomeço de transferência de dados ponto de verificação cancelamento transf. dados	o	o	o	o	o

Tab. 2.4: Unidades Funcionais e Classes de Serviço

Esses serviços são realizados invocando uma sequência de primitivas de serviço de arquivo. A tabela 2.5 apresenta as primitivas de serviço associadas com o tipo, usuário que emite a primitiva, as funções de cada uma e o regime a que pertencem.

NOME DA PRIMITIVA	TIPO CONFIRMADA	EMITIDA PELO	FUNÇÃO	REGIME
F_INITIALIZE	sim	inic.	Estabelecimento da Associação de Aplicação	Conexão FTAM
F_TERMINATE	sim	inic	Liberação normal da Associação de Aplicação	
F_U_ABORT	não	um ou outro	Liberação abrupta da Associacão de Aplicação	
F_P_ABORT	não	provedor	Idem a F_U_ABORT	
F_SELECT	sim	inic	Seleção de um ar quivo no sistema de arq. remoto	Seleção de Arquivo
F_DESELECT	sim	inic	Desseleção de ar quivo anterior mente selecionado	
F_CREATE	sim	inic	Criação (seleção) de um arquivo no sist. arq. remoto	
F_DELETE	sim	inic	Remoção (dessele ção) de arquivo anteriormente selecionado	

Tab. 2.5: Primitivas do Serviço FTAM

NOME DA PRIMITIVA	TIPO CONFIRMADA	EMITIDA PELO	FUNÇÃO	REGIME
F_READ_ ATTRIBUTE	sim	inic	Leitura dos atributos do arquivo selecionado	Seleção de Arquivo
F_CHANGE_ ATTRIBUTE	sim	inic	Modificação dos atributos do arquivo selecionado	
F_OPEN	sim	inic	Abertura do arquivo selecionado	Abertura de Arquivo
F_CLOSE	sim	inic	Fechamento do arquivo selecionado	
F_RECOVER	sim	inic	Recuperação do regime de abertura de arquivo	
F_LOCATE	sim	inic	Localização de uma FADU do arquivo aberto	
F_ERASE	sim	inic	Localização de uma FADU do arquivo aberto	
F_READ	não	inic	Pedido de leitura do arquivo selecionado	Transferência de Dados
F_WRITE	não	inic	Pedido de gravação de dados no arq. selecionado	
F_DATA	não	enviador	Dados do arquivo transferido	
F_DATA_END	não	enviador	Indicação de fim de dados do arquivo transferido	

continuação Tabela 2.5

NOME DA PRIMITIVA	TIPO CONFIRMADA	EMITIDA PELO	FUNÇÃO	REGIME
F_RESTART	sim	um ou outro	Pedido de retransmissão de dados do arquivo sendo transferido	Transferência de Dados
F_CHECK	sim	enviador	Confirmação dos dados recebidos	
F_CANCEL	sim	um ou outro	Cancelamento da transferência de dados em andamento	
F_TRANSFER_END	sim	inic	Término (normal) da transferência de dados	

continuação Tabela 2.5

2.3.3 PROTOCOLO DE TRANSFERÊNCIA, ACESSO E GERENCIAMENTO DE ARQUIVOS

O protocolo FTAM detalha o comportamento que deve ser executado pela entidade da camada para oferecer os serviços de transferência, acesso e gerenciamento de arquivo.

Este protocolo deve garantir o encadeamento dos regimes definidos anteriormente, tratando e verificando a consistência das primitivas de serviço emitidas pelo elemento de usuário e as unidades de dados de protocolo recebidas da entidade parceira.

A ISO especifica dois tipos de protocolos que suportam os serviços de arquivos definidos anteriormente: o PROTOCOLO BÁSICO e o PROTOCOLO DE RECUPERAÇÃO DE ERRO.

O protocolo básico suporta o serviço de arquivo corrigível pelo usuário, fornecendo as seguintes funções:

. representação das primitivas de serviço como uma sequência de UDPs (Unidades de Dados de Protocolo) para transmissão pelo CASE e Apresentação;
. agrupamento, quando apropriado, das primitivas de serviço numa única primitiva P-Data request;
. garantia do progresso do protocolo.

O protocolo de recuperação de erro suporta o serviço de arquivo confiável e fornece as seguintes funções:

. gerenciamento da informação de recuperação de erro durante a operação normal do serviço de arquivo;

. reinício da transferência de dados após interrupção, dentro do regime de transferência de dados;
. recuperação do término anormal do regime de seleção ou abertura de arquivo e
. recuperação de término anormal que destrói o serviço de arquivo corrigível pelo usuário.

Existe uma correspondência entre as primitivas de serviço e as UDPs (Unidades de Dados de Protocolo), isto é, para cada primitiva de serviço emitida por um usuário existe uma unidade de dados de protocolo com o mesmo nome e basicamente com os mesmos parâmetros da primitiva. Por exemplo:

PRIMITIVA DE SERVIÇO	UDP
F-INITIALIZE request	UDP F-INITIALIZE request
F-INITIALIZE response	UDP F-INITIALIZE response

As UDPs do protocolo de transferência, acesso e gerenciamento de arquivos são tipos de dados complexos, definidos por uma sintaxe abstrata - ASN.1 ("Abstract Sintaxe Notation One") [14],[15]. A sequência em que as UDPs são trocadas é a mesma seguida pelo serviço, obedecendo o mesmo encadeamento de regimes e fases.

O FTAM utiliza-se dos serviços do CASE [16] e da cama-

da de APRESENTAÇÃO [17]. O CASE, subconjunto ACSE, fornece somente os serviços relacionados com o regime de estabelecimento de conexão e a camada de Apresentação encarrega-se de prestar todos os serviços relacionados com os outros regimes. Esses conceitos são apresentados de forma esquemática na figura 2.8.

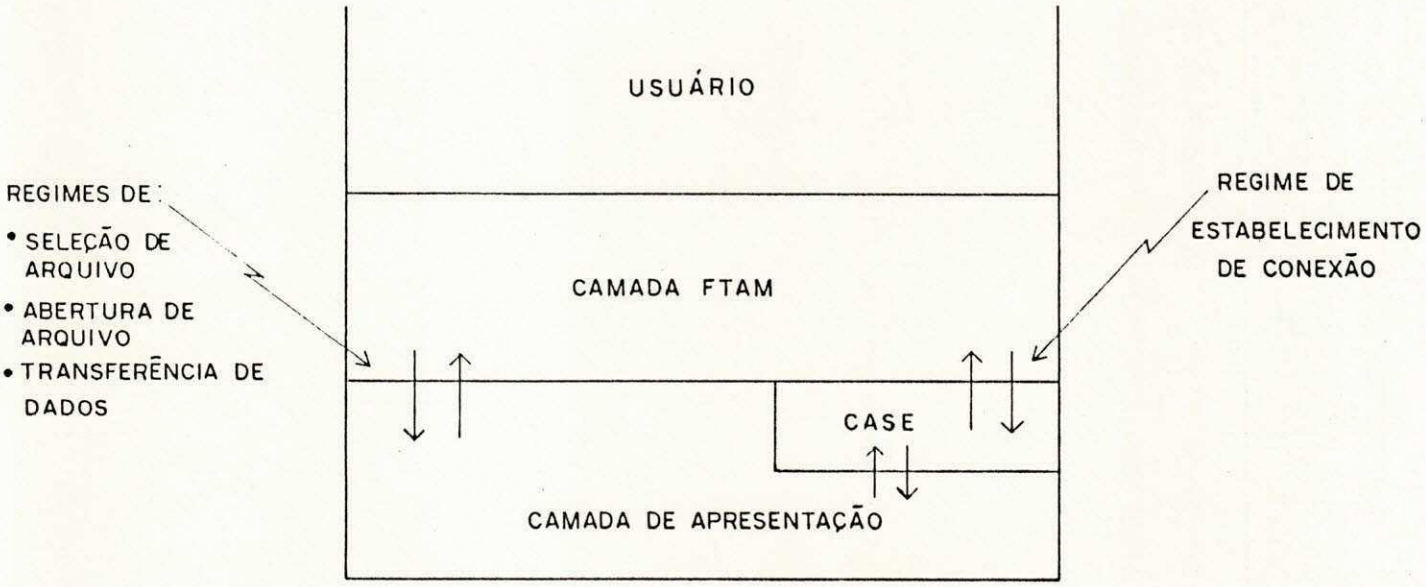


Fig. 2.8 Relação Camada Ftam Com Case e Apresentação

As tabelas 2.6 e 2.7 mostram as primitivas do CASE e Apresentação que conduzem as UDPs do FTAM, bem como as unidades funcionais que se relacionam com cada UDP.

NOME	CONDUZIDA POR	UNIDADES FUNCIONAIS
F-INITIALIZE request	A-ASSOCIATE	Núcleo
F-INITIALIZE response *	A-ASSOCIATE	Núcleo
F-TERMINATE request	A-RELEASE	Núcleo
F-TERMINATE response *	A-RELEASE	Núcleo
F-P-ABORT request	A-ABORT	Núcleo
F-U-ABORT request	A-U-ABORT	Núcleo

Tab. 2.6: UDPs de Estabelecimento do Regime FTAM - conduzidas nas primitivas do CASE

As UDPs marcadas com asterisco são conduzidas nas primitivas de serviço response e confirm. As demais são levadas nas primitivas de serviço request e indication.

NOME	CONDUZIDA POR	UNIDADES FUNCIONAIS
F-SELECT request	P-DATA	Núcleo
F-SELECT response	P-DATA	Núcleo
F-DESELECT request	P-DATA	Núcleo
F-DESELECT response	P-DATA	Núcleo
F-CREATE request	P-DATA	Gerência Arq.Limitada
F-CREATE response	P-DATA	Gerência Arq.Limitada
F-DELETE request	P-DATA	Gerência Arq.Limitada
F-DELETE response	P-DATA	Gerência Arq.Limitada
F-READ-ATTRIB request	P-DATA	Gerência Arq.Limitada
F-READ-ATTRIB response	P-DATA	Gerência Arq.Limitada
F-CHANGE-ATTRIB request	P-DATA	Gerência Arq.Limitada
F-CHANGE-ATTRIB response	P-DATA	Gerência Arq.Limitada
F-OPEN request	P-DATA	Núcleo
F-OPEN response	P-DATA	Núcleo
F-CLOSE request	P-DATA	Núcleo
F-CLOSE response	P-DATA	Núcleo
F-BEGIN-GROUP request	P-DATA	Agrupamento
F-BEGIN-GROUP response	P-DATA	Agrupamento
F-END-GROUP request	P-DATA	Agrupamento
F-ENDE-GROUP response	P-DATA	Agrupamento
F-RECOVER request	P-DATA	Recuperação
F-RECOVER response	P-DATA	Recuperação
F-LOCATE request	P-DATA	Acesso
F-LOCATE response	P-DATA	Acesso
F-ERASE request	P-DATA	Acesso
F-ERASE response	P-DATA	Acesso

Tab. 2.7: UDPs do Regime de Arquivo - conduzidas nas primitivas de Apresentação

Capítulo 3

CONSIDERAÇÕES SOBRE O SISTEMA OPERACIONAL UNIX

3.1 INTRODUÇÃO

O sistema operacional UNIX é um sistema multi-tarefa, multiusuário e de uso geral. Possui um sistema de arquivos hierarquizado em árvore, onde todos os recursos são tratados como arquivos. Os programas são executados como processos que, por sua vez, podem criar outros processos, em uma relação de dependência hierárquica do tipo "pai" e "filhos".

O UNIX possui chamadas ao supervisor ("system calls") que fornecem o controle necessário para fazer algumas operações que não estão disponíveis na biblioteca de entrada e saída ou que estão disponíveis, mas não são bastante eficientes para satisfazer as necessidades de determinados programas do usuário.

Na implementação do FTAM foram utilizadas várias chamadas ao supervisor. Pela importância que apresentam no Modelo Funcional da implementação que será apresentado no capítulo 4, merecem destaque aquelas relacionadas com criação de processos e comunicação interprocessos. A seguir, faz-se uma descrição sobre elas.

3.2 CHAMADAS AO SUPERVISOR PARA CRIAÇÃO DE PROCESSOS

As chamadas ao supervisor utilizadas para criação de processos são as seguintes:

. **fork ( ) e**

. **a família execl ( )**

Para se entender essas chamadas, é necessário primeiramente compreender o método que o Unix utiliza para transformar um programa em um processo e, a partir daí, distinguir entre um e outro.

Se a compilação foi executada com sucesso, o link-editor produz um arquivo binário executável contendo as seguintes seções:

CABEÇALHO: que define os tamanhos de todas as outras seções no arquivo e também identifica o ponto de entrada onde a execução deve começar quando o programa tornar-se um processo.

TEXTO: são as instruções que a UCP executará.

DADOS: consiste de todos os dados inicializados.

BSS: consiste de todos os dados não inicializados. Quando o programa é transformado em um processo, o Unix inicializa para zero todas as variáveis nesta secão.

RELOCAÇÃO: define como o link-editor deve modificar o arquivo de programa quando este é ligado com outros arquivos de programa. Quando a ligação é terminada com sucesso, a parte de relocação do arquivo programa é removida.

TABELA DE SíMBOLOS: relaciona os símbolos para localização no processo.

Um processo é constituído dos três segmentos listados abaixo:

TEXTO: é a imagem da seção de texto do programa.

DADOS: esta seção é constituída da parte de Dados do arquivo programa e da seção BSS com as variáveis inicializadas com zero. Esta área pode ser aumentada ou diminuída o quanto necessário. As

chamadas sbrk ( ) e brk ( ) controlam o seu tamanho.

PILHA: esta porção é criada quando o Unix constrói o processo. Ela é usada para guardar todas as variáveis alocadas, os argumentos do programa e o ambiente do processo. O Unix, se necessário, faz a pilha aumentar automaticamente.

A figura 3.1 mostra como as partes de um arquivo binário executável são transformadas em um processo.

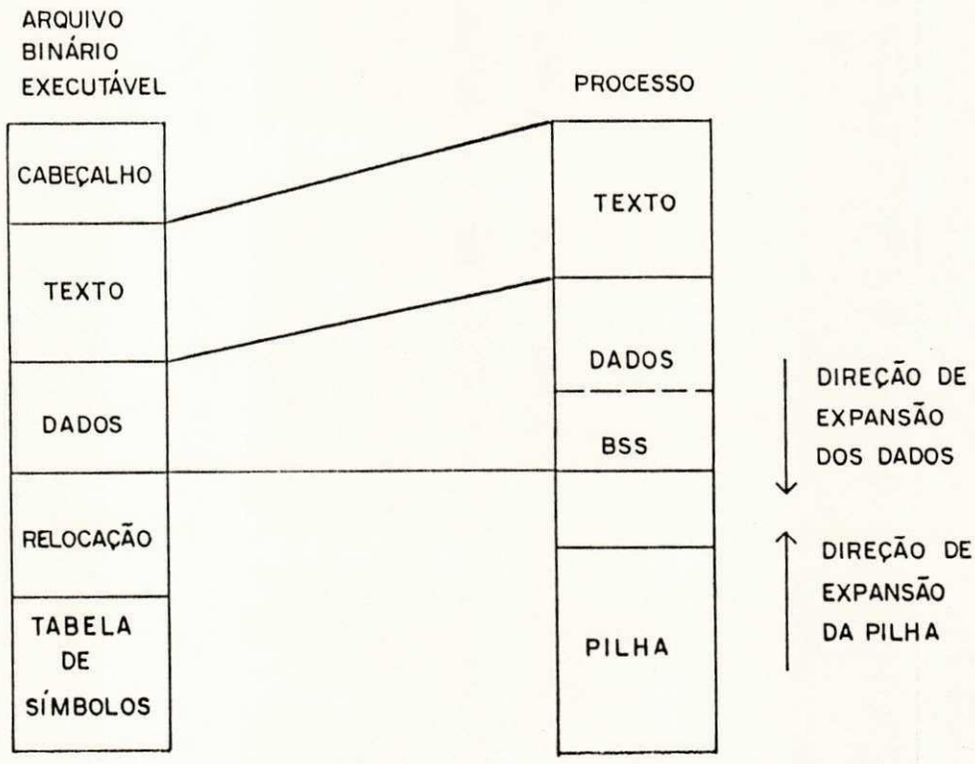


Fig. 3.1 : Transformação de um Arquivo Binário Executável em um Processo

Em resumo, pode-se dizer que um processo é todo um ambiente de execução de um programa, consistindo de instruções, dados do usuário e segmentos de dados do sistema, enquanto que um programa é um arquivo contendo instruções e dados que serão usados para inicializar as instruções e dados de usuário de um processo [18].

As chamadas ao supervisor que transformam um programa em um processo são as da família exec. Na implementação do FTAM usa-se a chamada execl ( ), cuja sintaxe é a seguinte:

```
int
execl (percurso,arg0,[arg1,...,argn],0)
char  *percurso, *arg0, *arg1,..., *argn;
```

A chamada execl ( ) transforma o processo chamador em um "novo processo", que é construído a partir de um arquivo binário executável. O número do processo não muda, apenas a IMAGEM do processo muda, ou seja, os segmentos.

O argumento percurso aponta para o nome de percurso que identifica o arquivo binário executável a ser transformado em um processo. Arg0, arg1,..., argn definem os argumentos a serem passados para o novo processo. Eles são apontadores para cadeias de caracteres terminadas pelo caracter nulo.

Quando um arquivo binário executável é transformado em um processo, o Unix preserva algumas características do processo chamador. Dentre elas tem-se:

. O valor da prioridade;
. Identificação do processo e do grupo;
. O tempo que resta até um sinal de alarme de relógio;
. O diretório de trabalho e o diretório raiz;
. Máscara de criação do modo de arquivo;
. Limite do tamanho do arquivo;
. Todos os arquivos abertos.

A família exec transforma um arquivo binário executável em um processo que substitui o processo que fez a chamada exec, isto é, não cria um novo processo e sim substitui o processo original. Caso se deseje salvar o processo original, deve-se criar duas cópias dele: uma que faz a chamada exec e a outra que continua a sua execução. A primitiva responsável por essa função é a fork ( ), cuja sintaxe é a seguinte:

```
int
fork ( )
```

A primitiva fork ( ) faz o sistema Unix criar um novo processo (processo filho), que é uma cópia exata da imagem do processo chamador (processo pai). O novo processo herda várias das características do processo pai. Dentre elas, tem-se:

. O ambiente;
. Todos os sinais;
. Todos os arquivos abertos;
. A identificação de usuário e de grupo;
. O diretório de trabalho e o diretório raiz;
. A máscara de criação de arquivo;
. O limite do tamanho de arquivo;
. O último tempo até o alarme de um clock de sinal.

Se bem sucedida, a chamada fork ( ) retorna zero para o processo filho e a identificação do processo filho para o processo pai, caso contrário retorna -1.

Existe uma forte relação entre essas duas chamadas ao supervisor pois, sem fork, exec é de uso limitado e, sem exec, fork não tem uso prático.

3.3 CHAMADAS AO SUPERVISOR PARA COMUNICAÇÃO INTERPROCESSOS

Na implementação do FTAM utilizou-se os seguintes recursos para comunicação interprocessos:

. Filas de Mensagens e

. Memória Compartilhada.

Esses dois mecanismos utilizam um tipo comum de estrutura de dados, contendo informações que serão usadas na determinação de permissões, quando da realização de operações envolvendo comunicações interprocessos (ipc - "Interprocess Communication"). A estrutura é a seguinte:

```
struct  ipc_perm {          /* estrutura de permis-
                               sões */
        ushort cuid;        /* identificação de usuá-
                               rio do criador */
        ushort cgid;        /* identificação de grupo
                               do criador */
        ushort uid;         /* identificação de usuá-
                               rio (atual dono) */
        ushort gid;         /* identificação de grupo
                               (atual dono) */
        ushort mode;        /* permissões de leitura
                               e escrita */
);
```

3.3.1 FILAS DE MENSAGENS

A mensagem é uma unidade de informação de tamanho variável sem formato pré-definido, podendo ser de tipos diferentes.

As mensagens são enviadas para as filas e as filas de mensagens possuem uma chave que permite aos processos terem acesso a elas. Cada fila de mensagem tem um dono, um grupo e um conjunto de permissões que definem se o dono, grupo ou todos os outros podem ler e escrever mensagens na fila.

Todas as operações com filas de mensagens usam um identificador de fila que é conceitualmente similar a um descritor de arquivo. Esse identificador de fila é um inteiro positivo único obtido através da chamada ao supervisor **msgget ( )**. Cada identificador está associado com a fila de mensagem e uma estrutura de dados. A estrutura de dados associada é a seguinte:

```
struct msqid_ds {
        struct ipc_perm  msg_perm;   /* permissões de operação
                                        */
        ushort           msg_qnum;   /* número de mensagens na
                                        fila */
        ushort           msg_qbytes; /* número máx. de bytes na
                                        fila */
        ushort           msg_lspid;  /* ident.do proc. que fez a
                                        ultima chamada msgsnd */
        ushort           msg_lrpid;  /* ident.do proc. que fez a
                                        ultima chamada msgrcv */

        ushort           msg_stime;  /* data da última chamada
                                        msgsnd */
        ushort           msg_rtime;  /* data da última chamada
                                        msgrcv */
        ushort           msg_ctime;  /* data */
};
```

Abaixo, mostra-se a sintaxe da **msgget ( )**:

```
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/msg.h>

int
msgget (chave,msgflg)
key_t  chave;
int    msgflg;
```

Com a msgget ( ) um identificador de fila de mensagem e sua estrutura de dados msqid associada são criados para a chave fornecida.

A chave é o nome da fila de mensagem e o msgflg é uma combinação do seguinte:

IPC_CREAT - cria-se uma fila de mensagem denominada chave caso ela ainda não exista. Do contrário, IPC_CREAT será ignorado.

IPC_EXCL - quando combinado com IPC_CREAT força o retorno de um erro se a fila de mensagem já existir.

modo - ao criar-se uma fila usando a chamada ao supervisor msgget( ), os 9 bits menos significativos de msgflg definem o modo de acesso.

Uma vez que a fila de mensagem foi criada, pode-se enviar e receber mensagens com as chamadas msgsnd( ) e msgrcv ( ), cuja sintaxe é:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int
msgsnd (msqid,msgp,msgsz,msgflg)
int     msqid, msgsz, msgflg;
struct  msgbuf *msgp;

int
msgrcv (msqid,msgp,msgsz,msgtyp,msgflg)
int     msqid, msgsz, msgflg;
struct  msgbuf *msgp;
long    msgtyp;
```

A chamada msgsnd ( ) é usada para enviar uma mensagem para a fila associada ao identificador de fila de mensagens msqid retornado por msgget ( ). O argumento msgp é um apontador para o buffer do usuário que contém a mensagem, cuja estrutura de dados deve ser da forma:

```
struct msgbuf {
        long mtype;     /* tipo da mensagem */
        char mtext[ ]; /* texto da mensagem */
);
```

A msgrcv( ) lê uma mensagem da fila associada ao msqid e coloca no buffer do usuário apontado por msgp.

O argumento msgsz é o tamanho de mtext em bytes e msgflg é uma combinação do seguinte:

.IPC_NOWAIT - para msgsnd ( ), IPC_NOWAIT significa que caso não haja espaço no núcleo para armazenar a mensagem, o valor -1 deverá ser retornado. Os processos normalmente aguardam um espaço para armazenamento da mensagem.

para msgrcv ( ), significa que se não houver mensagens a serem lidas será retornado o valor -1 imediatamente. Normalmente os processos esperam até que haja mensagens para ler.

.MSG_NOERROR- ignorado para msgsnd ( ).

para msgrcv( ), significa que mensagens de tamanho superior a **tamanho** serão reduzidas ao número de octetos especificado por **tamanho**; a parte truncada se perde e não há indicação do truncamento para o processo solicitante.

Finalmente, o argumento **msgtyp** define o tipo de mensagem a ser recebida, de acordo com a tabela 3.2:

TIPO	DESCRIÇÃO
0	A primeira mensagem da fila é retirada, independente do seu tipo (acesso FIFO)
>0	A primeira mensagem do tipo "msgtyp" é retirada da fila
<0	A primeira mensagem cujo tipo for menor ou igual ao valor absoluto de "msgtyp" será recebida

Tab. 3.1: Valores assumidos pelo argumento msgtyp.

Msgrcv ( ) retorna o número de bytes recebidos e msgsnd ( ) retorna zero.

A última chamada relacionada com filas de mensagens é a msgctl ( ) que proporciona uma variedade de operações de controle de mensagens. A sintaxe de msgctl ( ) é:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

int
msgctl (msqid, cmd, buf)
int     msqid, cmd;
struct  msqid_ds  *buf;
```

As operações oferecidas por msgctl ( ) são especificadas no argumento **cmd**. Os valores disponíveis são:

. IPC_STAT - coloca o valor corrente de cada membro da estrutura de dados associada com msqid na estrutura apontada por buf.

. IPC_SET - atribui aos campos msg_perm.uid, msg_perm.gid, msg_perm.mode e msg_qbytes os valores correspondentes encontrados na estrutura apontada por **buf**.

. IPC_RMID - remove o identificador de fila de mensagem **msqid**.

Se for bem sucedida, msgctl retorna 0, caso contrário retorna -1.

3.3.2 MEMÓRIA COMPARTILHADA

Memória compartilhada é a memória principal do computador compartilhada por um ou mais processos [18]. Isto é, a mesma porção física da memória principal aparece no espaço de endereçamento lógico de um ou mais processos. A figura 3.2 mostra dois processos compartilhando a mesma área de memória principal do computador.

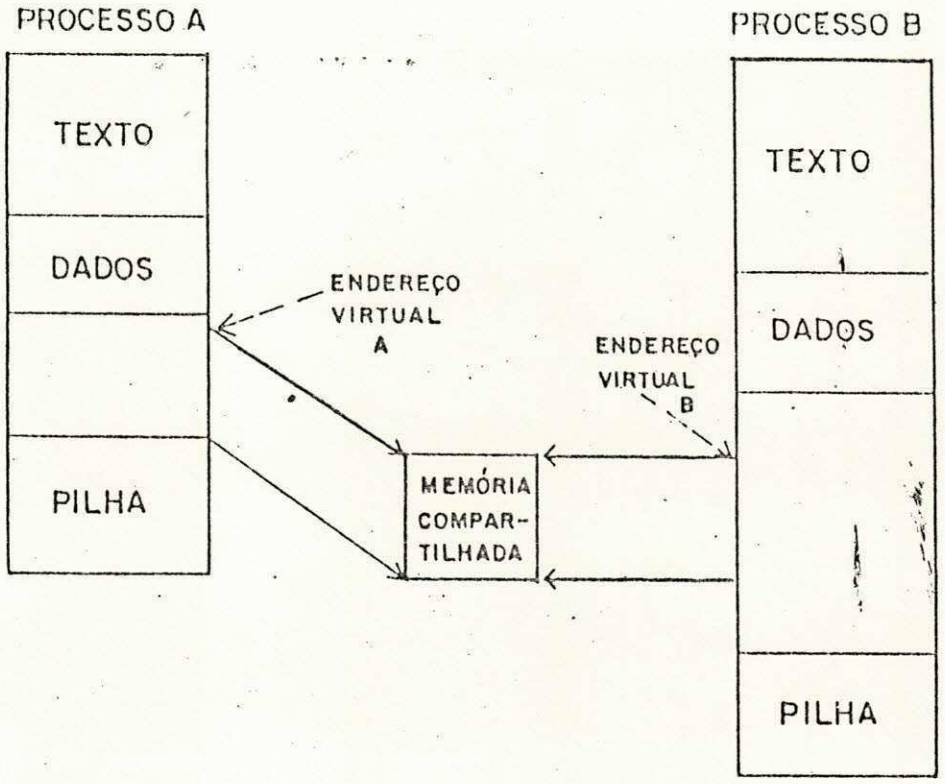


Fig. 3.2 : Compartilhamento de Memória Entre Dois Processos

O acesso à memória compartilhada começa com a chamada shmget ( ) cuja sintaxe é:

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>

int
shmget (chave, tamanho, shmflg)
key_t  chave;
int    tamanho, shmflg;
```

Com a chamada shmget ( ) um identificador de memória compartilhada com sua estrutura de dados associada e um segmento de memória compartilhada são criados para a chave fornecida. A estrutura de dados associada ao segmento de memória compartilhada é a seguinte:

```
struct shmid_ds {
        struct ipc_perm shm_perm;    /* permissões de operação
                                        */
        int             shm_segsz;   /* tamanho do segmento */
        ushort          shm_cpid;    /* ident.do proc.criador */
        ushort          shm_lpid;    /* ident.do proc. a fazer a
                                        última operação */
        short           shm_nattch;  /* número de vinculações
                                        correntes */
        short           shm_cnattch; /* número de processos na
                                        memória vinculados */
        time_t          shm_atime;   /* data da última vincula-
                                        ção */
        time_t          shm_dtime;   /* data da última desvincu-
                                        lação */
        time_t          shm_ctime;   /* data da última alteração
                                        */
};
```

O argumento tamanho é o tamanho em bytes do segmento de memória compartilhada e shmflg é uma combinação do seguinte:

. IPC_CREAT - o segmento de memória compartilhada denominado chave é criado se ele ainda não existe. Caso o segmento já exista, IPC_CREAT é ignorado.

. IPC_EXCL - quando combinado com IPC_CREAT, IPC_EXCL retorna um erro quando o segmento de memória compartilhada já existe.

. modo - os 9 bits menos significativos de shmflg definem o modo de acesso ao segmento de memória compartilhada.

Uma vez criado o segmento de memória compartilhada, as chamadas shmat ( ) e shmdt ( ) são usadas para inserí-lo e remove-lo no espaço de endereçamento do processo chamador. Suas sintaxes são:

```
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/shm.h>

char
shmat (shmid, at_addr, shmflg)
int     shmid, shmflg;
char    *at_addr;

int
shmdt (dt_addr)
char    *dt_addr;
```

O argumento shmid é o identificador do segmento de memória compartilhada retornado por shmget ( ).

O at_addr é o endereço no espaço de endereçamento do processo onde o segmento de memória compartilhada será vinculado. Se esse endereço for igual a 0, o segmento será vinculado a um endereço selecionado pelo sistema UNIX. O dt_addr representa o endereço no espaço de endereçamento do processo que será desvinculado do segmento de memória compartilhada e o seu valor é o endereço retornado pela chamada shmat ( ).

O shmflg pode ser SHM_RDONLY significando que o segmento de memória compartilhada será vinculado somente para leitura; ou zero, significando que o segmento será vinculado para leitura e escrita.

Se bem sucedida, shmat ( ) retorna o endereço no espaço de endereçamento do processo onde o segmento de memória compartilhada foi vinculado. shmdt ( ) retorna zero. Caso contrário, retornam -1.

As operações para controle de memória compartilhada são oferecidas pela chamada shmctl ( ), cuja sintaxe é:

```
#include  <sys/types.h>
#include  <sys/ipc.h>
#include  <sys/shm.h>

int
shmctl (shmid, cmd, buf)
int     shmid, cmd;
struct  shmid_ds  *buf;
```

Essas operações são especificadas pelo argumento cmd cujos valores válidos são:

. IPC_STAT - coloca o valor atual de cada membro da estrutura de dados associada à shmid na estrutura apontada por buf.

. IPC_SET - seta os membros shm_perm.uid, shm_perm.gid, shm_perm.mode com o valor encontrado na estrutura apontada por buf.

. IPC_RMID - remove do sistema o identificador de memória compartilhada especificado por shmid.

Se executada com sucesso, a função msgctl ( ) retorna zero, caso contrário -1.

Capítulo 4

ESTRUTURAÇÃO E IMPLEMENTAÇÃO DO FTAM

4.1 INTRODUÇÃO

O objetivo deste capítulo é apresentar o modelo funcional utilizado na implementação dos serviços e protocolo FTAM e descrever os processos e estruturas de dados envolvidas.

Nele, faz-se considerações sobre os seguintes aspectos:

- a estrutura de processos;
- as facilidades de comunicação para troca de mensagens e sincronização entre processos;
- as estruturas de dados utilizadas;
- a interface entre camadas adjacentes e
- a forma como são efetuadas as trocas de primitivas de serviço e unidades de dados de protocolo (UDPs).

4.2 MODELO FUNCIONAL DA IMPLEMENTAÇÃO

Nesta seção apresenta-se o modelo funcional para a implementação do protocolo e das primitivas de acesso aos serviços FTAM, descrevendo os seguintes aspectos:

- a funcionalidade dos processos envolvidos;
- a comunicação interprocessos;
- a interface com as camadas adjacentes e
- as estruturas de dados de trabalho.

4.2.1 PROCESSOS

Uma estrutura multi-processo implementa o serviço FTAM e coordena as interfaces entre camadas. Nessa estrutura existe um "processo principal" que gerencia todos os demais, decidindo sobre a criação, alocação e morte destes.

A funcionalidade dos processos utilizados é a seguinte (Figura 4.1):

PGIC - PROCESSO GERENTE DA INTERFACE E DA CAMADA

É o processo que gerencia toda a camada, ou seja, os processos PGC e as interfaces com a camada inferior e superior. Controla todos os recursos utilizados pela camada e disponíveis pelo sistema operacional.

PGC - PROCESSO GERENTE DE CONEXÃO

É o processo que implementa a máquina do protocolo FTAM, gerenciando portanto as conexões ativas do protocolo citado.

O processo PGC executa a máquina de protocolo FTAM, verificando a sequencialidade das primitivas, fazendo a análise de predicados, executando as transições de estado e os procedimentos para cada estado da conexão; estimulado por eventos internos e externos à camada.

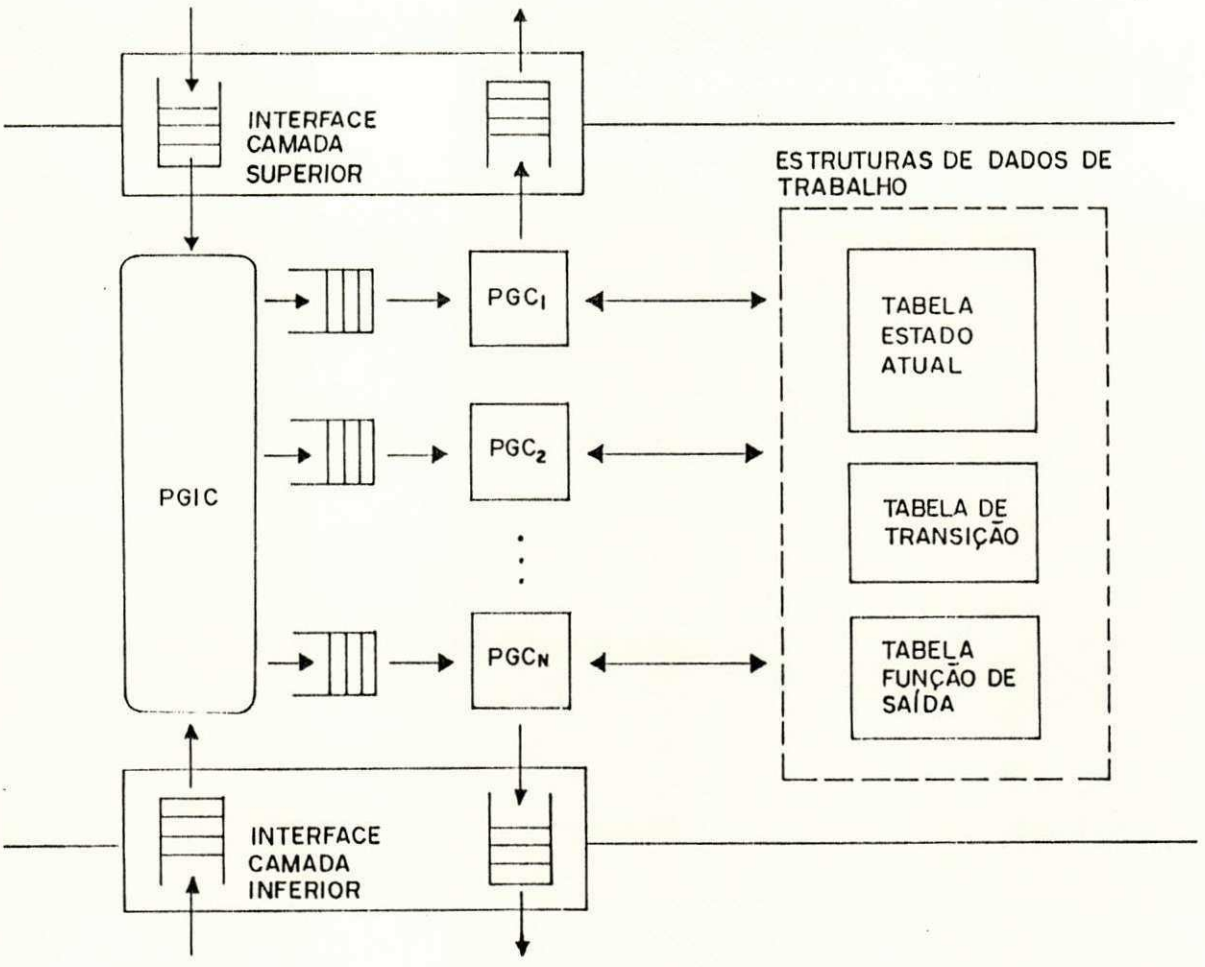


Fig. 4.1 : Esquema Funcional da Implementação

Existem instâncias do processo PGC para cada conexão FTAM ativa, isto é, cada processo PGC gerencia um ponto final de conexão interligando os processos aplicativos através de uma instância de mesmo tipo, no caso, o FTAM [19] (Figura 4.2).

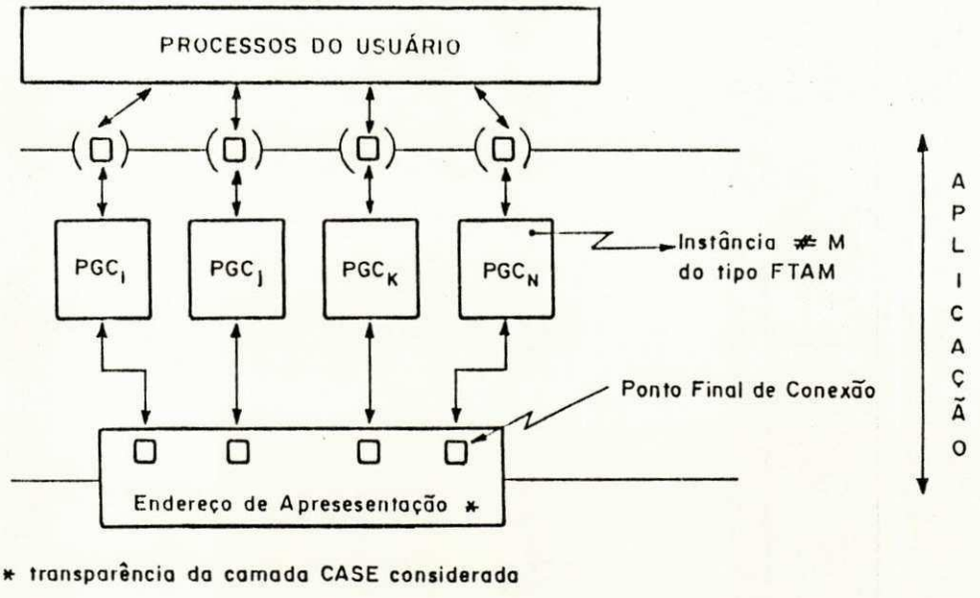


Fig. 4.2 - Instanciação dos Processos na Implementação do FTAM

O PGC usa uma estrutura de dados flexível, composta pelas tabelas de estado atual, transição e função de saída (Figura 4.1) discutidas adiante. Com base nessas estruturas de dados, ele executa funções tais como:

- desmontagem e montagem das primitivas e das unidades de dados de protocolo (UDPs);
- verificação da coerência dos parâmetros das primitivas e das UDPs;
- manutenção de todas as informações referentes ao estado atual da conexão e
- envio das primitivas e das UDPs montadas para as interfaces de saída.

O processo PGIC é o responsável pela gerência da camada, incluindo assim, a gerência dos demais processos envolvidos e a gerência das interfaces entre camadas adjacentes. Em resumo, o PGIC executa as seguintes tarefas básicas:

- criação dinâmica de processos PGC na medida da necessidade da camada, ou seja, em função do número de usuários ativos;
- criação de filas de mensagens utilizadas na comunicação com as camadas inferior e superior;
- criação de filas de mensagens para comunicação com os processos PGC;
- criação do segmento de memória compartilhada;
- monitoração dos recursos utilizados do sistema operacional (filas, memória compartilhada...) e
- recepção de mensagens nas interfaces e roteamento para o processo PGC gerente da conexão correspondente.

4.2.1.1 COMUNICAÇÃO ENTRE PROCESSOS DO MODELO

Numa estrutura multi-processo por camada [20], faz-se necessário a utilização de mecanismos de comunicação interprocesso (MCI) do sistema operacional utilizado [21] para:

- passagem de mensagens e
- passagem de parâmetros.

As mensagens contêm, tipicamente, as primitivas passadas entre camadas e as primitivas internas à camada, trocadas entre os processos PGIC e PGCs.

Os parâmetros passados são tipicamente informações utilizadas pelos processos durante todo o período em que a conexão permanece ativa, tais como: identificação de processos, semáforos, endereços etc....

Nesta implementação, os MCIs utilizados foram:

. filas de mensagens;

. segmento de memória compartilhada("shared memory") e

. sinais.

As filas de mensagens são mecanismos de interação fraca, bloqueante e, portanto, apropriado para o processamento de protocolos. Nesta implementação, quatro filas de mensagens fazem a interface entre camadas e uma fila adicional é utilizada para cada processo PGC ativo receber as mensagens que lhes são endereçadas (Figura 4.1).

Para a passagem de parâmetros entre processos utilizou-se o mecanismo de memória compartilhada. Como definido no Capítulo 3, a memória compartilhada corresponde à mesma porção física da memória principal, referenciada no espaço de endereçamento lógico de um ou mais processos. Isso permite que os processos PGIC e PGC tenham acesso a uma mesma área da memória onde estão parâmetros comuns aos dois.

A tabela de estado atual é a estrutura de dados que reside nesse segmento de memória compartilhada. Ela contém os parâmetros comuns aos processos PGIC e PGC, tais como: endereços, semáforos, número de processos, buffers, identificadores de filas, etc...

A sincronização entre os processos se dá através de um mecanismo de sinais via tabela de estado atual. O PGIC ao enviar uma mensagem para a fila do processo PGC, sinaliza-o, através da tabela de estado atual, de que existe mensagem para ele. Ao receber o sinal, o PGC retira a mensagem da fila e sinaliza o PGIC, via tabela de estado atual, que a mensagem foi recebida.

Uma ilustração dos mecanismos de comunicação interprocessos e da sincronização é mostrada na figura 4.3.

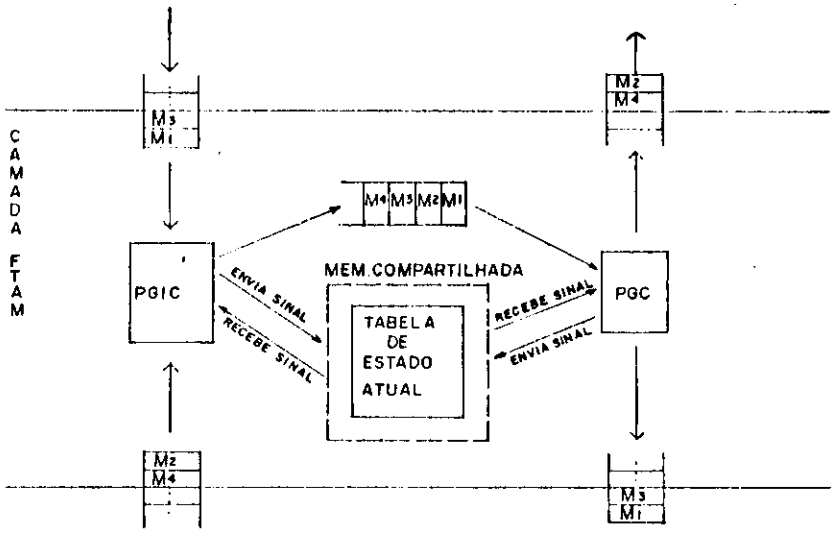


Fig. 4.3 : Mecanismos de Comunicação Interprocessos e Sincronização da Camada FTAM

4.2.2 INTERFACE COM AS CAMADAS ADJACENTES

Nesta implementação, a interface com as camadas adjacentes é composta por quatro filas de mensagens:

- fila de entrada superior;
- fila de entrada inferior;
- fila de saída superior e
- fila de saída inferior.

A **fila de entrada superior** recebe as primitivas de serviço oriundas dos processos de aplicação local. A **fila de entrada inferior** recebe as primitivas do CASE (ACSE) ou de Apresentação, carregando as UDPs do protocolo FTAM advindas da entidade remota.

A **fila de saída superior** contém as primitivas de serviço de confirmação que serão recebidas pela aplicação local. A **fila de saída inferior** contém as primitivas do CASE (ACSE) ou de Apresentação com as UDPs do FTAM que serão enviadas para a entidade remota.

O processo PGIC é o responsável pela criação, gerência e remoção destas filas. As operações permitidas para cada uma delas serão descritas mais adiante.

A figura 4.4 apresenta o esquema funcional das filas de mensagens envolvidas na implementação.

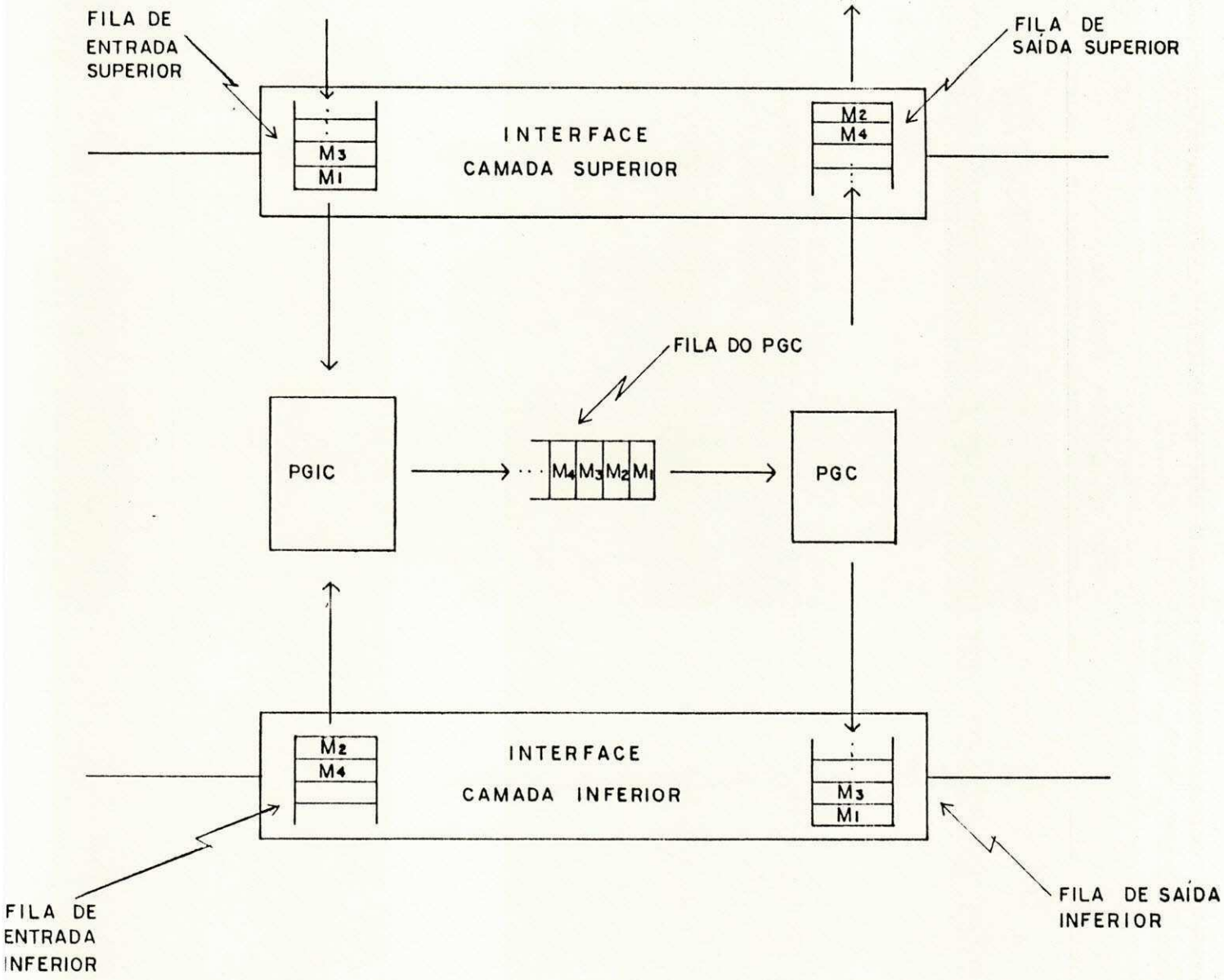


Fig. 4.4 : Esquema Funcional das Filas de Mensagens

4.2.4 ESTRUTURAS DE DADOS

As estruturas de dados básicas utilizadas pelos processos PGIC e PGC são as seguintes:

. Tabela de Estado Atual (TEA);
. Tabela de Transição (TT) e
. Tabela de Função de Saída (TFS).

A TEA armazena as informações referentes ao estado da conexão (endereços, classes e nível de serviço, unidades funcionais, atributos de arquivo, etc...) e parâmetros utilizados pelos processos PGIC e PGC (semáforos, buffers, número de processos, identificadores de filas, etc...).

Para cada entrada da tabela de estado atual existe um processo PGC associado. A inicialização dessa tabela é feita pelo processo PGIC.

A estrutura de dados da tabela de estado atual é mostrada na figura 4.5.

```
struct t_est_at {
        int       idf_assc;          /* identificação da associação */
        int       pr_gr_cnx;         /* identificação do processo gerente da (conexão */
        int       idfila;            /* identificação da fila do PGC */
        int       f_s_sup;           /* fila de saída superior */
        int       f_s_inf;           /* fila de saída inferior */
        int       est_cnx;           /* estado da conexão */
        int       sinal;             /* sinal de que existe mensagem na fila */
        int       flag;              /* flag de entrada livre (0) ou ocupada (1) */
        struct    t_msg  *msg;       /* apontador para o buffer intermediário do PGC */
        struct    t_p_ftam  p_ftam;  /* parâmetros do FTAM */
};


struct t_p_ftam {
        char      t_apl_ch;          /* título da aplicação chamada */
        char      t_apl_ct;          /* título da aplicação chamante */
        int       e_ap_ch;           /* endereço de apresentação chamado */
        int       e_ap_ct;           /* endereço de apresentação chamante */
        BOOLEAN   g_cont_ap;         /* gerência do contexto de apresentação */
        int       niv_serv;          /* nível de serviço */
        int       clas_serv;         /* classe de serviço */
        BITSTRING und_func;          /* unidades funcionais */
        BITSTRING gp_atb;            /* grupo de atributos */
        int       rbck;              /* disponibilidade de retrocesso */
        int       q_s_com;           /* qualidade do serviço de comunicação */
        struct    t_lst_cont lst_cont; /* lista do tipo de conteúdo */
        G_STRING  id_inic;           /* identidade do iniciador              */
        G_STRING  cont;              /* contabilidade */
        union     t_senha  senh;     /* senha do sistema de arquivo */
        int       chkp;              /* janela de checkpoint */
        int       r_est;             /* resultado de estado */
        int       r_ac;              /* resultado de ação */
        char      t_apl_r;           /* título da aplicação respondente */
        int       e_ap_r;            /* endereço de apresentação respondente */
        struct    t_diagn  diagn;    /* diagnóstico */
};
```

Fig. 4.5: Estrutura de Dados da Tabela de Estado Atual

A TT controla a verificação de predicados a partir da ocorrência de eventos e aponta para um conjunto de funções de saída, armazenadas na tabela de função de saída. Assim sendo, a partir da tabela de transição são inicialmente ativadas as funções que testam os predicados para o evento ocorrido, para em seguida serem executadas as ações de saída. Essas ações de saída são acionadas de acordo com o resultado do teste dos predicados já executados. Portanto, a tabela de transição aponta para um arranjo de funções de saída que são selecionadas a partir do valor de retorno das funções de teste de predicados.

A figura 4.6 ilustra a estrutura da tabela de transição:

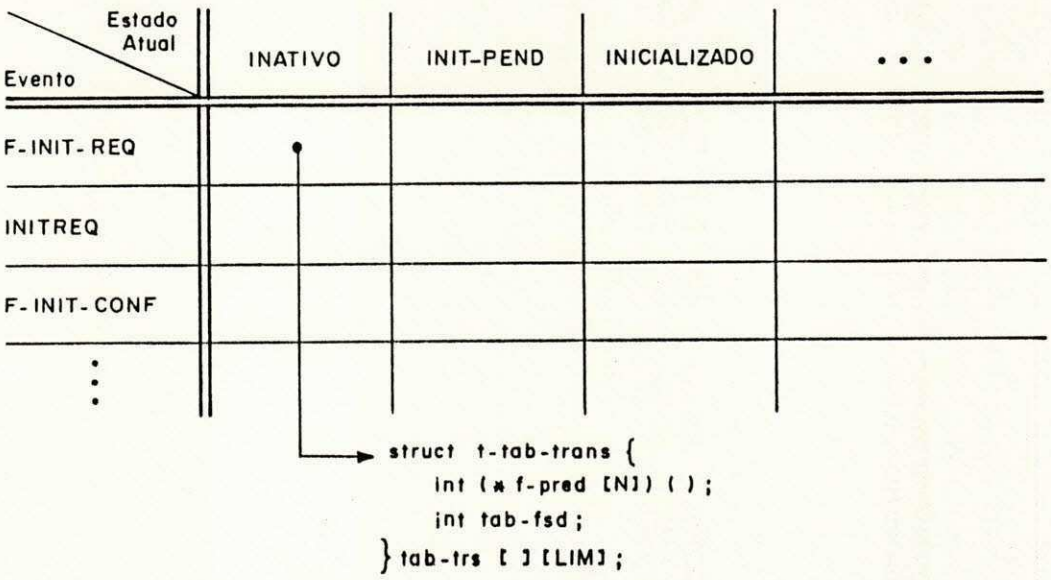


Fig. 4.6 : Estrutura de Dados da Tabela de Transição

Tem-se um arranjo bidimensional indexado pelos eventos e estados do FTAM. Para cada dupla evento/estado tem-se um arranjo de apontadores de funções de teste de predicado, que são executadas sequencialmente e um inteiro indexando uma entrada da tabela de função de saída descrita a seguir.

A TFS lança todas as ações pertinentes à transição ocorrida. Assim sendo, para cada dupla evento/estado, que leva a uma transição, existem arranjos de arranjos de apontadores de funções de saída. A seleção de qual arranjo de funções é acionado, levando a execução de um conjunto de ações de número variável, é determinada, num primeiro nível, pelo valor do campo tab_fsd (Figura 4.6) na estrutura da tabela de transição, e num segundo nível pelo valor de retorno das funções de teste de predicado, acionadas também através da tabela de transição [22].

Uma ilustração de uma entrada da tabela de função de saída é apresentada na figura 4.7:

```
int   ( * f_ac [ ] [9] [7]) ( ) = {
      { {NULL}, {inirq_pdu,ac1_rf,ac3_rf,init_pd,NULL},
        {f6_inicf_pr,closed,NULL} },
            .
            .
            .
            .
            .
);
```

Fig. 4.7: Estrutura da Tabela de Função de Saída

Tem-se um arranjo tridimensional de apontadores de função. O primeiro nível é indexado pela dupla evento/estado, o segundo pelo teste de predicados e o terceiro corresponde ao número máximo de funções de saída.

As estruturas de dados TT e TFS são manipuladas exclusivamente pelos processos PGCs.

Em resumo, esta é uma estrutura clássica [23] [24] onde as tabelas descritas são uma representação de como "caminhar" executando funções, a partir de um ponto de partida (estado da conexão) e em função de variáveis de contorno (eventos, predicados,...). Essa estratégia além de compactar o tamanho do código, permite acionamentos eficientes que deve repercutir no desempenho do protocolo.

4.3 DETALHAMENTO DOS ELEMENTOS DO MODELO FUNCIONAL

Uma vez apresentado o modelo funcional da implementação, descrevendo como os processos interagem entre si, as interfaces com as camadas adjacentes e as estruturas de dados utilizadas; faz-se a seguir, um estudo mais detalhado dos elementos envolvidos neste modelo.

4.3.1 PROCESSO GERENTE DA INTERFACE E DA CAMADA - PGIC

O PGIC é o processo que concentra toda a parte da implementação que é dependente do sistema operacional, por exemplo: a criação e morte de processos, a comunicação entre eles e a sincronização.

O PGIC constrói todo o ambiente necessário para a execução do protocolo de transferência, acesso e gerenciamento de arquivo. Assim sendo, ele cria e gerencia as quatro filas de mensagens que fazem interface com as camadas adjacentes, cria também a fila que faz a passagem das mensagens para o PGC, gerencia o segmento de memória compartilhada e aloca/libera dinamicamente buffers necessários para identificação das primitivas de serviço.

As filas de mensagens são criadas através da chamada ao supervisor **msgget ( )** conforme descrição feita no capítulo 3.

As operações permitidas para as filas são as seguintes:

- **Filas de Entrada Superior e Inferior** : permissão de leitura para PGIC.
- **Filas de Saída Superior e Inferior** : permissão de escrita para PGC.
- **Fila de Comunicação entre PGIC e PGC** : permissão de escrita para o PGIC e de leitura para o PGC.

O acesso às mensagens é feito de acordo com a sua ordem de chegada, isto é, a primeira mensagem enviada é a primeira a ser recebida. Corresponde então a uma disciplina FIFO - "First-In First-Out".

Conforme indicado no capítulo 3, a estrutura de dados da mensagem é a seguinte em linguagem C (Figura 4.8).

```
#define  TAM_MSG   1020     /* tamanho da mensagem */

struct  msgbuf {
        long mtype;            /* tipo da mensagem */
        char mtext [TAM_MSG]; /* texto da mensagem */
```

Fig. 4.8: Estrutura de Dados da Mensagem

O tamanho da mensagem foi calculado baseando-se nos seguintes argumentos:

- a maior primitiva ou UDP suportada pelo FTAM tem um tamanho de 550 octetos;
- o número de mensagens na fila necessário à execução do protocolo sem comprometer o seu desempenho, estima-se que está compreendido entre

$$3 < \text{NUM MENS.NA FILA} < 8$$

. o número máximo de octetos permitidos na fila, na configuração do sistema operacional UNIX utilizado, é 8.192 octetos.

Fazendo-se:

X = NÚMERO MÁXIMO DE OCTETOS NA FILA

Y = NÚMERO MÁXIMO DE MENSAGENS NA FILA

Tem-se:

TAMANHO DA MENSAGEM = X/Y = 8.192/8

= 1.024 octetos/mensagem

Como na estrutura da mensagem existe um campo "tipo da mensagem" que é um long, precisa-se separar , no nosso caso, quatro octetos para armazenar o valor deste campo.

Assim sendo, definiu-se que cada mensagem teria um tamanho máximo de 1020 octetos, que além de comportar o tamanho máximo das primitivas e das UDPs, permite um número de oito mensagens por fila, o que é razoável dado a sequencialidade da máquina de estado.

O funcionamento do processo PGIC é descrito a seguir, de acordo com o fluxograma apresentado na figura 4.9.

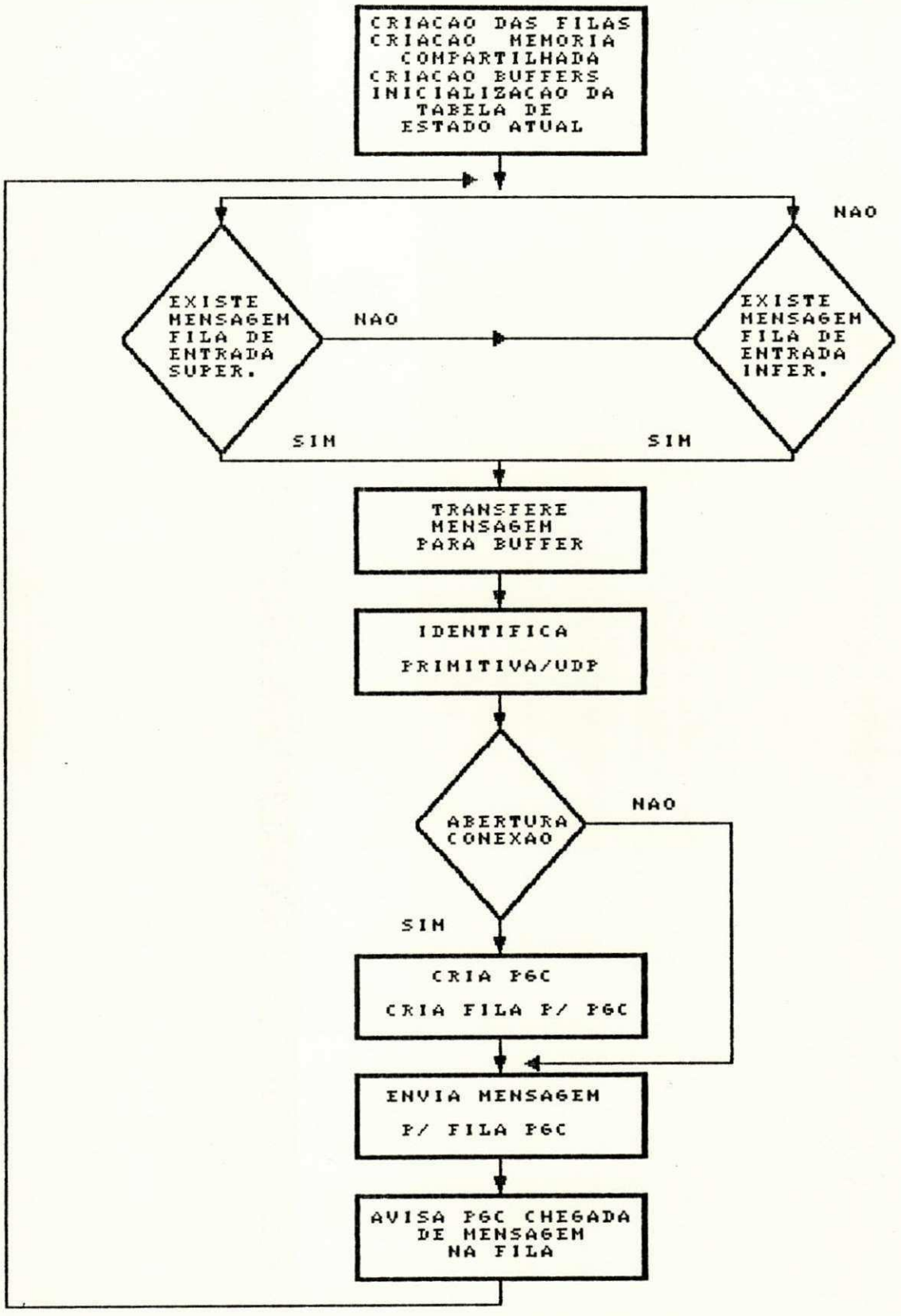


FIGURA 4.9 : FLUXOGRAMA GERAL DO PROCESSO PGIC.

As mensagens lidas pelo PGIC, através da chamada ao supervisor msgrcv ( ), são colocadas num buffer intermediário para que seja identificado que primitiva está sendo recebida e para quem se destina. Esse buffer é alocado dinamicamente e o seu tamanho é o de uma estrutura de mensagem.

Caso a primitiva seja uma solicitação para abertura de conexão, o PGIC cria através das chamadas ao supervisor fork ( ) e execl ( ), um processo PGC que será a partir deste momento, o responsável pela gerência da conexão a ser estabelecida, até que a mesma seja desfeita. Em seguida, cria uma fila de mensagem que será a via de comunicação entre o PGIC e PGC criado, por onde passarão as primitivas dirigidas àquela conexão.

Além da comunicação via fila de mensagem, o PGIC e PGC também se comunicam através de um segmento de memória compartilhada. Nesse segmento é inserida a tabela de estado atual que contém informações de estado e de controle da conexão. Por isso, após criar o processo gerente da conexão (PGC) e a sua fila correspondente, o PGIC colocará nesse segmento, informações de controle tais como:

- identificação do processo PGC criado;
- fila de mensagem do processo PGC criado;
- identificação das filas de saída da interface entre camadas,
- identificação da associação, etc...

A mensagem é então enviada para a fila do PGC através da chamada ao supervisor **msgsnd ( )**. O PGIC avisa o PGC através de um mecanismo de sinais via tabela de estado atual, da existência de mensagem na fila para ser tratada por ele.

Se a primitiva recebida não é uma solicitação de conexão, significa que esta já foi estabelecida. Nesse caso, o PGIC efetua os seguintes procedimentos:

. identifica para qual processo PGC se destina aquela mensagem;
. identifica qual a fila do processo PGC;
. envia a mensagem para a fila identificada e
. sinaliza o PGC da chegada de um evento.

O segmento de memória compartilhada com sua estrutura de dados associada são criados com a chamada **shmget ( )**. Conforme definido no capítulo 3, os parâmetros necessários à essa chamada são os seguintes:

. chave que representa o nome do segmento;
. tamanho do segmento e
. flag que define a forma de criação e o modo de acesso do segmento de memória compartilhada.

O segmento foi criado com permissões de leitura e escrita para os processos PGC e PGIC. Nesta fase da implementação, o tamanho definido para esse segmento comporta no máximo cinco processos PGCs ativos simultaneamente.

Os valores máximos para os recursos em questão, configurados no sistema operacional UNIX utilizado, são:

. Tamanho máximo do segmento de memória compartilhada: 20.400 octetos.
. Número máximo de processos ativos simultaneamente: 100 processos.

Portanto, como a estrutura de dados e de processo é bastante flexível, o tamanho do segmento pode ser aumentado para suportar mais processos ativos, até que se atinja um valor limite onde não se comprometa o desempenho do sistema e nem se alcance um tamanho maior que o máximo imposto pelo sistema operacional.

Neste trabalho não foi feita uma avaliação do valor limite que possa vir a comprometer o tempo de resposta dos usuários do sistema.

De certa forma, a limitação do número de conexões ativas em função da própria limitação dos recursos do sistema operacional, é uma das desvantagens da estrutura de processos utilizada. Mas, é importante observar que, normalmente, esses recursos

são parametrizáveis, podendo, portanto, serem ajustados.

Para inserir a tabela de estado atual no segmento de memória compartilhada criado, utiliza-se a chamada ao supervisor shmat ( ). Essa chamada vincula a tabela de estado atual a um endereço no espaço de endereçamento lógico do processo PGIC. Esse endereço pode ser selecionado pelo sistema ou calculado pelo processo do usuário. No caso desta implementação, o cálculo do endereço é feito pelo processo PGIC, da seguinte forma:

```
ender = sbrk (0) + MAXMEM + SHMLBA - 1;
```

onde:

MAXMEM = quantidade de octeto que deve ser somada ao endereço retornado por sbrk( ) a fim de reservar memória para a função malloc( ).

SHMLBA = endereço do limite inferior do segmento de memória compartilhada.

A função sbrk ( ) é usada para mudar dinamicamente a quantidade de espaço alocada para o segmento de dados de um processo [25]. A mudança é feita reinicializando-se o "valor limite"

do processo. O "valor limite" é o endereço da primeira posição após o fim do segmento de dados. A quantidade de espaço alocada cresce à medida que o valor limite cresce.

Como o valor do incremento utilizado na sbrk ( ) foi zero, fez-se com que o endereço apontasse para a primeira posição após o espaço alocado para o segmento de dados. Isso foi feito porque a função que faz alocação dinâmica de memória (malloc ( )) trabalha a partir desse endereço. Assim sendo, precisa-se deixar um espaço a ser utilizado pela malloc ( ). Esse espaço é um valor em octetos definido para MAXMEM, que será adicionado ao endereço retornado por sbrk (0).

SHMLBA é um valor definido pelo sistema, que representa o endereço do limite inferior do segmento. Esse valor também é adicionado no cálculo do endereço e diminuído de 1 para ajuste.

A chamada shmat ( ) retorna o endereço inicial do segmento de memória compartilhada. Portanto, de posse do endereço final do segmento, calculado acima, a shmat ( ) devolve o endereço inicial do segmento para o PGIC e internamente para o próprio sistema operacional.

A figura 4.10 ilustra o cálculo do endereço de vinculação.

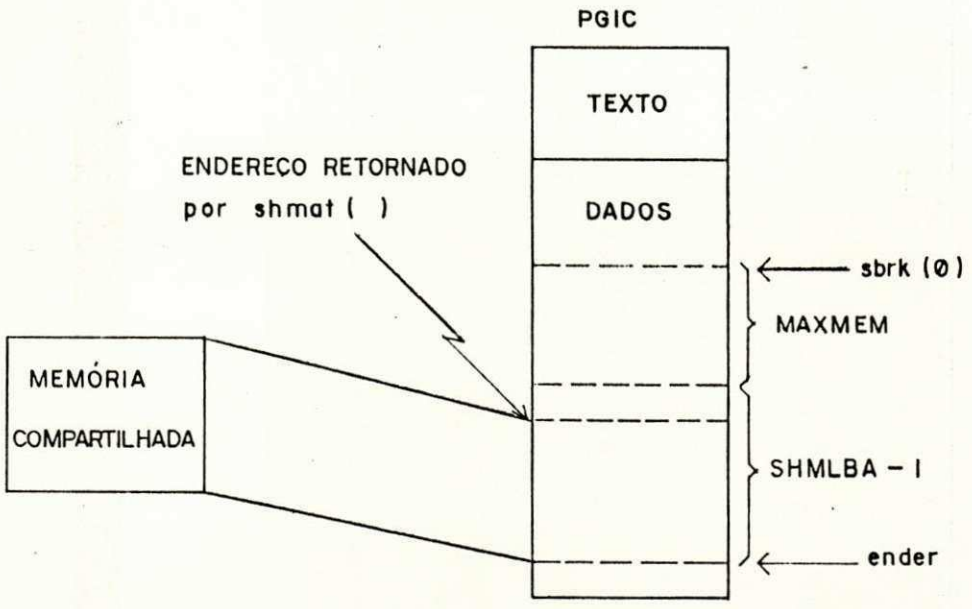


Fig. 4.10 : Vinculação do Segmento de Memória Compartilhada

O processo PGC também fará a vinculação do segmento de memória compartilhada ao seu espaço de endereçamento lógico conforme mostrado adiante na seção 4.3.2.

No processo PGIC existem situações que provocam erros setoriais e erros fatais. Quando um deles ocorre, uma mensagem de erro é enviada aos usuários.

Os erros setoriais do PGIC são aqueles que impedem a execução de um determinado processo PGC, mas não repercutem na execução dos outros processos PGCs ativos. Eles ocorrem quando:

. não existe área livre no segmento de memória compartilhada para criação do processo PGC;
. existe área livre no segmento de memória compartilhada, mas o PGIC não consegue criar o PGC;
. o PGC foi criado, mas não pode executar;
. o PGIC não pôde criar a fila de mensagem que faz a comunicação com o PGC;
. a fila foi criada, mas o PGIC não consegue enviar a mensagem recebida para a fila do PGC.

Os erros fatais são aqueles que provocam a liberação de todos os recursos alocados e termina a execução do PGIC. Eles ocorrem quando:

. a alocação dinâmica de memória não é bem sucedida;
. uma das filas que fazem a interface com as camadas adjacentes não é criada;
. o segmento de memória compartilhada não é criado ou não é vinculado ao PGIC e
. a mensagem contida nas filas de entrada superior ou inferior não pôde ser lida.

4.3.2 PROCESSO GERENTE DE CONEXÃO - PGC

O PGC é o responsável pela implementação da máquina do protocolo FTAM. Para cada usuário ativo existe uma instância do processo PGC responsável por esta conexão. Ele é ativado pelo processo PGIC através das chamadas ao supervisor **fork ( )** e **execl ( )**.

Ao ser ativado, o PGC identifica o segmento de memória compartilhada (**shmget ( )**) criado pelo PGIC e faz a vinculação deste segmento a um endereço no seu espaço de endereçamento lógico (**shmat ( )**). O cálculo desse endereço é feito da mesma forma apresentada no processo PGIC da seção anterior. Assim sendo, o PGC também associa a tabela de estado atual ao segmento de memória compartilhada.

Para cada processo PGC ativo existe uma entrada na tabela de estado atual, contendo todas as informações referentes a este processo:

- estado da conexão;
- semáforos e
- parâmetros necessários para sincronização e comunicação do PGIC com o PGC.

Portanto, o PGC precisa identificar qual entrada da tabela se refere à sua conexão para buscar informações tais como: qual a sua fila, se existe mensagem nesta fila, quais as filas de saída para onde ele deve enviar a mensagem depois de processada, o estado da conexão, etc....

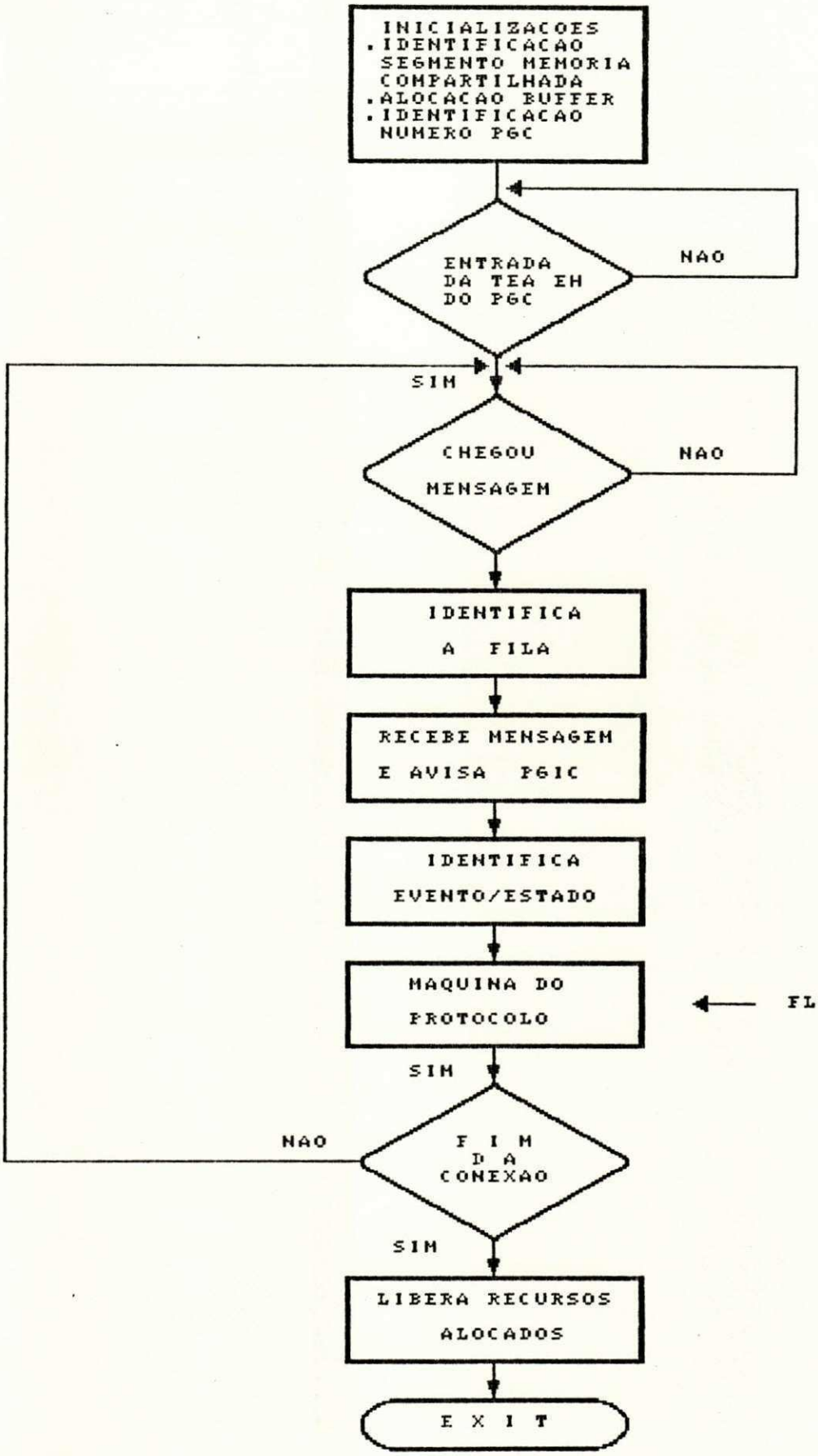
As figuras 4.11 e 4.12 mostram os fluxogramas do processo PGC.

Ao receber o aviso do PGIC da chegada de mensagem, o PGC recebe a primitiva, avisa o PGIC que já retirou a mensagem da fila, identifica-a e verifica o estado da conexão.

De posse dos parâmetros evento e estado, o PGC aciona a máquina do protocolo FTAM. Essa máquina é implementada através das tabelas de transição e função de saída, cujo acesso é feito apenas pelo PGC.

A função que executa a máquina de protocolo, primeiramente faz uma entrada na tabela de transição com os parâmetros evento e estado da conexão. Como foi visto na seção 4.2.4 para cada dupla evento/estado existe um arranjo de apontadores de funções de teste de predicado e um inteiro que indexa uma entrada na tabela de função de saída.

O teste de predicado é executado e retorna um valor que será um outro indexador na tabela de função de saída.



← FLUXOGRAMA FIG. 4.12

JRA 4.11: FLUXOGRAMA GERAL SIMPLICADO DO PROCESSO GERENTE DE CONEXAO-PGC

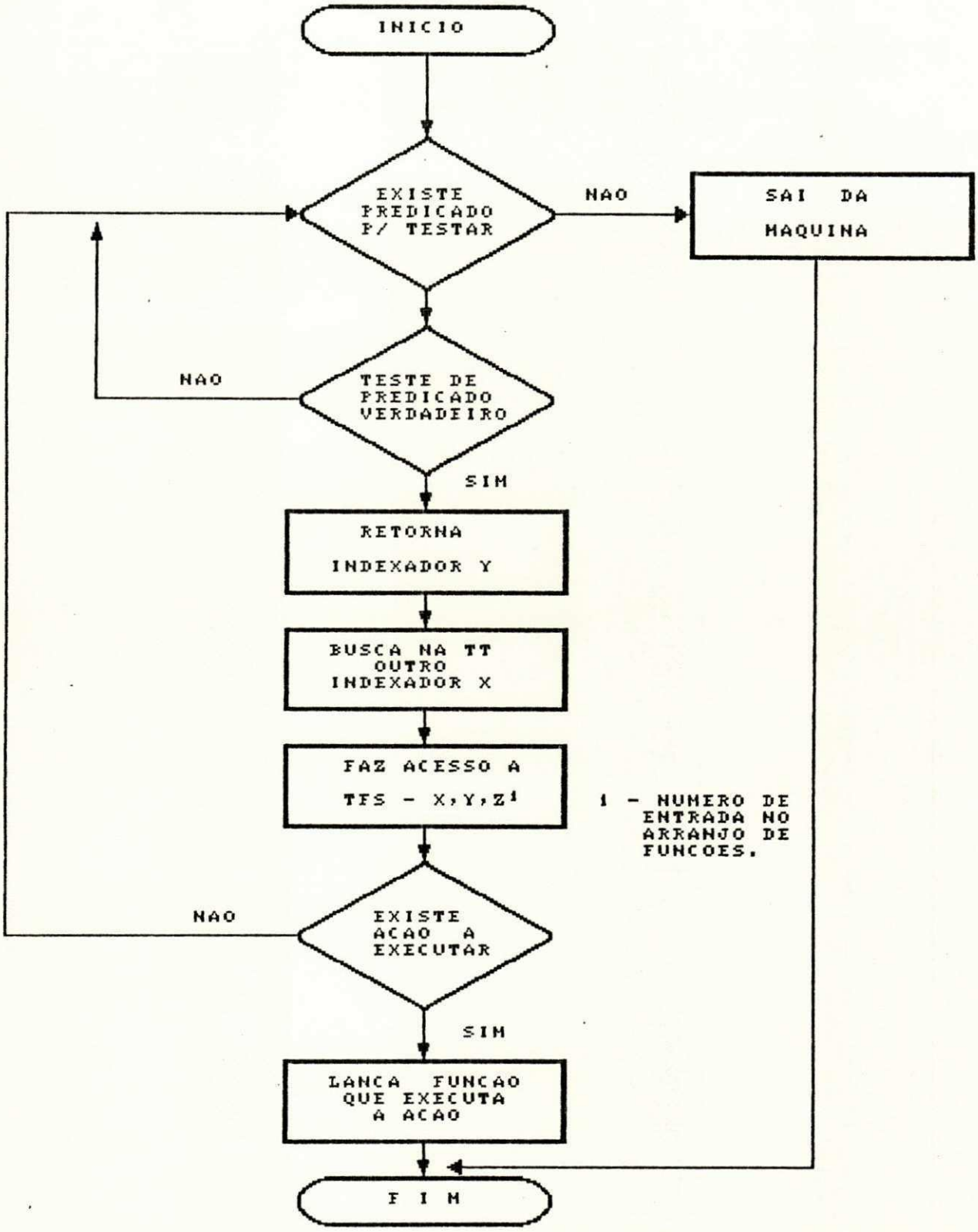


FIGURA 4.12:FLUXOGRAMA DA MAQUINA DE PROTOCOLO DO PPROCESSO PGC

Na tabela de função de saída existe um arranjo de arranjos de apontadores de funções para cada dupla estado/evento. A seleção de qual função deve ser acionada é feita exatamente por esses dois indexadores encontrados através da tabela de transição e um terceiro indexador que corresponde ao número de funções no arranjo.

As ações para a transição ocorrida são executadas e a primitiva com a UDP construída é enviada pelo PGC para as filas de saída da interface entre camadas. Se for uma primitiva de confirmação esta é enviada para a fila de saída superior e se for uma UDP esta é enviada para a fila de saída inferior, numa primitiva do CASE ou Apresentação.

Enviada a mensagem para as filas adequadas, o PGC atualiza o estado da conexão na tabela de estado atual e fica num estado esperando mensagem até que receba um sinal do PGIC da chegada de um novo evento na fila.

Quando a conexão estabelecida é encerrada, o PGC libera a sua entrada no segmento de memória compartilhada (tabela de estado atual) e termina sua execução.

As situações que provocam erro no processo PGC somente repercutem na conexão gerenciada por ele. Os outros processos PGCs ativos continuam executando normalmente.

O PGC envia uma mensagem de erro para o usuário e termina sua execução, caso aconteça uma das seguintes situações:

. a alocação dinâmica de memória não foi bem sucedida;
. não consegue ler a mensagem contida na sua fila;
. não identifica o segmento de memória compartilhada;
. não consegue vincular o segmento de memória compartilhada ao seu espaço de endereçamento lógico ou
. diagnósticos fatais das ações do protocolo.

O anexo A mostra um exemplo do funcionamento do protocolo para o estabelecimento de conexão FTAM, implementado pelos procesos PGIC e PGC.

4.3.3 TABELAS - INICIALIZAÇÃO E FLEXIBILIDADE

O tipo de estruturação utilizada nas tabelas permite a implantação gradual das classes de servicos do FTAM, sem modificações na implementação das classes já desenvolvidas e instaladas.

Para se entender o que deve ser feito para a anexação de uma nova classe de serviço, é importante que se conheça como foram inicializadas as tabelas de transição e função de saída.

Para cada classe de serviço do FTAM existe um conjunto de funções associadas a ela. Todas essas funções estão inicializadas como apontadores de funções nas tabelas de transição e função de saída.

Na tabela de transição (TT) estão inicializadas todas as funções que testam os predicados para cada combinação evento ocorrido/estado da conexão. Na tabela de função de saída (TFS) estão inicializadas todas as funções responsáveis pela execução das ações de saída pertinentes à transição ocorrida.

Essas tabelas foram inicializadas de acordo com as Tabelas de Estado apresentadas no documento de especificação do protocolo FTAM [8].

Cada entrada na Tabela de Estado especifica o evento de saída apropriado (juntamente com alguma ação específica), e o novo estado do autômato para a combinação evento de chegada/estado atual. Somente combinações válidas têm uma entrada na tabela, todas as outras combinações são deixadas em branco e tratadas como um erro de protocolo.

O formato das Tabelas de Estado especificadas no padrão FTAM é mostrado na figura 4.13.

ESTADO / EVENTO	E_1	E_2	E_3	...	...	E_n
EV_1		1				
EV_2						
EV_3			2			
⋮						
EVn						

Onde:

1: P_1: ESx, AEx $\Longrightarrow$ Ex

P_2: ESy, AEy $\Longrightarrow$ Ey

2: ESz, AEz $\Longrightarrow$ Ez

$P_{1,2}$: predicado a ser testado

ESx,y,z : evento de saída x, y, z

AEx,y,z : ação específica x, y, z

Ex,y,z : estado x, y, z

OBS: Existem entradas que não necessitam do teste de predicado

Fig. 4.13 : Formato das Tabelas de Estado do FTAM

Baseado no exemplo da estrutura da Tabela de Estado apresentada na figura 4.13, mostra-se a seguir como é feito o mapeamento desta estrutura para a inicialização das tabelas TT e TFS.

Na figura 4.14 apresenta-se a inicialização da tabela de transição para o exemplo sugerido.

```
struct t_tab_trans {
        int ( * f_pred [N]) ( );  /*arranjo de apontadores de
                                    funções retornando um in-
                                    teiro */
        int tab_fsd;              /*número de entrada na TFS*/
) tab_trs [ ] [EST] = {
  {{tp1,tp2,NULL},1},
  {{s_pred,NULL},2}
    .
    .
    .
};
```

Fig. 4.14: Inicialização da Tabela de Transição

onde:

N : número máximo de apontadores de funções no arranjo

EST: número máximo de estados do protocolo FTAM

tp1 : apontador de função que testa P1

tp2 : apontador de função que testa P2

s_pred : apontador de função sem predicado

A inicialização da Tabela de Função de Saída para o exemplo apresentado é mostrada na figura 4.15:

```
int ( * f_ac [ ] [X] [Y]) ( ) = {
    { (NULL), (err_ptcl,NULL) },
    { (NULL), (ESx,AEx,Ex,NULL), (ESy,AEy,Ey,NULL) },
    { (NULL), (ESz,AEz,Ez,NULL) }
};
onde: X = nº de arranjos de apontadores de funções
      Y = nº de apontadores de funções
```

Fig. 4.15: Inicialização da Tabela de Função de Saída

O interrelacionamento das tabelas TT e TFS se dá da seguinte forma:

- De posse dos parâmetros evento e estado, encontra-se na TT a entrada correspondente a esta combinação. Por exemplo, a entrada (0,0) (Figura 4.14).
- Nessa entrada, o apontador de função tp1 aciona a função que testa o predicado P1.
- Se P1 for verdadeiro, a função retornará um inteiro que corresponde ao número de indexação na listagem de procedimentos da TFS. Neste caso, retorna 1.

. Na tabela TT encontra-se o outro parâmetro de indexação da TFS. Neste exemplo, na entrada (0,0) tem-se o valor 1.

. Entra-se então na TFS com os indexadores (1,1) e um terceiro indexador que corresponde ao número de ações no array. Logo, as entradas serão (1,1,0), (1,1,1), (1,1,2) e (1,1,3) que correspondem exatamente às ações (ESx, AEx, Ex, NULL).

. Se P1 for falso, retornará 0. Portanto, a entrada na TFS seria (1,0,0) onde encontraria-se o NULL. Neste caso testa-se então P2 que retornará o valor 2.

. Entra-se na TFS com os indexadores (1,2,0), (1,2,1), (1,2,2) e (1,2,3) que corresponde às ações (ESy, AEy, Ey, NULL).

Esses procedimentos foram apresentados no fluxograma da máquina de protocolo na figura 4.12.

Esta estruturação das tabelas permite que, de acordo com as necessidades do usuário, se vá implementando gradativamente as funções relacionadas com qualquer classe de serviço do protocolo FTAM. Na realidade, as estruturas de dados foram dimensionadas para suportar a implementação do pacote FTAM completo.

Esta estratégia permite também, uma busca e acionamentos eficientes, associados a uma inicialização relativamente simples através da definição de valores de apontadores em tabelas.

Em resumo, como as tabelas de transição e função de saída foram inicializadas e estruturadas para suportar qualquer classe de serviço do FTAM, para a anexação de uma nova classe deve-se fazer o seguinte:

. selecionar as funções referentes à classe de serviço escolhida e
. implementar as funções selecionadas.

Como pode-se observar, este método torna a implementação intrínsecamente modular, o que diminui a complexidade de desenvolvimento das classes de serviço do protocolo FTAM.

4.3.4 DEFINIÇÃO DAS PRIMITIVAS E MAPEAMENTO DAS UDPs EM ASN1

Em relação às estruturas de dados das primitivas de serviço, seguiu-se as recomendações ISO contidas no documento do padrão FTAM. A figura 4.16 mostra um exemplo da estrutura de dados da primitiva F-INITIALIZE request .

```
/*
* Primitiva F-INITIALIZE-request
*/
   struct t_in_rq {

            int  id_serv;              /* identificador da primitiva de serviço */

            char t_apl_ch [APL];       /* título da aplicação chamada */

            char t_apl_ct [APL];       /* título da aplicação chamante */

            int  e_ap_ch;              /* endereço de apresentação chamado */

            int  e_ap_ct;              /* endereço de apresentação chamante */

            BOOLEAN g_cont_ap;         /* gerência do contexto de apresentação */

            int  niv_serv;             /* nível de serviço */

            int  clas_serv;            /* classe de serviço */

            BITSTRING und_func;        /* unidades funcionais */

            BITSTRING gp_atb;          /* grupo de atributos */

            int  rbck;                 /* disponibilidade de retrocesso */

            int  q_s_com;              /* qualidade do serviço de comunicação */

            struct t_lst_cont lst_cont; /* lista do tipo de conteúdo */

            G_STRING id_i;             /* identidade do iniciador */

            G_STRING cont;             /* contabilidade */

            union t_senha senh;        /* senha do sistema de arquivo * /

            int  chkp;                 /* janela de checkpoint */

};
```

Fig, 4.16: Exemplo da Estrutura de Dados da Primitiva F-INITIALIZE request

As estruturas de dados das UDPs foram definidas à partir da descrição em ASN.1 ("Abstract Syntax Notation One") também contidas no documento padrão FTAM. As UDPs especificadas em ASN.1 foram mapeadas manualmente (sem ferramentas de tradução) em estruturas de dados em linguagem "C".

Em ASN.1 o princípio adotado para definir os tipos de dados associados com as variáveis é o mesmo usado pela maioria das linguagens de programação de alto nível. Cada variável é declarada e o tipo de dado associado com ela é também definido.

Os tipos de dados suportados pela sintaxe ASN.1 são os seguintes:

1) Tipos Simples
   - BOOLEAN
   - INTEGER
   - BITSTRING
   - OCTETSTRING
   - NULL
   - GraphicString
   - GeneralizedTime
   - EXTERNAL
   - OBJECT IDENTIFIER
   - ObjectDescriptor

2) Tipos Estruturados

. SEQUENCE

. SEQUENCE OF

. SET

. SET OF

. CHOICE

. Tagged

Os tipos suportados em ASN.1 podem ser membros de quatro classes:

. UNIVERSAL: estes são tipos gerais, tal como o inteiro.

. CONTEXT-SPECIFIC: estão relacionados ao contexto específico no qual estão sendo usados.

. APPLICATION: são comuns a uma entidade de aplicação.

. PRIVATE: estes são definidos pelo usuário.

As palavras chaves usadas em ASN.1 estão sempre escritas em letras maiúsculas.

ASN.1 suporta o conceito de atribuir um identificador ("tag") para um tipo de dado existente. Usa-se o "tag", portanto, quando um tipo de dado particular deve ser distinguível de outro

tipo de dado similar. Quando uma variável é "etiquetada", o número do "tag" é inserido nos colchetes depois do nome da variável.

Outra facilidade suportada em ASN.1 é a possibilidade de se declarar uma variável como implícita. Isso é feito por meio da palavra IMPLICIT que é escrita depois do nome da variável. O número do "tag" deve estar presente e escrito antes do tipo do identificador. Esse conceito tornar-se-á mais claro quando for apresentado mais tarde um exemplo de uma UDP codificada em ASN.1.

O conjunto completo de UDPs relacionadas com uma entidade de protocolo particular é definido como um modulo. O nome de um modulo é conhecido como "**module definition**" e, no exemplo da figura 4.17, este é dado como **ISOFTAM-FTAM DEFINITIONS**. Este é seguido pelo símbolo de atribuição (::=) e o corpo do módulo é então definido entre as palavras chaves **BEGIN** e **END**.

Seguindo BEGIN, o tipo CHOICE é usado para indicar que as UDPs usadas no FTAM pertencem a um dos tres tipos: FTAM-Regime-PDU, File-PDU, Bulk-Data-PDU. Um tipo CHOICE adicional é então usado para indicar que existem seis tipos diferentes de UDPs associadas com o tipo FTAM-Regime-PDU: F-INITIALIZE-request, F-INITIALIZE-response e assim por diante. Cada uma delas são identificadas com um "tag" para que possam subsequencialmente serem distinguidas uma das outras. Os "tags" são seguidos por IMPLICIT, isto significa que nenhuma definição adicional é necessária, tal como o tipo da UDP.

O tipo estruturado **SEQUENCE** (similar ao struct em "C") é usado nesta definição para indicar que a UDP contém um número limitado de elementos de dados que podem ser simples ou estruturados. O primeiro elemento "**protocol-id**" é definido como um SEQUENCE de dois tipos (BITSTRING/INTEGER). O próximo elemento "**presentation-context-management**" é do tipo BOOLEAN. Os campos "**service-level**", "**service-class**" e "**rollback-available**" são definidos como INTEGER e os valores possíveis de cada são mostrados nos parênteses. Os elementos "**functional-units**" e "**atribute-groups**" são do tipo BITSTRING. Alguns elementos na sequência são declarados como OPTIONAL, podendo ou não estarem presentes na UDP codificada.

```
ISO8571 - FTAM DEFINITIONS ::=
BEGIN
PDU ::= CHOICE {
                FTAM-Regime-PDU,
                File-PDU,
                Bulk-Data-PDU
               }
FTAM-Regime-PDU ::= CHOICE {
                    [0]  IMPLICIT F_INITIALIZE_request,
                    [1]  IMPLICIT F_INITIALIZE_response,
                    [2]  IMPLICIT F_TERMINATE_request,
                    [3]  IMPLICIT F_TERMINATE_response,
                    [4]  IMPLICIT F_U_ABORT_request,
                    [5]  IMPLICIT F_P_ABORT_response
```

Fig. 4.17: Codificação da UDP F-INITIALIZE request em ASN.1

```
F_INITIALIZE_request ::= SEQUENCE {
  protocol-id              Protocol-Version
       DEFAULT             {major {version-1}, minor revision-1},
  presentation-context-management [1] IMPLICIT BOOLEAN
       DEFAULT             FALSE,
  service-level            Service-Level
       DEFAULT             reliable,
  service-class            Service-Class
       DEFAULT             transfer-class,
  functional-units         Functional-Units  OPTIONAL,
  attribute-groups         Attribute-Groups
       DEFAULT             { },
  rollback-availability    Rollback-Availability OPTIONAL,
  contents-type-list       Contents-Type-List    OPTIONAL,
  initiator-identity       User-Identity         OPTIONAL,
  account                  Account               OPTIONAL,
  filestore-password       Password              OPTIONAL,
  checkpoint-window [8]    IMPLICIT INTEGER
      DEFAULT          1
}
```

continuação Fig. 4.17

```
Protocol-Version ::= [0] IMPLICIT SEQUENCE {
   major [0] IMPLICIT BITSTRING  {version-1   (0)},
   minor [1] IMPLICIT INTEGER    {revision-1  (1)},
             DEFAULT  revision-1

Service-Level   ::= [2] IMPLICIT INTEGER
                        {reliable (0), user-correctable (1)}

Service-Class   ::= [3] IMPLICIT INTEGER
                        {transfer-class                 (0),
                         access-class                   (1),
                         management-class               (2),
                         transfer-end-management-class  (3),
                         unconstrained-class            (4) }

Functional-Units ::= [4] IMPLICIT BITSTRING
                         {read                          (2),
                          write                         (3),
                          file-access                   (4),
                          limited-file-management       (5),
                          enhanced-file-management      (6),
                          grouping                      (7),
                          recovery                      (8),
                          restart-data-transfer         (9) }

Attribute-Groups  ::= [5] IMPLICIT BITSTRING
                          {storage    (0),
                           security   (1),
                           private    (2) }

Rollback-Availability ::= [6] IMPLICIT INTEGER
                           {no-rollback           (0),
                            rollback-availability (1)}

Contents-Type-List ::= [7] IMPLICIT SEQUENCE {
  document-types [0] IMPLICIT SEQUENCE OF Document-Type-Name,
  constraint-sets-and-abstract-syntaxes [1] IMPLICIT SEQUENCE {
  constraint-sets [0] IMPLICIT SEQUENCE OF Constraint-Set-Name,
  abstract-syntaxes [1] IMPLICIT SEQUENCE OF
                        Abstract-Syntax-Name }
    }
```

continuação Fig. 4.17

```
User-Identity ::= [APPLICATION 4] IMPLICIT GraphicString

Account ::= [APPLICATION 5] IMPLICIT GraphicString

Password ::= [APPLICATION 6] CHOICE
                              (GraphicString, OCTETSTRING)

Document-Type-Name ::= [APPLICATION 7]
                              IMPLICIT OBJECT IDENTIFIER

Constraint-Set-Name ::= [APPLICATION 8]
                              IMPLICIT OBJECT IDENTIFIER

Abstract-Syntax-Name ::= [APPLICATION 9]
                              IMPLICIT OBJECT IDENTIFIER
```

continuação Fig. 4.17

Na figura 4.18 mostra-se a UDP F-INITIALIZE request em linguagem "C", codificada a partir da definição em ASN.1.

```
#define    TEX    85

typedef    int    BITSTRING;
typedef    char   BOOLEAN;
typedef    char   G_STRING [TEX];
typedef    char   O_STRING;

/*
 * UDP F-INITIALIZE request
 */

struct t_lst_cont {
      G_STRING  n_t_dc;     /* nome do tipo do documento */
      struct {
        G_STRING n_cj_r;   /* nome do conjunto de restrições */
        G_STRING n_st_a;   /* nome da sintaxe abstrata */
      } cj_stx;
};
struct t_versao {
      BITSTRING  p_vers;
      int p_rev;
};
union  t_senha {
      G_STRING  g_senh;
      O_STRING  o_senh;
};
struct t_in_req {
      int  id_pdu;                /* identificador da udp */
      struct t_versao versao;     /* versao do protocolo */
      BOOLEAN g_cont_ap;          /* gerência do contexto de    apresentação */
      int  niv_serv;              /* nível de serviço */
      int  clas_serv;             /* classe de serviço */
      BITSTRING und_func;         /* unidades funcionais */
      BITSTRING gp_atb;           /* grupo de atributos */
      int  rbck;                  /* retrocesso */
      struct t_lst_cont lst_cont; /* lista do tipo de conteúdo */
      G_STRING id_i;              /* identidade do iniciador */
      G_STRING cont;              /* contabilidade */
      union t_senha senh;         /* senha do sistema de arquivo */
      int  chkp;                  /* janela de checkpoint */
};
```

Fig. 4.18: Codificação da UDP F-INITIALIZE request em linguagem C.

Os tipos SEQUENCE e CHOICE em ASN.1 são mapeados para os tipos em "C" "struct" e "union", respectivamente.

O tipo BITSTRING é definido em ASN.1 como um string de oito bits cada um representando um elemento particular. Eles são ligados para 1 ou 0, dependendo se o elemento é ou não requerido. Como a capacidade de memória do computador utilizado na implementação era suficiente, não necessitava-se compactar vários objetos em um único octeto. Assim sendo, definiu-se que o tipo BITSTRING seria um int em "C", o que facilitaria a implementação, pois neste caso não há necessidade de se trabalhar com manipulação de bits.

Os tipos BOOLEAN e O_STRING foram definidos como caracter e o O_STRING como um array de caracteres.

Capítulo 5

COMPARAÇÃO DO MODELO DE IMPLEMENTAÇÃO COM UM MODELO ALTERNATIVO

5.1 INTRODUÇÃO

Uma arquitetura alternativa de implementação de um sistema de comunicação OSI é apresentada em Halsall [23]. Mostra-se, a seguir, um breve relato do modelo de Halsall e faz-se um comparativo com o modelo funcional do protocolo FTAM implementado.

5.2 MODELO DE HALSALL

O sistema de comunicação OSI é implementado no modelo de Halsall através de um conjunto de tarefas (processos), uma por camada de protocolo, com tarefas adicionais para execução de determinadas funções, como por exemplo, o gerenciamnto local.

Os processos se comunicam através de um conjunto de filas tipo FIFO ("First-in-First-out") ou "mailboxes", como mostra a figura 5.1. Essa comunicação entre os processos faz uso das facilidades de um núcleo tempo real, que executa também funções de escalonamento de tarefas e controle de interrupções.

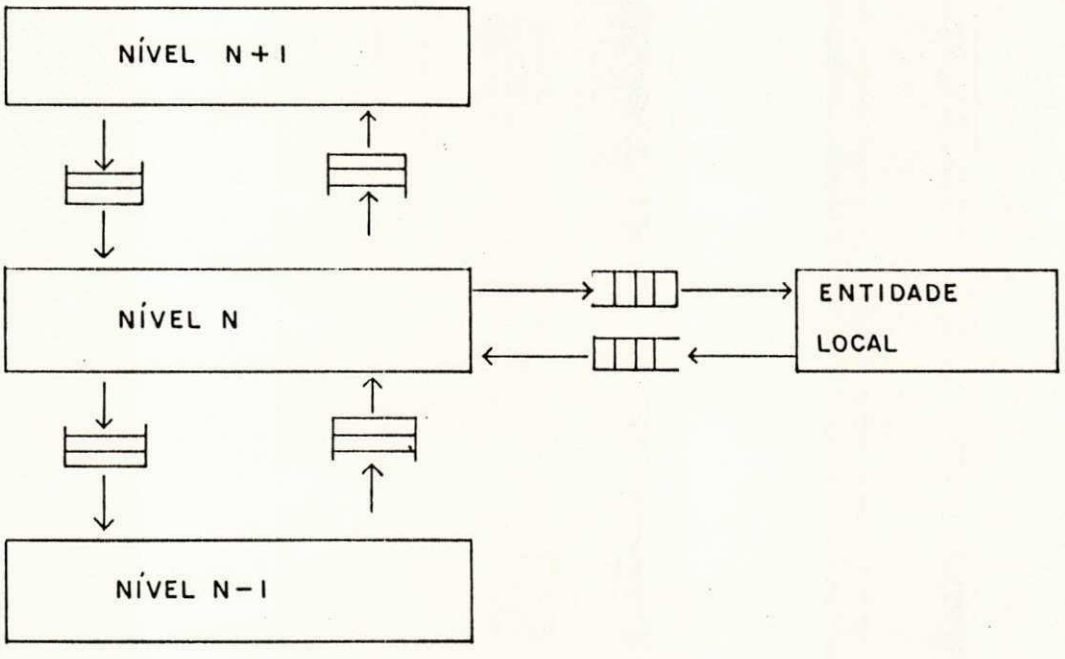


Fig. 5.1 : Comunicação Entre Processos no Modelo Halsall

Uma camada de protocolo comunica-se com outra camada adjacente por meio de primitivas de serviço associadas com cada camada. Cada primitiva, com seus parâmetros, é criada num formato definido (de domínio local), ocupando um "buffer" de memória denominado BCE (Bloco de Controle de Evento). A estrutura do BCE é dependente da estrutura de dados da linguagem de implementação de cada entidade de protocolo, uma vez que diferentemente da sintaxe rígida associada com a UDP, os parâmetros das primitivas de serviço estão na sintaxe local. Além disso o BCE é único para todas as primitivas associadas com cada camada de protocolo, possuindo um cabeçalho para identificação do tipo da primitiva.

A passagem de primitivas entre camadas (processos) é feita invocando-se primitivas de comunicação interprocessos para o núcleo local, tendo-se o endereço do apontador do BCE como parâmetro. Isso provoca a inserção, pelo núcleo, do apontador no final da fila de comunicação. Quando o processo receptor é escalonado para execução, ele examina suas filas de entrada (acima e abaixo) para verificar se há um BCE esperando processamento. Isto resulta em uma UDP sendo gerada e uma mensagem formatada sendo criada (BCE) e passada para uma de suas filas de saída.

O modelo discute também a necessidade de se definir uma fila específica para cada ponto de acesso de serviço (canal) que estiver ativo, uma vez que existem situações de protocolo que devem ser trabalhadas de forma e em tempos diferentes para cada

canal, cujo gerenciamento se torna complicado se houver uma fila única para todos os canais. Este mecanismo aumenta consideravelmente o número de filas à medida que muitos canais se tornam simultaneamente ativos, porém possibilita um controle eficiente do fluxo de mensagens em cada canal. Um aspecto que o modelo não discute é o gerenciamento dessas filas, feito pelo núcleo, com relação a ativação e desativação dos canais.

O modelo propõe que os dados associados com a primitiva sejam armazenados num "buffer" separado, denominado BDU (Buffer de Dados do Usuário). O BDU contém as UDPs acumuladas por cada uma das camadas superiores, constituindo, na camada mais inferior, o conjunto de "bytes" a ser transmitido pelo nível físico.

Cada BCE possui um campo apontador para a BDU, que é o ponteiro para o endereço que contém os dados de usuário associados com a primitiva, e um campo que define o tamanho da BDU.

O modelo declara os BCEs e suas BDUs associadas, como estruturas de dados globais, uma vez que estas devem ser acessíveis à todas as camadas. Um "pool" de BCEs e BDUs é criado para cada camada na inicialização do sistema e os apontadores para estes buffers são ligados na forma de uma lista. O núcleo deve controlar a utilização destes buffers (obtenção e devolução de e para a fila) e implicitamente conhecer a estrutura dos BCEs.

5.2.2 COMPARAÇÃO ENTRE OS MODELOS

Um primeiro ponto a se considerar é quanto a estruturação de processos adotada. No modelo de Halsall tem-se um processo por camada que gerencia todos os pontos finais de conexão. No FTAM implementado, conforme visto no capítulo 4, a estruturação dos processos é a seguinte:

- cada processo PGC gerencia um ponto final de conexão interligando os processos aplicativos através de uma instância do mesmo tipo, no caso, o FTAM e
- o processo gerente PGIC garante a instanciação correta e parametriza a comunicação com os PGCs.

A estrutura de processos adotada nesta implementação do FTAM garante o paralelismo da execução do protocolo pela disciplina de escalonamento de tarefas do núcleo, que distribui com eqüidade a capacidade de processamento disponível entre as conexões ativas. Isto, certamente irá repercutir positivamente no desempenho do protocolo como um todo.

Uma outra vantagem da estrutura utilizada é que pode-se eliminar a necessidade de múltiplas filas por ponto de conexão, pois existe o Processo Gerente da Interface e da Camada - PGIC que pode viabilizar a distribuição das mensagens para cada ponto de

conexão com a utilização de apenas uma fila. Desta forma, diminui-se o número de recursos alocados do sistema operacional.

Esta estrutura multiprocesso por camada, também viabiliza o tratamento prioritário dos pontos de conexão, se o protocolo o desejar e se o núcleo permitir uma disciplina de escalonamento com prioridades.

Uma desvantagem deste tipo de estruturação adotada no FTAM é quanto à limitação do número de conexões ativas em função da limitação do número de processos que o sistema operacional pode suportar. Mas é bom lembrar que este recurso é parametrizável, podendo, portanto, ser ajustado.

Quanto a interface de comunicação entre as camadas adjacentes, os dois modelos utilizam mecanismos iguais - filas de mensagens. O modelo de Halsall sugere que se defina uma fila específica para cada ponto de acesso a serviço mas não discute o gerenciamento destas filas, feito pelo núcleo. O modelo funcional do FTAM apresentado no capítulo 4, propõe quatro filas de mensagens (entrada e saída superior e inferior) que serão utilizadas para a passagem de primitivas e UDPs de todas as conexões ativas. O gerenciamento dessas filas é feito pelo processo gerente da interface e da camada - PGIC. Neste caso, o PGIC equivale funcionalmente à estrutura de múltiplas filas.

No modelo de Halsall, para todas as primitivas de serviço existe um único buffer (BCE) que está associado com cada camada de protocolo. Esse buffer contém um apontador para um BDU (Buffer de Dados do Usuário). No FTAM implementado existe um buffer para cada primitiva pois elas estão definidas separadamente, com sua área de dados também já definida na própria estrutura.

A alocação e liberação dos buffers necessários à execução do protocolo FTAM é feita por cada um dos processos (PGCs e PGIC) que os utilizam. No modelo Halsall o gerenciamento dos buffers é feito pelo núcleo.

Um gerenciamento global de memória não é de implementação evidente na maioria dos sistemas operacionais. Por exemplo, no UNIX, tal facilidade deveria ser feita com o uso de memória compartilhada, que apresentou dificuldades tanto no que diz respeito ao seu uso, quanto à sua própria implantação no sistema operacional do computador sendo utilizado.

A alocação de memória por processo (camada) é de implementação menos obscura. A dificuldade decorrente é que deve-se fazer a passagem de dados entre camadas via filas.

No Halsall todo o gerenciamento da comunicação (sincronização, escalonamento de tarefas, controle de interrupções....) é feito pelo núcleo. No FTAM desenvolvido estas tarefas são de responsabilidade do PGIC.

Dada a própria natureza de um sistema operacional multiusuário e multitarefa, estas funções devem ser necessariamente gerenciadas pelos processos em questão.

A hipótese de gerenciamento direto se aplica melhor nos casos em que a implementação é realizada num hardware específico (tipo Processador Frontal), onde tem-se controle da funcionalidade do núcleo instalado.

Em resumo, o modelo de implementação adotado neste trabalho propõe soluções que buscam a melhoria de desempenho e potencialidade do protocolo.

Capítulo 6

TESTES DA IMPLEMENTAÇÃO DO FTAM

6.1 INTRODUÇÃO

Neste capítulo são apresentados os testes efetuados sobre o elemento de serviço de aplicação SASE/FTAM. Nele, descreve-se a funcionalidade dos processos envolvidos e mostra-se um exemplo de uma sequência de testes.

6.2 FUNCIONALIDADE DOS PROCESSOS DE TESTE DO PROTOCOLO

Os testes aqui apresentados são de escopo limitado e têm como objetivo verificar a operacionalidade do FTAM implementado. O objetivo dos testes de operacionalidade é examinar como a máquina de protocolo implementada responde às primitivas de serviço e aos elementos de protocolo gerados, em condições normais ou na presença de erros, verificando-se as propriedades de repetitividade (dada uma entrada observa-se sempre a mesma saída) e a inexistência de impasses.

Esses testes foram feitos simulando-se duas máquinas interligadas através da execução de duas instâncias do processo PGIC numa única máquina. O primeiro PGIC ativado é considerado o Iniciador e o segundo é o Respondedor. Os testes foram executados para uma única associação de aplicação.

Os processos responsáveis pelos testes de execução do protocolo FTAM são os seguintes:

. PROCESSO DE TESTE SUPERIOR - PTS

. PROCESSO DE TESTE INFERIOR - PTI

As funções básicas do processo PTS são as seguintes (Figura 6.1):

. construção das primitivas de serviço;

. envio das primitivas às filas de entrada superiores dos processos PGIC iniciador e respondedor;

. recepção das primitivas nas filas de saída superiores dos processos PGIC iniciador e respondedor;

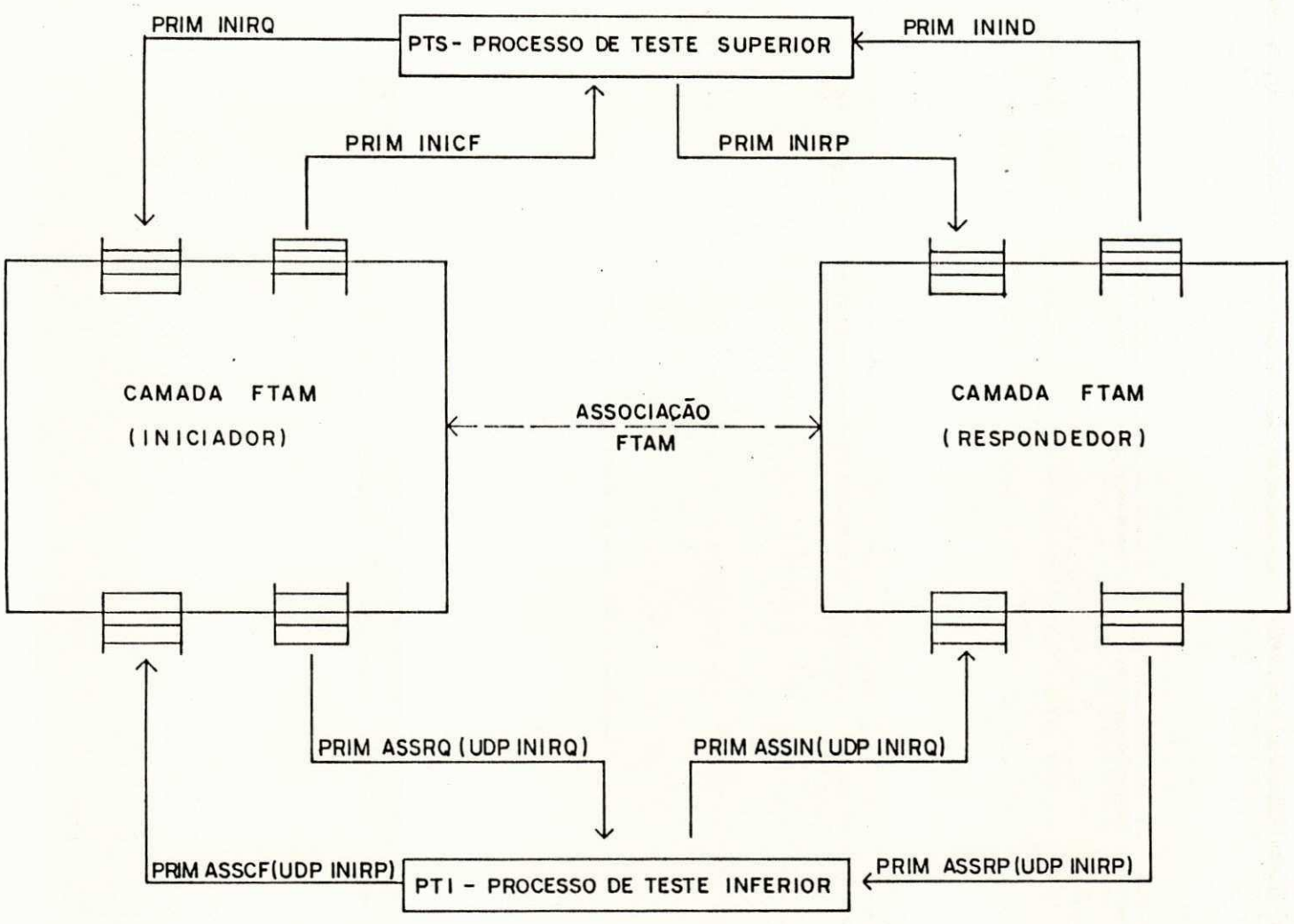
. emissão de mensagem ao usuário, baseada nos eventos ocorridos na sequência de testes.

O processo PTI é responsável pelo seguinte (Figura 6.1):

. recepção, nas filas de saída inferiores dos PGICs iniciador e respondedor, das primitivas request e response do CASE ou da camada de Apresentação, contendo as UDPs construídas pelo protocolo FTAM.

. construção das primitivas "indication" e "confirm" do CASE ou da camada de Apresentação;

. envio das primitivas construídas às filas de entrada inferiores dos PGICs iniciador e respondedor.



FTAM

INIRQ – Initialize - Request
INIRP – Initialize - Response
ININD – Initialize - Indication
INICF – Initialize - Confirm

CASE

ASSRQ – Associate Request
ASSRP – Associate Response
ASSIN – Associate Indication
ASSCF – Associate Confirm

Fig. 6.1 : Esquema Funcional dos Processos de Testes do Protocolo FTAM-Trocas de Primitivas e UDPs no Estabelecimentos de Conexão FTAM

Quanto às propriedades de repetitividade, o protocolo comportou-se conforme o esperado, isto é, para uma mesma primitiva ou UDP recebida observava-se sempre a mesma saída.

As condições de erro testadas relacionam-se ao envio de primitivas e UDPs com erro ou primitivas e UDPs que não devem ser recebidas para o estado em que se encontra a conexão. Nesses casos, o protocolo ao verificar a inconsistência das primitivas ou UDPs envia um diagnóstico para o processo de teste superior. Baseado no diagnóstico recebido, o PTS emite uma mensagem de erro para o usuário.

Dado à sequencialidade do protocolo implementado, não se verificou a presença de impasses para os testes com uma única entidade de aplicação ativa. Como não se elaborou uma sequência de testes para várias entidades de aplicação ativas, não se garante que nestes casos inexistam impasses.

6.3 EXEMPLO DO FUNCIONAMENTO DOS TESTES

Do ponto de vista de operação funcional do protocolo foi implementada a fase de estabelecimento de conexão. Portanto, o funcionamento dos processos PTS e PTI será mostrado tomando como exemplo a fase de estabelecimento de conexão, conforme a figura 6.1.

Primeiramente o PTS faz as seguintes inicializações:

. identifica as quatro filas de mensagens superiores (PGIC iniciador e respondedor) e
. cria, dinamicamente, um buffer intermediário para identificação das mensagens lidas nas filas de saída superiores.

Em seguida, constrói a primitiva F-INITIALIZE request e a envia para a fila de entrada superior do PGIC iniciador. Neste momento, o PTS fica varrendo alternadamente as filas de saída superiores dos PGICs iniciador e respondedor.

O protocolo FTAM recebe a primitiva, constrói a UDP Initialize request e a envia para a fila de saída inferior do iniciador numa primitiva A-Associate request do CASE.

O PTI ao ser ativado identifica as quatro filas de mensagens inferiores (duas filas do iniciador e duas filas do respondedor) e aloca um buffer intermediário. Em seguida recebe a primitiva A-Associate request contendo a UDP Initialize request. Constrói uma primitiva A-Associate indication e a envia para a fila de entrada inferior do processo respondedor. Fica varrendo alternadamente as filas de saída inferior do respondedor e do iniciador.

A primitiva A-Associate indication é recebida pelo respondedor e mapeada numa primitiva F-Initialize indication que é enviada para a sua fila de saída superior.

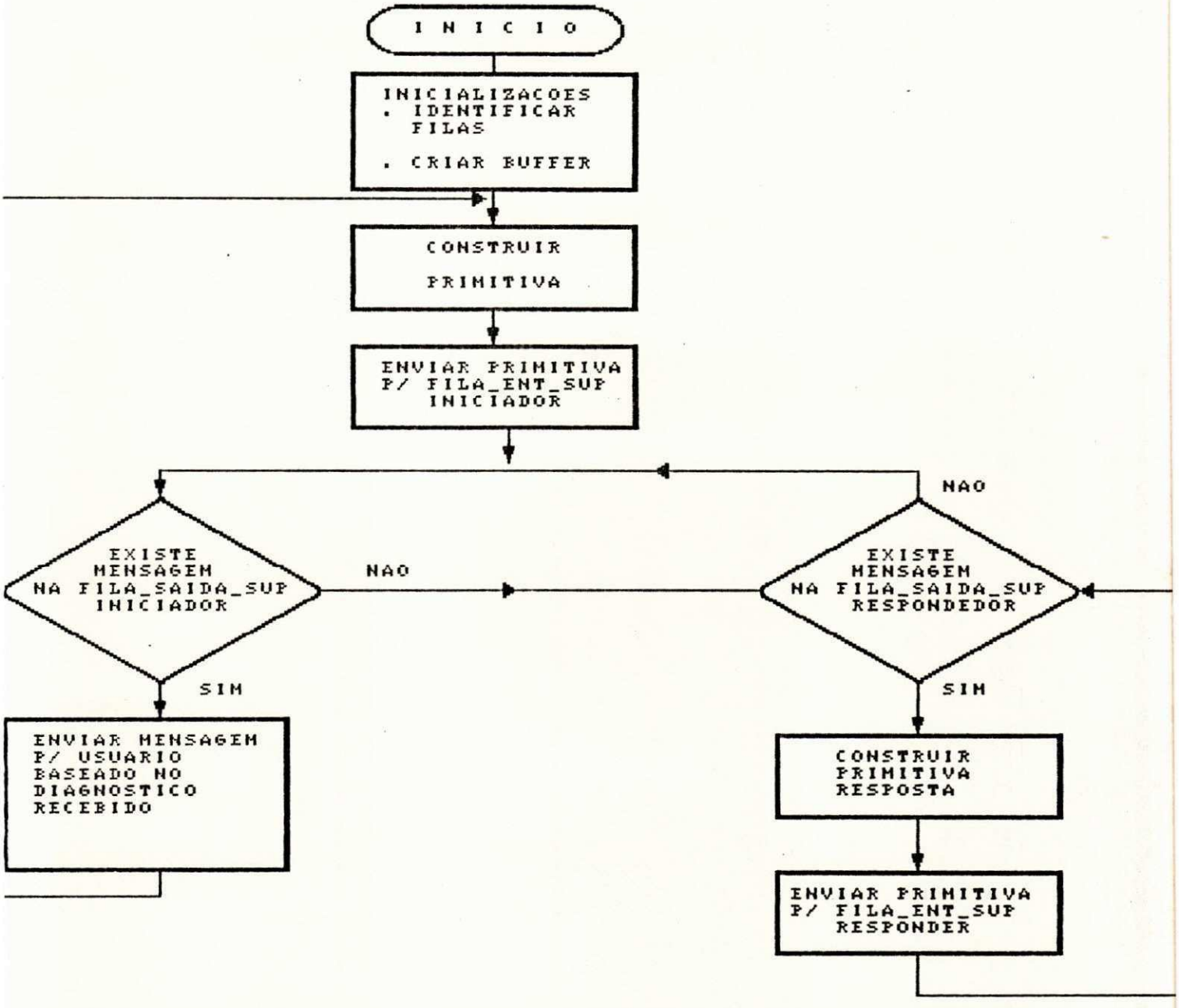
O PTS recebe a primitiva F-Initialize indication, constrói uma F-Initialize response, envia-a para a fila de entrada superior do respondedor e volta a varrer alternadamente as filas de saída do iniciador e respondedor.

O FTAM respondedor constrói uma UDP Initialize response, baseada na primitiva F-Initialize response recebida e a envia para sua fila de saída inferior numa primitiva A-Associate response.

O processo de teste inferior recebe a A-Associate response e constrói a A-Associate confirm que contém a UDP Initialize response. Envia essa primitiva (A-Associate confirm) para a fila de entrada inferior do processo iniciador.

O iniciador mapeia a primitiva A-Associate confirm para uma F-Initialize-confirm e a envia para a sua fila de saída superior. O PTS verifica o campo diagnóstico dessa primitiva e envia uma mensagem para o usuário avisando do sucesso ou não do estabelecimento de conexão.

Os fluxogramas gerais dos processos de teste superior e inferior são mostrados nas figura 6.2 e 6.3.



URA 6.2: FLUXOGRAMA GERAL DO PROCESSO DE TESTE SUPERIOR - PTS

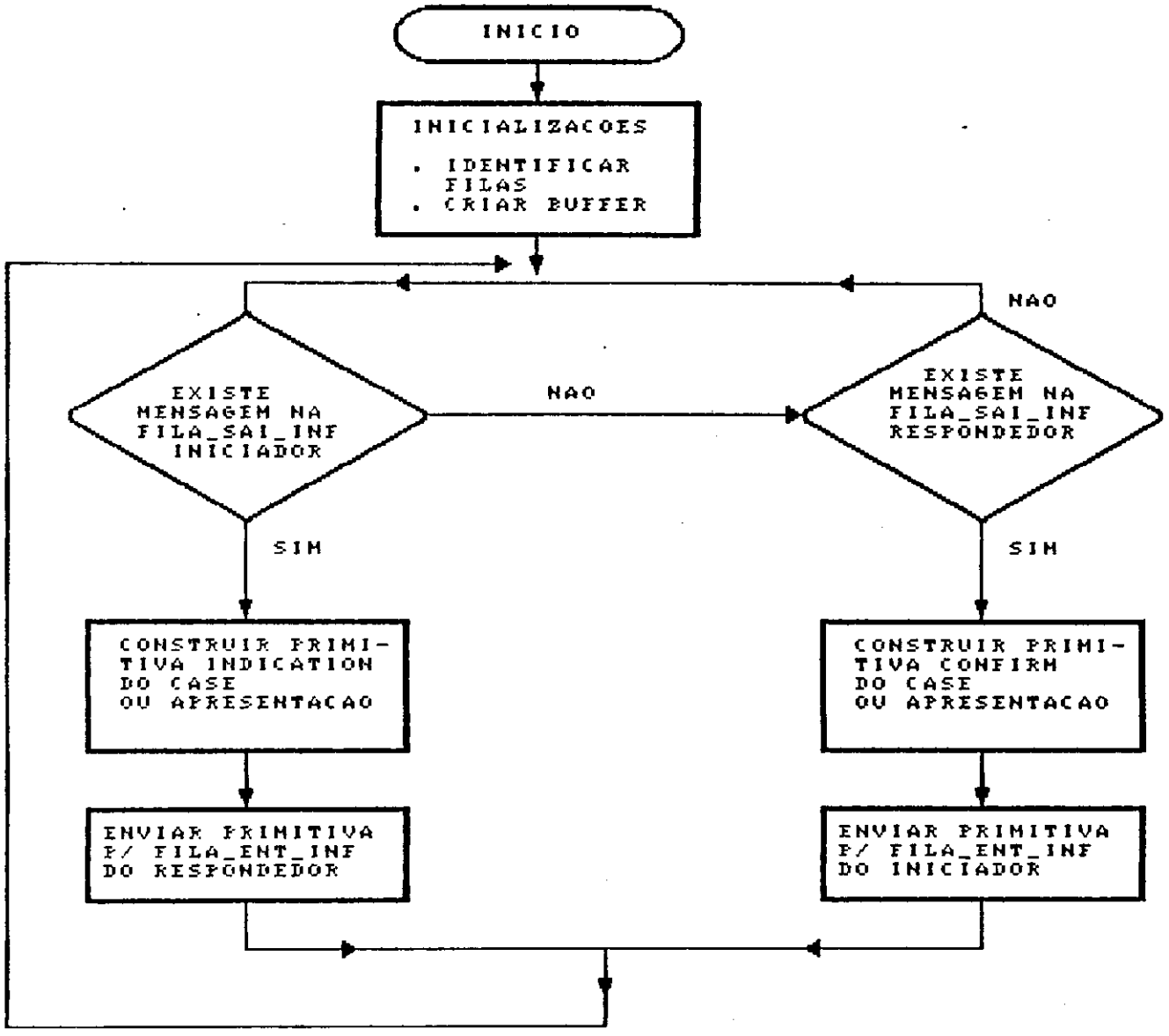


FIG. 6.3: FLUXOGRAMA GERAL DO PROCESSO DE TESTE INFERIOR - PTI

Capítulo 7

CONSIDERAÇÕES FINAIS

Um dos aspectos considerados na implementação do protocolo FTAM foi o seguimento das especificações do padrão conforme definido pela ISO. Para possibilitar a realização de tal objetivo utilizou-se um sistema computacional de porte considerável, rodando um sistema operacional com recursos suficientes.

Os resultados obtidos com a implementação mostrou que o nível de suporte necessário para viabilizar o desenvolvimento do FTAM é encontrado normalmente em computadores com uma capacidade computacional igual ou superior ao que se convenciona chamar de "supermicro".

No que diz respeito ao modelo funcional, implementou-se uma estrutura flexível para os processos e para os dados.

A estrutura de processos permite a implementação de funções de gerência mais eficientes, permitindo inclusive a implementação de um protocolo de gerência de camada de rede. Em relação à implementação realizada, destacam-se as seguintes facilidades:

- alocação dinâmica de processos;
- controle dos recursos do sistema operacional do computador e
- pseudo paralelismo na execução do protocolo por conexão, característica esta que deve repercutir positivamente no desempenho do protocolo.

Uma estruturação alternativa de processos também pode ser suportada nesta implementação do FTAM. Ela consiste na alocação de um processo PGC por camada mas com vários PGCs num mesmo endereço de apresentação.

Em relação às desvantagens da estrutura de processos utilizada, destaca-se a limitação do número de conexões ativas em função da própria limitação dos recursos do sistema operacional, tais como: número máximo de processos ativos, bufferização de filas, número de filas, tamanho do segmento de memória compartilhada, etc....Mas, esses recursos podem ser ajustados, pois são todos parametrizáveis.

Um aspecto muito importante relacionado às estruturas de dados, é o fato de terem sido dimensionadas para suportar o pacote FTAM completo.

O FTAM implementado representa um volume de código razoável, que pode ser percebido através de uma análise das estruturas de dados das tabelas e do número de linhas de código dos processos utilizados. Em seguida, apresenta-se alguns dados:

. TABELA DE ESTADO ATUAL

- Tamanho do Objeto - 4.080 octetos por entrada
- Número de Entradas - igual ao número máximo de conexões ativas previstas.

. TABELA DE TRANSIÇÃO

- Tamanho do Objeto - 32 octetos por entrada (máximo de 6 funções para testes de predicados)

- Número de Entradas :

Num. de Eventos X Num. de Estados = 89 X 62

= 5.518 entradas

. TABELA DE FUNÇÃO DE SAíDA

- Aciona até 7 funções de saída para as combinações evento/estado possíveis e testes de predicados.

. PROCESSO GERENTE DA INTERFACE E DA CAMADA (PGIC)

- Linhas de Código: 1.747 linhas

. PROCESSO GERENTE DE CONEXÃO (PGC)

- Linhas de Código: 9.901 linhas

Anexo A

Exemplo do Estabelecimento de Conexão no Protocolo FTAM Implementado

O exemplo do funcionamento do protocolo FTAM implementado para o estabelecimento de conexão é descrito de acordo com a figura A.1.

A entidade iniciadora recebe na fila de entrada superior a primitiva F-INITIALIZE request advinda do usuário da aplicação local. Essa mensagem é lida pelo PGIC local.

Como a primitiva é de estabelecimento de conexão, o PGIC local cria um processo PGC que fará a gerência desta conexão. Cria também uma fila para comunicação com o PGC e envia a mensagem para a referida fila.

O PGC recebe a primitiva, aciona a máquina de protocolo através das tabelas de transição e função de saída, faz os testes de predicados e executa as ações específicas para este evento. Constrói então, uma UDP F-INITIALIZE request e a envia para a fila de saída inferior numa primitiva A-ASSOCIATE request do CASE/ACSE.

A primitiva A-ASSOCIATE request carregando a UDP F-INITIALIZE request é, no nosso caso, enviada para a fila de entrada inferior da entidade remota pelo Processo de Teste Inferior - PTI (ver Capítulo 6).

O PGIC remoto lê a fila de entrada inferior, cria processo PGC, constrói a fila de comunicação deste PGC e envia a primitiva/UDP recebida para esta fila. Esse PGC remoto recebe a mensagem, aciona a máquina de protocolo, executa as ações necessárias e constrói uma primitiva F-INITIALIZE indication que é enviada para a fila de saída superior da entidade respondedora.

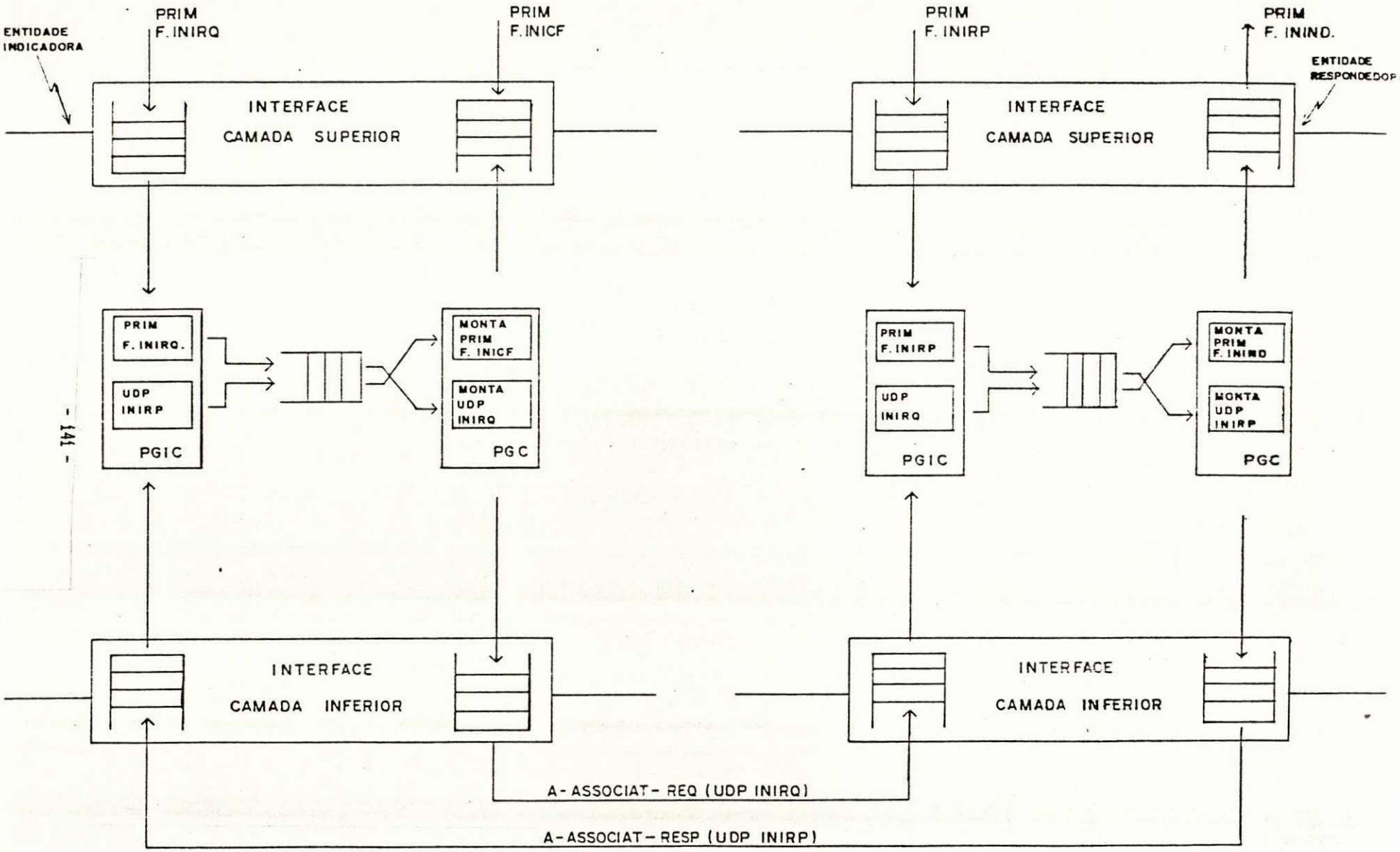


Fig. A.1 : Estabelecimento de Conexão FTAM no Protocolo Implementado

O usuário respondedor, ao receber a F-INITIALIZE indication, via Processo de Teste Superior - PTS, envia uma primitiva F-INITIALIZE response para a fila de entrada superior. Essa mensagem é lida pelo PGIC respondedor que localiza, através da tabela de estado atual (segmento de memória compartilhada), o processo e a fila para o qual se destina aquela mensagem.

Em seguida a mensagem é enviada para a fila do PGC adequado. O PGC recebe a mensagem e novamente aciona a máquina de protocolo, construindo uma UDP F-INITIALIZE response que é enviada para a fila de saída inferior numa primitiva A-ASSOCIATE response do CASE/ACSE.

Essa primitiva carregando a UDP é, no nosso caso, enviada para a fila de entrada inferior da entidade iniciadora pelo Processo de Teste Inferior - PTI.

O PGC da entidade iniciadora recebe a UDP, aciona a máquina de protocolo, executa os procedimentos para o evento recebido, constrói uma primitiva F-INITIALIZE confirm e a envia para a fila de saída superior que será lida pela aplicação local.

Caso a confirmação da entidade respondedora seja positiva, a conexão está estabelecida.

Anexo B

CÓDIGO FONTE DOS PROCESSOS (SIMPLIFICADO)

```
#include        (stdio.h)
#include        (sys/types.h)
#include        (sys/ipc.h)
#include        (sys/msg.h)
#include        (sys/shm.h)
#include        (signal.h)
#include        "g_param.h"
#include        "f_param.h"
#include        "extern1.h"
#include        "tab_atual.h"
#include        (errno.h)

#undef          SHMLBA
#define         SHMLBA          32768           /* Incluí este valor baseado no código
                                                 * de Jacques pois estava dando problemas
                                                 * devido as macros ctob e stoc não
                                                 * estarem definidas
                                                 */
#define         FTAM1_MEM       ((key_t)01)     /* Chave com nome para memória compartilhada */
#define         MAXMEM          (100 *1024L)    /* Bytes para malloc ( ) */
#define         PATH            "/usr/\@cliz/fontes/pgci"
#define         PERM            0666

main ( )
{
        struct  t_msg           *msgp;
        struct  t_in_rq         *in_rq_p;
        struct  t_assc_ind      *assc_ind_p;
        struct  t_est_at        *in_shm_pgis( ),
                                *varre_tab ( ),
                                *area_livre ( );
        struct  t_est_at        *apm_pgis, *ap_area;
        void    init ( );
        void    joga_buffer ( );
        void    copia_msg ( );
        void    init_conf_neg ( );
        int f1, f3, idf1, idf2, idf3, idf4, idf, id_assc,
            i, j, x,id_proc, erro, prim, f_pgc;

        if ((msgp = (struct t_msg *) malloc (sizeof (struct t_msg))) == NULL){
             printf ("ERRO FATAL: Buffer para PGIS não foi alocado\n");
             exit(1);
        }
        apm_pgis = in_shm_pgis ( );
```

```
    if ((idf1 = cria_fila (F_E_SUP1)) == -1) {
        printf ("ERRO FATAL: Fila de Entrada Superior não foi criada \n");
        cfree (msgp);
        exit (1);
    }
    if ((idf2 = cria_fila (F_S_SUP1)) == -1) {
        printf ("ERRO FATAL: Fila de Saída Superior não foi criada \n");
        printf ("errno = %d\n",errno);
        cfree (msgp);
        exit (1);
    }
    if ((idf3 = cria_fila (F_E_INF1)) == -1) {
        printf ("ERRO FATAL: Fila de Entrada Inferior não foi criada \n");
        printf ("errno = %d\n",errno);
        cfree (msgp);
        exit (1);
    }
    if ((idf4 = cria_fila (F_S_INF1)) == -1) {
        printf ("ERRO FATAL: Fila de Saída Inferior não foi criada \n");
        printf ("errno= %d\n",errno);
        cfree (msgp);
        exit (1);
    }

    init (apm_pgis);
while (1) {
    if (f1 = varre_fila (idf1)) {
       if ((x = joga_buffer (idf1,msgp)) == -1){
         printf ("ERRO FATAL: PGIS1 não conseguiu retirar msg da fila de saída superior\n");
         lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
         exit (1);
       }
       if ((prim = ident_prim (msgp)) == F_INIR0) {
           if ((ap_area = area_livre (apm_pgis)) != (struct t_est_at *) -1) {
                if ((id_proc = fork( )) != -1) {
                     if (id_proc == 0) {
                         if ((erro = execl(PATH,0)) == -1){
                              printf ("errno =%d\n",errno);
                              lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
                              exit (1);
                         }
                     }
```

```
                    else {
                        if ((f_pgc = cria_fila (id_proc)) != -1){
                            in_rq_p =  (struct t_in_rq *) &(msgp -> txt_msg [0]);
                            ap_area -> flag = 1;
                            ap_area -> idf_assc = in_rq_p -> e_ap_ch;
                            ap_area -> pr_gr_cnx = id_proc;
                            ap_area -> idfila = f_pgc;
                            ap_area -> f_s_sup = idf2;
                            ap_area -> f_s_inf = idf4;
                            copia_msg (f_pgc,msgp,ap_area);
                        }
                        else {
                             init_conf_neg (msgp,idf2);
                             kill (id_proc,SIGKILL);
                        }
                    }
                }
                else {
                     init_conf_neg (msgp,idf2);
                     lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
                }
            }
            else {
                 init_conf_neg (msgp,idf2);
                 lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
            }
        }
        else {
             id_assc = ident_assc (msgp);
             ap_area = varre_tab (id_assc,apm_pgis);
             idf = ap_area -> idfila;
             copia_msg (idf,msgp,ap_area);
        }
    }
    if (f3 = varre_fila (idf3)) {
        if ((x = joga_buffer (idf1,msgp)) == -1){
            printf ("ERRO FATAL: PGIS1 não conseguiu retirar msg da fila de saída superior\n");
            lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
            exit (1);
        }
        if ((prim = ident_prim (msgp)) == ASSC_IN) {
            if ((ap_area = area_livre (apm_pgis)) != (struct t_est_at *)-1) {
                if ((id_proc = fork( )) != -1) {
                    if (id_proc == 0) {
```

```
                    if ((erro = execl(PATH,0)) == -1){
                        printf ("ERRO FATAL: Processo PGC não pode executar\n");
                        lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
                        exit (1);
                    }
                }
                else {
                    if ((f_pgc = cria_fila (id_proc)) != -1){
                        assc_ind_p = (struct t_assc_ind *) & (msgp -> txt_msg [0]);
                        ap_area -> flag = 1;
                        ap_area -> idf_assc = assc_ind_p -> e_ap_ct;
                        ap_area -> pr_gr_cnx = id_proc;
                        ap_area -> idfila = f_pgc;
                        ap_area -> f_s_sup = idf2;
                        ap_area -> f_s_inf = idf4;
                        ap_area -> flag = 1;
                        if ((y = copia_msg (f_pgc,msgp,ap_area)) == -1){
                            printf ("ERRO FATAL: PGIS1 não conseguiu escrever msg na fila pgc\n");
                            lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
                            exit (1);
                        }
                    }
                    else {
                        init_conf_neg (msgp,idf2);
                        kill (id_proc,SIGKILL);
                    }
                }
            }
            else {
                init_conf_neg (msgp,idf2);
                lib_recursos(idf1,idf2,idf3,idf4,apm_pgis,msgp);
            }
        }
        else {
            init_conf_neg (msgp,idf2);
            lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp);
        }
    }
    else {
        id_assc = ident_assc (msgp);
        ap_area = varre_tab (id_assc, apm_pgis);
        idf = ap_area -> idfila;
        copia_msg (idf,msgp,ap_area);
    }
  }
 }
}
```

```
int
cria_fila (chave)
int chave;
{
        int ident;

        ident = msgget (chave,IPC_CREAT|PERM);
        return (ident);
}

int
varre_fila (id)
int id;
{
        struct msqid_ds  buf1, *bufip;

        bufip = &buf1;
        msgctl (id,IPC_STAT,bufip);
        if (bufip -> msg_qnum ) {
            return (1);
        }
        else
            return (0);
}

int
joga_buffer (idf,msgp)
int idf;
struct  t_msg   *msgp;
{
        int x;

        if (msgrcv (idf,msgp,TAM_MSG,1,0) == -1)
            return (-1);
        else
            return (1);
}
```

```
int
ident_prim (msgp)
struct  t_msg  *msgp;
{
        int     prim;
        struct  t_tm_rq    *in_rq_p;          /* Aponta para uma primitiva do tipo terminate.
                                                 É usado para identificar se a primitiva é uma
                                                 F_initialize_request */
        in_rq_p =  (struct t_tm_rq *) & (msgp -) txt_msg[0]);
        prim = in_rq_p -) id_serv;

        return (prim);

}

int
ident_assc (msgp)
struct  t_msg  *msgp;
{
        int     assc;
        struct  t_tm_rq  *tm_rq_p;            /* Aponta para uma primitiva do tipo terminate.
                                                 É usado para identificar a associação de
                                                 qualquer primitiva, exceto F_Init_Req */
        tm_rq_p = (struct t_tm_rq *) & (msgp -) txt_msg [0]);
        assc = tm_rq_p -) id_assc;
        return (assc);
}

struct  t_est_at  *
varre_tab (id_assc, apm_pgis)
int id_assc;
struct  t_est_at   *apm_pgis;
{
        int i;

        for (i = 0; i ( NPROC; apm_pgis++,i++){
             if (apm_pgis -) idf_assc == id_assc) {
                 return (apm_pgis);
             }
        }
        return ((struct t_est_at *)-1);
}
```

```
void
copia_msg (idf,msgp,ap_area)
int idf;
struct  t_msg     *msgp;
struct  t_est_at  *ap_area;
{

        if (msgsnd (idf,msgp,TAM_MSG,0) == -1){
            printf ("ERRO FATAL: Não conseguiu enviar msg para a fila PGC \n");
            exit (1);
        }
        else {
             ap_area -> sinal += 1;
        }
}

struct  t_est_at  *
area_livre (ap)
struct  t_est_at  *ap;
{
        int j;

        for (j = 0; j < NPROC; ap++,j++) {
             if (ap -> flag == 0){
                 return (ap);
             }
        }
        printf ("Não existe área livre na tab_atual \n");
        return ((struct t_est_at *)-1);
}

void
init_conf_neg (msgp,idf2)
struct  t_msg   *msgp;
int     idf2;
{
        char    *mensagem;
        int     x;
```

```
        mensagem = "Não existe área livre na tabela de estado atual ou\
                    Não conseguiu criar fila para processo PGC ou \
                    Não conseguiu criar processo PGC";
        in_cnf.id_serv = F_INIRP;
        in_cnf.id_assc = ident_assc (msgp);
        in_cnf.r_est   = FALH;
        in_cnf.r_ac    = E_PRM;
        strcpy (in_cnf.t_apl_r ,"FTAM");
        in_cnf.e_ap_r  = 0;
        in_cnf.g_cont_ap = FALSO;
        in_cnf.und_func = 0;
        in_cnf.gp_atb = 0;
        in_cnf.rbck = 0;
        in_cnf.q_s_com = 0;
        in_cnf.diagn.tp_dgn = E_PRM;
        in_cnf.diagn.id_err = P_I_GF;
        in_cnf.diagn.ob_err = 0;
        in_cnf.diagn.ft_err = 0;
        in_cnf.diagn.es_sug = 0;
        x = sizeof (*mensagem);
        strncpy (in_cnf.diagn.d_ad,*mensagem,x);
        in_cnf.chkp = 0;

        msgp -) tip_msg = 0;
        strcpy (msgp -) txt_msg [0],in_cnf);
        msgsnd (idf2,msgp,TAM_MSG,0);
}

struct  t_est_at  *
in_shm_pgis ( )
{
        int shm_id;
        struct  t_est_at  *ap_mem;
        char    *ap_tab;
        char    *shmat ( );
        char    *sbrk ( );
        char    *ender;

        /* calculando endereço e deixando área para malloc ( ) */

        ender = sbrk (0) + MAXMEM + SHMLBA - 1;
        printf ("ender = %d\n",ender);
```

```
        shm_id = shmget (FTAM1_MEM,(NPROC * sizeof(struct t_est_at)),IPC_CREAT|PERM);
        if (shm_id == -1) {
            fprintf (stderr,"Não criou memória compartilhada - PGIS\n");
            exit (1);
        }
        printf ("shm_id = %d\n",shm_id);
        ap_mem = (struct t_est_at *) shmat (shm_id, (char *) ender,SHM_RND);
        if (ap_mem == (struct t_est_at *) -1){
            fprintf (stderr,"Não vinculou memória compartilhada\n");
            exit (1);
        }
        return (ap_mem);
}

void
init (apm_pgis)
struct  t_est_at  *apm_pgis;
{
        int i;

        for (i = 0; i < NPROC; apm_pgis++,i++){
             apm_pgis -> idf_assc = 0;
             apm_pgis -> pr_gr_cnx = 0;
             apm_pgis -> idfila = 0;
             apm_pgis -> f_s_sup = 0;
             apm_pgis -> f_s_inf = 0;
             apm_pgis -> est_cnx = CLOSED;
             apm_pgis -> sinal = 0;
             apm_pgis -> flag  = 0;
        }
}

void
lib_recursos (idf1,idf2,idf3,idf4,apm_pgis,msgp)
int idf1,idf2,idf3,idf4;
struct t_est_at *apm_pgis;
struct t_msg    *msgp;
{
        msgctl (idf1,IPC_RMID,NULL);
        msgctl (idf2,IPC_RMID,NULL);
        msgctl (idf3,IPC_RMID,NULL);
        msgctl (idf4,IPC_RMID,NULL);
        shmctl (apm_pgis,IPC_RMID,NULL);
        cfree  (msgp);
}
```

```
#include        (sys/types.h)
#include        (sys/ipc.h)
#include        (sys/shm.h)
#include        (sys/msg.h)
#include        (stdio.h)
#include        "f_param.h"
#include        "g_param.h"
#include        "extern1.h"
#include        "tab_atual.h"
#include        (errno.h)

#undef          SHMLBA
#define         SHMLBA          32768                                           */
#define         FTAM1_MEM       ((key_t)01)     /* Chave com nome para memória compartilhada */
#define         MAXMEM          (100 *1024L)    /* Bytes para malloc ( ) */
#define         PERM            0666

extern  struct t_tab_trans {
                int (*f_pred [8]) ( );
                int tab_fsd;
}tab_trs [][62];

extern  int     (*f_ac[][9][7]) ();

struct  t_est_at     *apm_pgc;

main ( )
{
        struct  t_msg     *msgp;
        struct  shmid_ds  *buf;
        struct  msqid_ds  *fila;
        int     id_proc, id_fil, id_event, est_cnx,i;
        void    maq_protocolo ( );
        struct  t_est_at  *in_pgc_shm ( );
        char    *malloc ( );
        char    *x;

        /* Utilização de massete recomendado por analista da EDISA pois nesta
         * versão (2.08) o malloc não funciona legal com filas de mensagens
         */
```

```
        x = malloc (1024);
        cfree (x);

        apm_pgc = in_pgc_shm ( );
        if ((msgp = (struct t_msg *) malloc (sizeof(struct t_msg))) == NULL){
             printf ("ERRO FATAL: Buffer para PGC não foi alocado\n");
             printf ("errno = %d\n",errno);
             shmctl (apm_pgc,IPC_RMID,buf);
             exit (.);
        }
        id_proc = getpid ( );
        for (; apm_pgc->pr_gr_cnx != id_proc; apm_pgc++)
             ;
        apm_pgc -) msgp = msgp;                 /* Coloquei o valor do endereço do buffer
                                                 * na mem. comp. para as funções da maq.
                                                 * de protocolo poderem acessar o buffer,
                                                 * caso contrário teria que passar parâmetro
                                                 */
inicio:
        while (apm_pgc->sinal == 0)
               ;
        id_fil = apm_pgc->idfila;

        if (msgrcv (id_fil,msgp,TAM_MSG,1,0) == -1) {
            printf ("ERRO FATAL: PGC não conseguiu ler mensagem da sua fila\n");
            printf ("errno = %d\n",errno);
            shmctl (apm_pgc,IPC_RMID,&buf);
            cfree  (msgp);
            msgctl (id_fil,IPC_RMID,fila);
            exit (1);
        }
        apm_pgc->sinal -= 1;
        id_event = ident_evento (msgp);
       est_cnx  = apm_pgc->est_cnx;
        maq_protocolo (id_event,est_cnx);
        if (apm_pgc->est_cnx == CLOSED){
            apm_pgc->flag = 0;
        }
       else {
             goto inicio;
        }
}
```

```
int
ident_evento (msgp)
struct  t_msg   *msgp;
{
        struct  t_tm_rq  *tm_rq_p;

        tm_rq_p = & (msgp -) txt_msg[0]);
        return (tm_rq_p -) id_serv);
}

struct t_est_at *
in_pgc_shm ( )
{
        int shm_id;
        struct  t_est_at  *ap_mem;
        struct  shmid_ds  *buf;
        char    *shmat ( );
        char    *sbrk ( );
        char    *ender;

        /* primeiro endereço disponível, deixando área para malloc ( ) */

        ender = sbrk (0) + MAXMEM + SHMLBA - 1;

        shm_id = shmget (FTAM1_MEM,(NPROC * sizeof (struct t_est_at)),IPC_CREAT|PERM);
        if (shm_id == -1) {
           fprintf (stderr,"Não criou memória compartilhada - PGC \n");
           exit (1);
        }
        ap_mem = (struct t_est_at *) shmat (shm_id, (char *) ender, SHM_RND);
        if (ap_mem == (struct t_est_at *)-1){
            printf ("Não vinculou memória compartilhada \n");
            shmctl (ap_mem,IPC_RMID,buf);
            exit (1);
        }
        return (ap_mem);

}
```

```
void
maq_protocolo (id_event,est_cnx)
int id_event, est_cnx;
{
        int (*f_prd) ( );
        int j, x, y, z;

        for (j=0; (f_prd = tab_trs[id_event][est_cnx].f_pred[j]) != NULL; j++){
             if ((x = (*f_prd) ()) != 0) {
                  y = tab_trs [id_event][est_cnx].tab_fsd;
                  for (z=0; f_ac [y][x][z] != NULL; z++){
                       (*f_ac [y][x][z]) ( );
                  }
                  break;
             }
        }
}
```

```
#include (sys/types.h)
#include (sys/ipc.h)
#include (sys/msg.h)
#include "g_param.h"
#include "f_param.h"
#include "extern1.h"
#include "tab_atual.h"

/*
 * FUNÇÕES REFERENTES À TABELA DE ESTADO DO REGIME FTAM - INICIADOR
 */

s_pred ( )
{
}

int tpi11_rf ( )                        /* Testa P1 do regime FTAM */
{

        extern  struct t_est_at *apm_pgc;
        struct  t_in_rq         *p_in_rq;

        p_in_rq = (struct t_in_rq *) & (apm_pgc -) msgp -) txt_msg[0]);

        if (p_in_rq -) niv_serv != CORR_US){
            return (2);
        }
        if (p_in_rq -) clas_serv != ACES)
            return (2);
        if (p_in_rq -) und_func != G_AR_LM && p_in_rq -) und_func != ' ' ){
            return (2);
        }
        if (p_in_rq -) rbck != S_RBCK )
            return (2);
        if (p_in_rq -) chkp != 0 && p_in_rq -) chkp != ' ' )
            return (2);
        else
            return (1);
}
```

```
inirq_pdu ( )                          /* PDU Initialize request */
{
        extern  struct  t_est_at *apm_pgc;
        struct  t_in_rq          *p_in_rq;

        p_in_rq = (struct t_in_rq *) & (apm_pgc -) msgp -) txt_msg[0]);

        in_req.id_pdu = INIRQ;
        in_req.versao.p_vers = 1.0;
        in_req.g_cont_ap = p_in_rq -) g_cont_ap;
        in_req.niv_serv = p_in_rq -) niv_serv;
        in_req.clas_serv = p_in_rq -) clas_serv;
        in_req.und_func = p_in_rq -) und_func;
        in_req.gp_atb = p_in_rq -) gp_atb;
        in_req.rbck = p_in_rq -) rbck;
        strcpy (in_req.lst_cont.n_t_dc,p_in_rq -) lst_cont.n_t_dc);
        strcpy (in_req.lst_cont.cj_stx.n_cj_r,p_in_rq -) lst_cont.cj_stx.n_cj_r);
        strcpy (in_req.lst_cont.cj_stx.n_st_a,p_in_rq -) lst_cont.cj_stx.n_st_a);
        strcpy (in_req.id_i,p_in_rq -) id_i);
        strcpy (in_req.cont,p_in_rq -) cont);
        strcpy (in_req.senh.g_senh,p_in_rq -) senh.g_senh);
        in_req.chkp = p_in_rq -) chkp;
}

inirq_aci_rf ( )                              /* Ação [1] do regime FTAM para a PDU INIRQ */
{
        struct  t_assc_rq    *p_assc_rq;
        struct  t_in_rq      *p_in_rq;
        struct  t_msg        *buffer;
        char    *memcpy ( );
        extern  struct  t_est_at     *apm_pgc;
        int     n, idf;

        /* Manipulação de apontadores para construção da msg com a primitiva
           A_Assc_Req carregando a pdu INIRQ */

        p_in_rq   = (struct t_in_rq *) & (apm_pgc -) msgp -) txt_msg[0]);

        n = sizeof (long) + sizeof (struct t_assc_rq) + sizeof (struct t_in_req);
        buffer = (struct t_msg *) malloc (n);
        buffer -) tip_msg = 1;
        p_assc_rq = (struct t_assc_rq *) &(buffer -) txt_msg [0]);
```

```
        p_assc_rq -)  id_serv  = ASSC_RQ;
        strcpy (p_assc_rq -) t_apl_ct , p_in_rq -) t_apl_ct);
        strcpy (p_assc_rq -) t_apl_ch , p_in_rq -) t_apl_ch);
        strcpy (p_assc_rq -) n_ct_apl ,"FTAM");
        p_assc_rq -) inf_us = ' ';
        p_assc_rq -) e_ap_ct  = p_in_rq -) e_ap_ct;
        p_assc_rq -) e_ap_ch  = p_in_rq -) e_ap_ch;
        p_assc_rq -) ctx_apr  = ' ';
        p_assc_rq -) l_df_ctxap = ' ';
        p_assc_rq -) n_def_ctxap = ' ';
        p_assc_rq -) ql_serv  = p_in_rq -) q_s_com;
        p_assc_rq -) req_apr = ' ';
        p_assc_rq -) req_sess = ' ';
        p_assc_rq -) n_pt_in_sinc = 0;
        p_assc_rq -) atb_tkn = ' ';
        p_assc_rq -) idf_cnx_sess = ' ';

        ++ p_assc_rq;
        memcpy (p_assc_rq, &in_req,sizeof (struct t_in_req));


        /* Enviar a primitiva construida para a fila de saida inferior */

        idf = apm_pgc -) f_s_inf;

        if (msgsnd (idf, buffer, TAM_MSG, 0) == -1) {
            printf ("ERRO FATAL: Não conseguiu enviar a primitiva F_Assc_Req para a fila de saída inferior\n");
            cfree  (buffer);
            cfree  (apm_pgc -) msgp);
            exit  (1);
        }
        cfree (buffer);
}

ac3_rf ( )                                  /* Ação [3] do regime FTAM - Nesta implementação
                                               optamos por inicializar as informações de
                                               estado no proprio pgis */
{
}

init_pd ( )
{
        extern  struct t_est_at  *apm_pgc;

        apm_pgc -) est_cnx = INIT_PD;
}
```

```
tprii_rf ( )                          /* Testa P5 do Regime FTAM */
{

        extern  struct t_est_at *apm_pgc;
        struct  t_in_req        *p_in_req;
        struct  t_assc_ind      *p_assc_ind;

        p_assc_ind = (struct t_assc_ind *) & (apm_pgc -> msgp -> txt_msg [0]);
        p_in_req  =.(struct t_in_req *) ++ p_assc_ind;

        if (p_in_req -> id_pdu != INIRQ)
            return (2);
        if (p_in_req -> versao.p_vers != 1.0)
            return (2);
        if (p_in_req -> niv_serv != CORR_US)
            return (2);
        if (p_in_req -> clas_serv != ACES)
            return (2);
        if (p_in_req -> und_func != G_AR_LM && p_in_req -> und_func != ' ')
            return (2);
        if (p_in_req -> rbck != S_RBCK)
            return (2);
        if (p_in_req -> chkp != 0 && p_in_req -> chkp != ' ')
          . return (2);
        else
            return (1);
}

f_iniin_pr ( )
{
        extern  struct t_est_at *apm_pgc;
        struct  t_assc_ind      *p_assc_ind;
        struct  t_in_req        *p_in_req;
        int     idf;

        /* Construção da primitiva F_INIT_IND */

        p_assc_ind = (struct t_assc_ind *) & (apm_pgc -> msgp -> txt_msg[0]);


        in_ind.id_serv = F_INIIN;
        strcpy (in_ind.t_apl_ch , p_assc_ind -> t_apl_ch);
        strcpy (in_ind.t_apl_ct , p_assc_ind -> t_apl_ct);
        in_ind.e_ap_ch  = p_assc_ind -> e_ap_ch;
      . in_ind.e_ap_ct  = p_assc_ind -> e_ap_ct;
        in_ind.q_s_com = p_assc_ind -> ql_serv;
```

```
        p_in_req   = (struct t_in_req *) ++ p_assc_ind;
        in_ind.g_cont_ap = p_in_req -> g_cont_ap;
        in_ind.niv_serv  = p_in_req -> niv_serv;
        in_ind.clas_serv = p_in_req -> clas_serv;
        in_ind.und_func  = p_in_req -> und_func;
        in_ind.gp_atb = p_in_req -> gp_atb;
        in_ind.rbck   = p_in_req -> rbck;
        strcpy (in_ind.lst_cont.n_t_dc, p_in_req -> lst_cont.n_t_dc);
        strcpy (in_ind.lst_cont.cj_stx.n_cj_r, p_in_req -> lst_cont.cj_stx.n_cj_r);
        strcpy (in_ind.lst_cont.cj_stx.n_st_a, p_in_req -> lst_cont.cj_stx.n_st_a);
        strcpy (in_ind.id_i, p_in_req -> id_i);
        strcpy (in_ind.cont, p_in_req -> cont);
        strcpy (in_ind.senh.g_senh, p_in_req -> senh.g_senh);
        in_ind.chkp = p_in_req -> chkp;

      /* Jogar a primitiva construída no buffer do pgc e depois envia-la
         para a fila de saída superior */

        idf = apm_pgc -> f_s_sup;
        apm_pgc -> msgp -> tip_msg = 1;
        memcpy (apm_pgc -> msgp -> txt_msg, &in_ind, sizeof (struct t_in_req));

        if (msgsnd (idf, apm_pgc -> msgp, TAM_MSG, 0) == -1) {
          printf ("ERRO FATAL: Não conseguiu enviar F_INIT_IND para a fila\
                               de saída superior \n");
          cfree (apm_pgc -> msgp);
          exit (1);
        }
}

tpr21_rf ( )                          /* Testa P3 do Regime FTAM */
{
        extern  struct  t_est_at  *apm_pgc;
        struct  t_in_rsp          *p_in_rsp;

        p_in_rsp = (struct t_in_rsp *) & (apm_pgc -> msgp -> txt_msg [0]);

        if (p_in_rsp -> r_est == SUCS)
           return (1);
        else
           return (2);
}
```

```
inirp_pdu ( )
{

        extern  struct t_est_at *apm_pgc;
        struct  t_in_rsp        *p_in_rsp;

        p_in_rsp = ( struct t_in_rsp *) & (apm_pgc -> msgp -> txt_msg[0]);

        in_resp.id_pdu = INIRP;
        in_resp.r_est  = p_in_rsp -> r_est;
        in_resp.r_ac   = p_in_rsp -> r_ac;
        in_resp.versao.p_vers = 1.0;
        in_resp.g_cont_ap = p_in_rsp -> g_cont_ap;
        in_resp.und_func  = p_in_rsp -> und_func;
        in_resp.gp_atb    = p_in_rsp -> gp_atb;
        in_resp.rbck      = p_in_rsp -> rbck;
        strcpy (in_resp.lst_cont.n_t_dc, p_in_rsp -> lst_cont.n_t_dc);
        strcpy (in_resp.lst_cont.cj_stx.n_cj_r, p_in_rsp -> lst_cont.cj_stx.n_cj_r);
        strcpy (in_resp.lst_cont.cj_stx.n_st_a, p_in_rsp -> lst_cont.cj_stx.n_st_a);
        in_resp.diagn.tp_dgn = p_in_rsp -> diagn.tp_dgn;
        in_resp.diagn.id_err = p_in_rsp -> diagn.id_err;
        in_resp.diagn.ob_err = p_in_rsp -> diagn.ob_err;
        in_resp.diagn.ft_err = p_in_rsp -> diagn.ft_err;
        in_resp.diagn.es_sug = p_in_rsp -> diagn.es_sug;
        strcpy (in_resp.diagn.d_ad, p_in_rsp -> diagn.d_ad);
        in_resp.chkp = p_in_rsp -> chkp;
}

inirp_acf_rf ( )
{

        extern  struct t_est_at *apm_pgc;
        struct  t_ass_rsp       *p_ass_rsp;
        struct  t_in_rsp        *p_in_rsp;
        struct  t_msg           *buffer;
        int     idf,n;

        /* Manipulação de apontadores para construção da msg contendo a
           primitiva ASSC_RSP com a pdu INIRP */

        p_in_rsp  = (struct t_in_rsp *) & (apm_pgc -> msgp -> txt_msg[0]);
        n = sizeof (long) + sizeof(struct t_ass_rsp) + sizeof (struct t_in_resp);
        buffer = (struct t_msg *) malloc (n);
        buffer -> tip_msg = 1;
```

```
        p_ass_rsp = (struct t_ass_rsp *) & (buffer -> txt_msg [0]);
        p_ass_rsp -> id_serv = ASSC_RP;
        p_ass_rsp -> id_assc = p_in_rsp -> e_ap_r;
        strcpy (p_ass_rsp -> t_apl_r , p_in_rsp -> t_apl_r);
        strcpy (p_ass_rsp -> n_ct_apl, "FTAM");
        p_ass_rsp -> inf_us = ' ';
        p_ass_rsp -> result = 0;
        p_ass_rsp -> e_ap_r  = p_in_rsp -> e_ap_r;
        p_ass_rsp -> r_df_ctxap = 0;
        p_ass_rsp -> ql_serv = p_in_rsp -> q_s_com;
        p_ass_rsp -> req_apr = ' ';
        p_ass_rsp -> req_sess = ' ';
        p_ass_rsp -> n_pt_in_sinc = 0;
        p_ass_rsp -> atb_tkn = ' ';
        p_ass_rsp -> idf_cnx_sess = ' ';

        ++ p_ass_rsp;
        memcpy (p_ass_rsp, &in_resp, sizeof (struct t_in_resp));

        /* Enviar msg para a fila de saída inferior */

        idf = apm_pgc -> f_s_inf;

        if (msgsnd (idf, buffer, TAM_MSG, 0) == -1) {
            printf ("ERRO FATAL: Não conseguiu enviar a primitiva F_ASSC_RSP \
                                 para a fila da saída inferior");
            cfree (apm_pgc -> msgp);
            cfree (buffer);
            exit (1);
        }
        cfree (buffer);
}

f_init_pd ( )
{

        extern  struct  t_est_at   *apm_pgc;

        apm_pgc -> est_cnx = F_INIT_PD;
}
```

```
tpi21_rf ( )                        /* Testa predicado P2 do regime FTAM */
{
        extern  struct  t_est_at    *apm_pgc;
        struct  t_ass_rsp           *p_ass_cnf;
        struct  t_in_resp           *p_in_resp;

        p_ass_cnf = (struct t_ass_rsp *) &(apm_pgc -> msgp ->txt_msg [0]);

        p_in_resp = (struct t_in_resp *) ++p_ass_cnf;
        if (p_in_resp -> r_est == SUCS)
            return (1);
        else
            return (2);
}

f_inicf_pr ( )
{
        extern  struct  t_est_at    *apm_pgc;
        struct  t_ass_rsp           *p_ass_cnf;
        struct  t_in_resp           *p_in_resp;
        int     idf;

        p_ass_cnf = (struct t_ass_rsp *) &(apm_pgc -> msgp ->txt_msg [0]);

        in_cnf.id_serv = F_INICF;
        in_cnf.id_assc = p_ass_cnf -> id_assc;
        strcpy (in_cnf.t_apl_r , p_ass_cnf -> t_apl_r);
        in_cnf.e_ap_r  = p_ass_cnf -> e_ap_r;
        in_cnf.q_s_com = p_ass_cnf -> ql_serv;

        p_in_resp = (struct t_in_resp *) ++p_ass_cnf;
        in_cnf.r_est = p_in_resp -> r_est;
        in_cnf.r_ac  = p_in_resp -> r_ac;
        in_cnf.g_cont_ap = p_in_resp -> g_cont_ap;
        in_cnf.und_func  = p_in_resp -> und_func;
        in_cnf.gp_atb = p_in_resp -> gp_atb;
        in_cnf.rbck   = p_in_resp -> rbck;
        strcpy (in_cnf.lst_cont.n_t_dc,p_in_resp -> lst_cont.n_t_dc);
        strcpy (in_cnf.lst_cont.cj_stx.n_cj_r,p_in_resp -> lst_cont.cj_stx.n_cj_r);
        strcpy (in_cnf.lst_cont.cj_stx.n_st_a,p_in_resp -> lst_cont.cj_stx.n_st_a);
        in_cnf.diagn.tp_dgn = p_in_resp -> diagn.tp_dgn;
        in_cnf.diagn.id_err = p_in_resp -> diagn.id_err;
        in_cnf.diagn.ob_err = p_in_resp -> diagn.ob_err;
        in_cnf.diagn.ft_err = p_in_resp -> diagn.ft_err;
        in_cnf.diagn.es_sug = p_in_resp -> diagn.es_sug;
        strcpy (in_cnf.diagn.d_ad, p_in_resp -> diagn.d_ad);
        in_cnf.chkp = p_in_resp -> chkp;
```

```
        idf = apm_pgc -) f_s_sup;
        apm_pgc -) msgp -)tip_msg = 1;
        memcpy (apm_pgc -) msgp -)txt_msg, &in_cnf, sizeof(struct t_in_resp));

        if (msgsnd (idf,apm_pgc -) msgp,TAM_MSG,0) == -1){
            printf ("ERRO FATAL: Processo PGC não conseguiu enviar a \
                                 primitiva F_INIT_CNF para a fila de \
                                 saída superior\n");
            cfree (apm_pgc -) msgp);
            exit (1);
        }
}

init ( )
{

        extern  struct  t_est_at   *apm_pgc;

        apm_pgc -) est_cnx = INIT;
}
```

Anexo C

CÓDIGO FONTE DAS ESTRUTURAS DE DADOS (SIMPLIFICADO)

```
/*
 * Primitiva F-INITIALIZE-request
 */

struct t_in_rq {
        int  id_serv;                   /* identificador da primitiva de serviço */
        char  t_apl_ch[APL];            /* título da aplicação chamada */
        char  t_apl_ct[APL];            /* título da aplicação chamante */
        int  e_ap_ch;                   /* endereço de apresentação chamado */
        int  e_ap_ct;                   /* endereço de apresentação chamante */
        BOOLEAN g_cont_ap;              /* gerência do contexto de apresentação */
        int niv_serv;                   /* nível de serviço */
        int clas_serv;                  /* classe de serviço */
        BITSTRING und_func;             /* unidades funcionais */
        BITSTRING gp_atb;               /* grupo de atributos */
        int rbck;                       /* disponibilidade de retrocesso */
        int q_s_com;                    /* qualidade do serviço de comunicação */
        struct t_lst_cont lst_cont;     /*lista do tipo de conteúdo */
        G_STRING id_i;                  /* identidade do iniciador */
        G_STRING cont;                  /* contabilidade */
        union t_senha senh;             /* senha do sistema de arquivo */
        int  chkp;                      /* janela de checkpoint */
};

/*
 * Primitiva F-INITIALIZE-response
 */

struct t_in_rsp {
        int  id_serv;                   /* identificador da primitiva de serviço */
        int  id_assc;                   /* identificador da associação */
        int r_est;                      /* resultado de estado */
        int r_ac;                       /* resultado de ação */
        char t_apl_r[APL];              /* título da aplicação respondente */
        int  e_ap_r;                    /* endereço de apresentação respondente */
        BOOLEAN g_cont_ap;              /* gerência do contexto de apresentação */
        BITSTRING und_func;             /* unidades funcionais */
        BITSTRING gp_atb;               /* grupo de atributos */
        int rbck;                       /* disponibilidade de rollback */
        int q_s_com;                    /* qualidade do serviço de comunicação */
        struct t_lst_cont lst_cont;     /* lista do tipo de conteúdo */
        struct t_diagn diagn;           /* diagnóstico */
        int chkp;                       /* janela de checkpoint */
};
```

```
/*
 * Primitiva F-TERMINATE-request
 */

 struct t_tm_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
 };

/*
 * Primitiva  F-TERMINATE-response
 */

 struct t_tm_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         struct t_custo custo;       /* custo da operação */
 };

/*
 * Primitiva F-U-ABORT-request
 */

 struct t_u_a_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
 };

/*
 * Primitiva F-P-ABORT-indication
 */

 struct t_p_a_ind {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
 };
```

```
/*
 * Primitiva F-SELECT-request
 */

 struct t_sl_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         G_STRING nom_arq;           /* nome do arquivo */
         BITSTRING.p_acss;           /* pedido de acesso */
         struct t_sen_ac sen_ac;     /* senha de acesso */
         struct t_c_conc c_conc;     /* controle de concorrência */
         G_STRING cont;              /* contabilidade */
};

/*
 * Primitiva F-SELECT-response
 */

 struct t_sl_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_est;                  /* resultado de estado */
         int r_ac;                   /* resultado de ação */
         G_STRING nom_arq;           /* nome do arquivo */
         struct t_diagn diagn;       /* diagnóstico */
 };

/*
 * Primitiva F-DESELECT-request
 */

 struct t_ds_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
};

/*
 * Primitiva F-DESELECT-response
 */

 struct t_ds_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_custo custo;       /* custo da operação */
         struct t_diagn diagn;       /* diagnóstico */
 };
```

```
/*
 * Primitiva F-CREATE-request
 */

struct t_cr_rq {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        int supp;                   /* superposição */
        struct t_atb atb;           /* atributos */
        union t_senha sen_cr;       /* senha de criação */
        BITSTRING p_acss;           /* pedido de acesso */
        struct t_sen_ac sen_ac;     /* senha de acesso */
        union t_senha sen_dl;       /* senha de deleção */
        struct t_c_conc c_conc;     /* controle de concorrência */
        G_STRING cont;              /* contabilidade */
};

/*
 * Primitiva F-CREATE-response
 */

struct t_cr_rsp {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        int r_est;                  /* resultado de estado */
        int r_ac;                   /* resultado de ação */
        struct t_atb atb;           /* atributos */
        struct t_diagn diagn;       /* diagnóstico */
};

/*
 * Primitiva F-DELETE-request
 */

struct t_dl_rq {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        union t_senha sen_dl;       /* senha de deleção */
};
```

```
/*
 * Primitiva F-DELETE-response
 */

struct t_dl_rsp {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        int r_ac;                   /* resultado de ação */
        struct t_custo custo;       /* custo da operação */
        struct t_diagn diagn;       /* diagnóstico */
};

/*
 * Primitiva F-READ-ATTRIB-request
 */

struct t_r_at_rq {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        BITSTRING n_atb;            /* nomes dos atributos */
};

/*
 * Primitiva F-READ-ATTRIB-response
 */

struct t_r_a_rsp {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        int r_ac;                   /* resultado de ação */
        struct t_atb atb;           /* atributos */
        struct t_diagn diagn;       /* diagnóstico */
};

/*
 * Primitiva F-OPEN-request
 */

struct t_op_rq {
        int id_serv;                /* identificador da primitiva de serviço */
        int id_assc;                /* identificador da associação */
        BITSTRING m_prc;            /* modo de processamento */
        struct t_t_cont t_cont;     /* tipo de conteúdo */
        struct t_c_conc c_conc;     /* controle de concorrência */
        int id_at;                  /* identificador de atividade */
        int m_rec;                  /* modo de recuperação */
};
```

```
/*
 * Primitiva F-OPEN-response
 */

 struct t_op_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_est;                  /* resultado de estado */
         int r_ac;                   /* resultado de ação */
         union t_tip_cont tip_cont;  /* tipo de conteúdo */
         struct t_c_conc c_conc;     /* controle de concorrência */
         struct t_diagn diagn;       /* diagnóstico */
         int m_rec;                  /* modo de recuperação */
 };

/*
 * Primitiva F-CLOSE-request
 */

 struct t_cl_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
 };

/*
 * Primitiva F-CLOSE-response
 */

 struct t_cl_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
 };

/*
 * Primitiva F-LOCATE-request
 */

 struct t_lc_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         struct t_udaa udaa;         /* unidade de dado de acesso a arquivo */
 };
```

```
/*
 * Primitiva  F-LOCATE-response
 */

 struct t_lc_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_udaa udaa;         /* unidade de dado de acesso a arquivo */
         struct t_diagn diagn;       /* diagnóstico */
 };

/*
 * Primitiva F-ERASE-request
 */

 struct t_er_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         struct t_udaa udaa;         /* unidade de dado de acesso a arquivo */
 };

/*
 * Primitiva F-ERASE-response
 */

 struct t_er_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
 };

/*
 * Primitiva F-READ-request
 */

 struct t_rd_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         struct t_udaa udaa;         /* unidade de dado de acesso a arquivo */
         int ct_ac;                  /* contexto de acesso */
 };
```

```
/*
 * Primitiva F-WRITE-request
 */

 struct t_wr_rq {
        int id_serv;                    /* identificador da primitiva de serviço */
        int id_assc;                    /* identificador da associação */
        int op_ud;                      /* operação com udaa */
        struct t_udaa udaa;             /* unidade de dado de acesso a arquivo */
        int ct_ac;                      /* contexto de acesso */
};

/*
 * Primitiva F-DATA-request
 */

struct t_dat_rq {
        int id_serv;                    /* identificador da primitiva de serviço */
        int id_assc;                    /* identificador da associação */
        int val_dado;                   /* valor de dado */
 };

/*
 * Primitiva F-DATA-END-request
 */

 struct t_d_e_rq {
         int id_serv;                   /* identificador da primitiva de serviço */
         int id_assc;                   /* identificador da associação */
         int r_ac;                      /* resultado de ação */
         struct t_diagn diagn;          /* diagnóstico */
 };

/*
 * Primitiva F-TRANSFER-END-request
 */

 struct t_t_e_rq {
         int id_serv;                   /* identificador da primitiva de serviço */
         int id_assc;                   /* identificador da associação */
 };
```

```
/*
 * Primitiva F-TRANSFER-END-response
 */

 struct t_t_e_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
};

/*
 * Primitiva F-CANCEL-request
 */

struct t_ca_rq {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
};

/*
 * Primitiva F-CANCEL-response
 */

struct t_ca_rsp {
         int id_serv;                /* identificador da primitiva de serviço */
         int id_assc;                /* identificador da associação */
         int r_ac;                   /* resultado de ação */
         struct t_diagn diagn;       /* diagnóstico */
};
```

```
/* DECLARAÇÃO DAS UDP'S.
 */

/*
 * UDP'S DE ESTABELECIMENTO DO REGIME FTAM
 */

/*
 * UDP F-INITIALIZE-request
 */

struct t_lst_cont {
        G_STRING n_t_dc;            /* nome do tipo do documento */
        struct {
            G_STRING n_cj_r;        /* nome do conjunto de restrições */
            G_STRING n_st_a;        /* nome da sintaxe abstrata */
        } cj_stx;
};

union t_versao {
       BITSTRING p_vers;
       int p_rev;
};

union t_senha {
       G_STRING g_senh;
       O_STRING o_senh;
};

struct t_in_req {
        int id_pdu;                 /* identificador da pdu */
        union t_versao versao;      /* versão do protocolo */
        BOOLEAN g_cont_ap;          /* gerência do contexto de apresentação */
        int niv_serv;               /* nível de serviço */
        int clas_serv;              /* classe de serviço */
        BITSTRING und_func;         /* unidades funcionais */
        BITSTRING gp_atb;           /* grupo de atributos */
        int rbck;                   /* disponibilidade de retrocesso */
        struct t_lst_cont lst_cont;/* lista do tipo de conteúdo */
        G_STRING id_i;              /* identidade do iniciador */
        G_STRING cont;              /* contablidade */
        union t_senha senh;         /* senha do sistema de arquivo */
        int chkp;                   /* janela de checkpoint */
};
```

```
/*
 * UDP F-INITIALIZE-response
 */

 struct t_diagn {
         int tp_dgn;              /* tipo de diagnóstico */
         int id_err;              /* identificação do erro */
         int ob_err;              /* observador do erro */
         int ft_err;              /* fonte do erro */
         int es_sug;              /* espera sugerida */
         G_STRING d_ad;           /* detalhes adicionais */
 };

 struct t_in_resp {
         int id_pdu;              /* identificador da UDP */
         int r_est;               /* resultado de estado */
         int r_ac;                /* resultado de ação */
         union t_versao versao;   /* versão do protocolo */
         BOOLEAN g_cont_ap;       /* gerência do contexto de apresentação */
         BITSTRING und_func;      /* unidades funcionais */
         BITSTRING gp_atb;        /* grupo de atributos */
         int rbck;                /* disponibilidade de retrocesso */
         struct t_lst_cont lst_cont;   /* lista tipo de conteúdo */
         struct t_diagn diagn;         /* diagnóstico */
         int chkp;                     /* janela de checkpoint */
 };

/*
 * UDP F-TERMINATE-request
 */

 struct t_tm_req {
         int id_pdu;               /* identidade da UDP */
 };

/*
 * UDP F-TERMINATE-response
 */

 struct t_custo {
         G_STRING id_rec;          /* identificador do recurso */
         G_STRING ud_cust;         /* unidade de custo */
         int valor;                /* valor do custo */
 };
```

```
struct t_tm_resp {
        int id_pdu;             /* identificador da pdu */
        struct t_custo custo;   /* custo da operação */
};

/*
 * UDP F-U-ABORT-request
 */

struct t_u_ab_req {
        int id_pdu;             /* identificador da pdu */
        int r_ac;               /* resultado de ação */
        struct t_diagn diagn;   /* diagnóstico */
};

/*
 * UDP F-P-ABORT-request
 */

struct t_p_ab_req {
        int id_pdu;             /* identificador da pdu */
        int r_ac;               /* resultado de ação */
        struct t_diagn diagn;   /* diagnóstico */
};


/*
 * UDP'S DO REGIME DE ARQUIVO
 */

/*
 * UDP F-SELECT-request
 */

struct t_sen_ac {
        union t_senha lt;       /* senha para leitura */
        union t_senha ins;      /* senha para inserção */
        union t_senha subs;     /* senha para substituição */
        union t_senha apg;      /* senha para apagamento */
        union t_senha acr;      /* senha para acréscimo */
        union t_senha l_at;     /* senha para leitura de atributos */
        union t_senha m_at;     /* senha para mudança de atributos */
        union t_senha d_snh;    /* senha para deleção de senha */
};
```

```
struct t_c_conc {
        int lt;                 /* bloqueio para leitura */
        int ins;                /* bloqueio para inserção */
        int subs;               /* bloqueio para substituição */
        int apg;                /* bloqueio para apagamento */
        int acr;                /* bloqueio para acréscimo */
        int l_at;               /* bloqueio para leitura de atributos */
        int m_at;               /* bloqueio para mudança de atributos */
        int d_aq;               /* bloqueio para deleção de arquivo */
};

struct t_sl_req {
        int id_pdu;             /* identificador da pdu */
        G_STRING nom_arq;       /* nome do arquivo */
        BITSTRING p_acss;       /* pedido de acesso */
        struct t_sen_ac sen_ac; /* senha de acesso */
        struct t_c_conc c_conc; /* controle de concorrência */
        G_STRING cont;          /* contabilidade */
};

/*
 * UDP F-SELECT-response
 */

struct t_sl_resp {
        int id_pdu;             /* identificador da pdu */
        int r_est;              /* resultado de estado */
        int r_ac;               /* resultado de ação */
        G_STRING nom_arq;       /* nome do arquivo */
        struct t_diagn diagn;   /* diagnóstico */
};

/*
 * UDP F-DESELECT-request
 */

struct t_ds_req {
        int id_pdu;             /* identificador da pdu */
};
```

```
/*
 * UDP F-DESELECT-response
 */

 struct t_ds_resp {
         int id_pdu;              /* identificador da pdu */
         int r_ac;                /* resultado de ação */
         struct t_custo custo;    /* custo da operação */
         struct t_diagn diagn;    /* diagnóstico */
 };

/*
 * UDP F-CREATE-request
 */

/* estrutura tipo identidade */

 struct t_idt {
         int s_v_dsp;             /* sem valor disponível */
         G_STRING v_rl;           /* identidade (valor real) */
 };

/* estrutura tipo data */

 struct t_dat {
         int s_v_dsp;             /* sem valor disponível */
         O_STRING v_rl;           /* valor real */
 };

/* estrutura tipo tamanho */

 struct t_tam {
         int s_v_dsp;             /* sem valor disponível */
         int v_rl;                /* valor real */
 };

/* estrutura tipo controle de acesso  */

 struct t_ctl_ac {
         BITSTRING p_acss;        /* pedido de acesso */
         G_STRING id_u;           /* identidade do usuário */
         struct t_sen_ac s_acs;   /* senha de acesso */
         union {
             G_STRING e_alc;      /* entidade alocada */
             EXTERNAL pdr;        /* padronizada */
         } e_apl;                 /* título da entidade de aplicação */
 };
```

```
/* estrutura tipo atributo */

union  t_tip_cont {
        G_STRING n_t_dc;
        struct {
             G_STRING n_cj_r;
             G_STRING n_st_ab;
        } crest_stab;              /* conjunto de restrições e sintaxe abstrata */
};

struct t_atb {

        /* Grupo Núcleo */

        G_STRING nome_arq;         /* nome do arquivo */
        union t_tip_cont tip_cont; /* tipo do conteúdo */

        /* Grupo de Armazenamento */

        G_STRING cont;             /* contabilidade de armazenamento */
        struct t_dat d_h_cr;       /* data e hora da criação */
        struct t_dat d_h_md;       /* data e hora da última modificação */
        struct t_dat d_h_lt;       /* data e hora da última leitura */
        struct t_dat d_h_at;       /* data e hora da última modificação de atributo */
        struct t_idt id_crd;       /* identidade do criador */
        struct t_idt id_u_md;      /* identidade do último modificador */
        struct t_idt id_u_lt;      /* identidade do último leitor */
        struct t_idt id_m_at;      /* identidade do último modificador de atributos */
        struct t_tam dsp_arq;      /* disponibilidade do arquivo */
        struct {
             int s_v_dsp;
             BITSTRING v_rl;
        } a_pmt;                   /* ações permitidas */
        struct t_tam tam_a;        /* tamanho do arquivo */
        struct t_tam tam_f;        /* tamanho futuro do arquivo */

        /* Grupo de Segurança */

        struct {
             int s_v_dsp;
             struct t_ctl_ac ctl_ac;
        } c_acs;                   /* controle de acesso */
```

```
        struct {
            int s_v_dsp;
            G_STRING v_rl;
        } n_cif;                    /* nome de cifragem (criptografia) */
        struct {
            int s_v_dsp;
            G_STRING v_rl;
        } q_lg;                     /* qualificações legais */

       /* Grupo Privado */

        struct {
            int s_v_dsp;
            int s_ab_n;             /* sintaxe abstrata não suportada */
            EXTERNAL v_rl;
        } u_pr;                     /* uso privativo */
};

struct t_cr_req {
        int id_pdu;                 /* identificador da pdu */
        int supp;                   /* superposição */
        struct t_atb atb;           /* atributos */
        union t_senha sen_cr;       /* senha de criação */
        BITSTRING p_acss;           /* pedido de acesso */
        struct t_sen_ac sen_ac;       /* senha de acesso */
        struct t_c_conc c_conc;     /* controle de concorrência */
        G_STRING cont;              /* contabilidade */
};

/*
 * UDP F-CREATE-response
 */

struct t_cr_resp {
        int id_pdu;                 /* identificador da pdu */
        int r_est;                  /* resultado de estado */
        int r_ac;                   /* resultado de ação */
        struct t_atb atb;           /* atributos */
        struct t_diagn diagn;       /* diagnóstico */
};
```

```
/*
 * UDP F-DELETE-request
 */

 struct t_dl_req {
         int id_pdu;                /* identificador da pdu */
         union t_senha sen_dl;      /* senha de deleção */
 };

/*
 * UDP F-DELETE-response
 */

 struct t_dl_resp {
         int id_pdu;                /* identificador da pdu */
         int r_ac;                  /* resultado de ação */
         struct t_custo custo;      /* custo da operação */
         struct t_diagn diagn;      /* diagnóstico */
 };

/*
 * UDP F-READ-ATTRIB-request
 */

 struct t_r_a_rq {
         int id_pdu;                /* identificador da pdu */
         BITSTRING n_atb;           /* nomes dos atributos */
 };

/*
 * UDP F-READ-ATTRIB-response
 */

 struct t_r_a_resp {
         int id_pdu;                /* identificador da pdu */
         int r_ac;                  /* resultado de ação */
         struct t_atb atb;          /* atributos */
         struct t_diagn diagn;      /* diagnóstico */
 };

/*
 * UDP F-OPEN-request
 */

 struct t_t_cont {
         int desc;                  /* desconhecido */
         union t_tip_cont tip_cont;
         };                         /* tipo de conteúdo */
```

```
struct t_op_req {
        int id_pdu;                 /* identificador da pdu */
        BITSTRING m_prc;            /* modo de processamento */
        struct t_t_cont t_cont;     /* tipo de conteúdo */
        struct t_c_conc c_conc;     /* controle de concorrência */
        int id_at;                  /* identificador de atividade */
        int m_rec;                  /* modo de recuperação */
        O_STRING r_ctx;             /* remove-contexts */
};
/*
 * UDP F-OPEN-response
 */

struct t_op_resp {
        int id_pdu;                 /* identificador da pdu */
        int r_est;                  /* resultado de estado */
        int r_ac;                   /* resultado de ação */
        union t_tip_cont tip_cont; /* tipo de conteúdo */
        struct t_c_conc c_conc;     /* controle de concorrência */
        struct t_diagn diagn;       /* diagnóstico */
        int m_rec;                  /* modo de recuperação */
};

/*
 * UDP F-CLOSE-request
 */

struct t_cl_req {
        int id_pdu;                 /* identificador da pdu */
};

/*
 * UDP F-CLOSE-response
 */

struct t_cl_resp {
        int id_pdu;                 /* identificador da pdu */
        int r_ac;                   /* resultado de ação */
        struct t_diagn diagn;       /* diagnóstico */
};
```

```
/*
 * UDP F-LOCATE-request
 */

union  t_n_nd {
        G_STRING cd_ftam;          /* código FTAM */
        EXTERNAL cd_usr;           /* código do usuário */
};


struct t_r_udaa {
        int pr_ul;                 /* primeiro ou último nodo */
        int rel;                   /* relativo - próximo, anterior ou nodo corrente */
        int in_f;                  /* início ou fim */
        union t_n_nd n_nd;         /* nome único */
        union t_n_nd l_nm;         /* lista de nome */
        int n_udaa;                /* número da udaa */
};

struct t_udaa {
        struct t_r_udaa r_udaa;    /* referência à udaa */
        int n_nvl;                 /* numero do nível */
};

struct t_lc_req {
        int id_pdu;                /* identificador da pdu */
        struct t_udaa udaa;        /* unidade de dado de acesso a arquivo */
};

/*
 * UDP F-LOCATE-response
 */

struct t_lc_resp {
        int id_pdu;                /* identificador da pdu */
        int r_ac;                  /* resultado de ação */
        struct t_udaa udaa;        /* unidade de dado de acesso a arquivo */
        struct t_diagn diagn;      /* diagnóstico */
};

/*
 * UDP F-ERASE-request
 */

struct t_er_req {
        int id_pdu;                /* identificador da pdu */
        struct t_udaa udaa;        /* unidade de dado de acesso a arquivo */
};
```

```
/*
 * UDP F-ERASE-response
 */

 struct t_er_resp {
         int id_pdu;              /* identificador da pdu */
         int r_ac;                /* resultado de ação */
         struct t_diagn diagn;    /* diagnóstico */
 };

/*
 * UDP'S DE TRANSFERÊNCIA DE DADOS
 */

/*
 * UDP F-READ-request
 */

 struct t_rd_req {
         int id_pdu;              /* identificador da pdu */
         struct t_udaa udaa;      /* unidade de dado de acesso a arquivo */
         int ct_ac;               /* contexto de acesso */
 };

 /*
  * UDP F-WRITE-request
  */

  struct t_wr_req {
          int id_pdu;             /* identificador da pdu */
          int op_ud;              /* operação com udaa */
          struct t_udaa udaa;     /* unidade de dado de acesso a arquivo */
          int ct_ac;              /* contexto de acesso */
  };

/*
 * UDP F-DATA-END-request
 */

 struct t_dt_e_req {
         int id_pdu;              /* identificador da pdu */
         int r_ac;                /* resultado de ação */
         struct t_diagn diagn;    /* diagnóstico */
 };
```

```
/*
 * UDP F-TRANSFER-END-request
 */

 struct t_t_e_req {
         int id_pdu;                /* identificador da pdu */
 };

/*
 * UDP F-TRANSFER-END-response
 */

 struct t_tf_e_resp {
         int id_pdu;                /* identificador da pdu */
         int r_ac;                  /* resultado de ação */
         struct t_diagn diagn;      /* diagnóstico */
 };
/*
 * UDP F-CANCEL-request
 */

 struct t_ca_req {
         int id_pdu;                /* identificador da pdu */
         int r_ac;                  /* resultado de ação */
         struct t_diagn diagn;      /* diagnóstico */
 };

/*
 * UDP F-CANCEL-response
 */

 struct t_ca_resp {
         int id_pdu;                /* identificador da pdu */
         int r_ac;                  /* resultado de ação */
         struct t_diagn diagn;      /* diagnóstico */
 };
```

```
/*
 * DEFINIÇÃO E INICIALIZAÇÃO DA TABELA DE TRANSIÇÃO
 */

#include        <stdio.h>

int s_pred    ( ),      tpi11_rf  ( ),     tpi21_rf  ( ),
    tpr11_rf  ( ),      tpr21_rf  ( ),     tpi21_ra  ( ),
    tpi61_ra  ( ),      tpi141_ra ( ),     tpi142_ra ( ),
    tpi143_ra ( ),      tpi152_ra ( ),     tpi212_ra ( ),
    tpi201_ra ( ),      tpi202_ra ( ),     tpi203_ra ( ),
    tpi302_ra ( ),      tpi303_ra ( ),     tpi304_ra ( ),
    tpi305_ra ( ),      tpi306_ra ( ),     tpi307_ra ( ),
    tpr21_ra  ( ),      tpr61_ra  ( ),     tpr141_ra ( ),
    tpr142_ra ( ),      tpr143_ra ( ),     tpr144_ra ( ),
    tpr201_ra ( ),      tpr202_ra ( ),     tpr203_ra ( ),
    tpr204_ra ( ),      tpr332_ra ( ),     tpr333_ra ( ),
    tpr334_ra ( ),      tpr335_ra ( );

struct t_tab_trans {
        int (*f_pred [8]) ();          /* arranjo de apontadores de funções
                                          retornando um inteiro */
        int tab_fsd;                   /* número de entrada na tabela de função
                                          de saída */
} tab_trs [ ] [62] = {

        /* Gerencia do Regime FTAM - Entidade Iniciando */

        {{tpi11_rf,NULL},1}, /* testa predicados da entrada 1 da máquina de estados */
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
        {{s_pred,NULL},0},
```

```
{{s_pred,NULL},0},
{{tpi21_rf,NULL},2}, /* testa predicados da entrada 2 da máquina de estados */
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
```

```
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},3},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},
```

```
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},4},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},{{s_pred,NULL},0},
{{s_pred,NULL},0},{{s_pred,NULL},0},
```

```
{{s_pred,NULL},0},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},{{s_pred,NULL},5},
{{s_pred,NULL},5},{{s_pred,NULL},5},

.
.
.
.
.
```

```
/*
 * DEFINIÇÃO E INICIALIZAÇÃO DA TABELA DE FUNÇÃO DE SAÍDA
 */

#include        (stdio.h)

   int  err_prtcl    ( ), inirq_pdu      ( ), inirq_ac1_rf ( ),
        ac3_rf       ( ), init_pd        ( ), f6_inicf_pr  ( ),
        ac6_rf       ( ), closed         ( ), f_inicf_pr   ( ),
        init         ( ), terrq_pdu      ( ), term_pd      ( ),
        f_tercf_pr   ( ), f_pabin_pr     ( ), ac5_rf       ( ),
        f_uabin_pr   ( ), uabrq_pdu      ( ), f_iniin_pr   ( ),
        f_init_pd    ( ), inirp6_pdu     ( ), inirp_pdu    ( ),
        f_terin_pr   ( ), f_term_pd      ( ), terrp_pdu    ( ),
        selrq2_pdu   ( ), ac2_ra         ( ), select_pd    ( ),
        f_selcf_pr   ( ), f8_selcf_pr    ( ), select       ( ),
        ac8_ra       ( ), desrq2_pdu     ( ), desel_pd     ( ),
        f_descf_pr   ( ), crerq2_pdu     ( ), create_pd    ( ),
        f_crecf_pr   ( ), delrq2_pdu     ( ), delete_pd    ( ),
        f_delcf_pr   ( ), ratrq2_pdu     ( ), f_ratcf_pr   ( ),
        catrq2_pdu   ( ), chg_att_pd     ( ), f_catcf_pr   ( ),
        opnrq2_pdu   ( ), ac31_ra        ( ), open_pd      ( ),
        f_opncf_pr   ( ), dxfridle       ( ), ac15_ra      ( ),
        p_altin_pd   ( ), p_altrp_pr     ( ), ac16_ra      ( ),
        clorq2_pdu   ( ), close_pd       ( ), f_clocf_pr   ( ),
        recrq2_pdu   ( ), ac31_ra        ( ), recover_pd   ( ),
        f_reccf_pr   ( ), ac8_ra         ( ), dxfridle_rec ( ),
        ac10_ra      ( ), p_altin_rec_pd ( ), p_altrp_pr   ( ),
        locrq2_pdu   ( ), locate_pd      ( ), f_loccf_pr   ( ),
        erarq2_pdu   ( ), erase_pd       ( ), f_eracf_pr   ( ),
        ac32_ra      ( ), grouping       ( ), ac33_ra      ( ),
        ac34_ra      ( ), group_pd       ( ), ac35_ra      ( ),
        ac36_ra      ( ), ac37_ra        ( ), f_selin_pr   ( ),
        f_select_pd  ( ), selrp2_pdu     ( ), f_desin_pr   ( ),
        f_deselect_pd( ), desrp2_pdu     ( ), f_crein_pr   ( ),
        f_create_pd  ( ), crerp2_pdu     ( ), f_delin_pr   ( ),
        f_delete_pd  ( ), delrp2_pdu     ( ), f_ratin_pr   ( ),
        ratrp2_pdu   ( ), f_read_att_pd  ( ), f_catin_pr   ( ),
        f_chg_att_pd ( ), catrp2_pdu     ( ), f_opnin_pr   ( ),
        ac31_ra      ( ), f_open_pd      ( ), opnrp2_pdu   ( ),
        p_altrq_pr   ( ), p_altcf_pd     ( ), f_cloin_pr   ( ),
        f_close_pd   ( ), clorp2_pdu     ( ), f_recin_pr   ( ),
        f_recover_pd ( ), recrp2_pdu     ( ), p_alcf_rec_pd( ),
        f_locin_pr   ( ), f_locate_pd    ( ), locrp2_pdu   ( ),
        f_erain_pr   ( ), f_erase_pd     ( ), erarp2_pdu   ( ),
```

```
    ac40_ra      ( ), ac43_ra       ( ), ac42_ra      ( ),
    f_group_pd   ( ), ac38_ra       ( ), ac39_ra      ( ),
    ac41_ra      ( ), p_altrq_pr    ( ), p_altcf_pd   ( ),
    ac41_md      ( ), rearq2_pdu    ( ), ac2_md       ( ),
    ac24_md      ( ), ac48_md       ( ), rea_symin_pd ( ),
    read         ( ), wrtrq2_pdu    ( ), p_symrq_pr   ( ),
    wrt_symcf_pd ( ), write         ( ), f_trecf_pr   ( ),
    p18_datrq_pr ( ), ac18_md       ( ), daterq2_pdu  ( ),
    write_end    ( ), f_datin_pr    ( ), f_daein_pr   ( ),
    read_end     ( ), trerq2_pdu    ( ), r_xfr_end    ( ),
    w_xfr_end    ( ), ac15_md       ( ), p_token_pd   ( ),
    f_trecf_pr   ( ), ac22_md       ( ), canrq26_pdu  ( ),
    ac26_md      ( ), cancel_pd     ( ), ac21_md      ( ),
    f_cancf_pr   ( ), ac23_md       ( ), f_canin_pr   ( ),
    f_cancel_pd  ( ), canrp26_pdu   ( ), ac49_md      ( ),
    ac25_md      ( ), ac29_md       ( ), ac27_md      ( ),
    p_symrq_pr   ( ), ac30_md       ( ), f_chkcf_pr   ( ),
    f_chkin_pr   ( ), p_symrp_pr    ( ), ac22_md      ( ),
    ac50_md      ( ), resrq_pdu     ( ), rrestart_pd  ( ),
    resrq_pdu    ( ), wrestart_pd   ( ), ac21_md      ( ),
    f_rescf_pr   ( ), f_resin_pr    ( ), f_rrestart_pd( ),
    f_wrestart_pd( ), resrp_pdu     ( ), ac46_md      ( ),
    p_symrp_pr   ( ), ac47_md       ( ), res_symin_pd ( ),
    can_symin_pd ( ), res_symcf_pd  ( ), can_symcf_pd ( ),
    rearq_pdu    ( ), ac44_md       ( ), ac45_md      ( ),
    rea_symin_pd ( ), wrtrq_pdu     ( ), p_symrq_pr   ( ),
    ac45_md      ( ), wrt_symcf_pd  ( ), ac30_md      ( ),
    rest_can_pd  ( ), f_cancf_pr    ( ), canrp_pdu    ( ),
    ac48_md      ( ), f_reain_pr    ( ), f_wrtin_pr   ( ),
    wrt_symin_pd ( ), p_symrq_pd    ( ), rea_symcf_pd ( ),
    p_datrq_pd   ( ), daerq_pdu     ( ), f_trein_pr   ( ),
    f_r_xfer_end ( ), f_w_xfer_end  ( ), trerp2_pdu    ( ),
    canrq_md     ( ), f_cancf_pr    ( ), ac23_md      ( ),
    f_canin_pr   ( ), canrp_pdu     ( ), f_reccf_pr   ( ),
    f8_crecf_pr  ( ), f8_opncf_pr   ( ), f8_reccf_pr  ( ),
    selrp82_pdu  ( ), crerp82_pdu   ( ), opnrp82_pdu  ( ),
    recrp82_pdu  ( ), inirp_aci_rf  ( ), uabrq_aci_rf ( ),
    inirp6_aci_rf( ), terrq_aci_rf  ( ), terrp_aci_rf ( );

int (*f_ac[3][9][7]) () = {
```

```
/* função para envio de mensagem de erro do protocolo */

{{NULL},{err_prtcl,NULL},
                NULL},                                  /* entrada 0 */

/* Gerência do Regime FTAM - Entidade Iniciando */

{{NULL},{inirq_pdu,inirq_aci_rf,ac3_rf,init_pd,NULL},   /* lista proc. #1.1 */
 {f6_inicf_pr,closed,NULL},                             /* lista proc. #1.2 */
                                NULL},                  /* entrada 1 */
{{NULL},{f_inicf_pr,init,NULL},                         /* lista proc. #2.1 */
 {f6_inicf_pr,closed,NULL},                             /* lista proc. #2.2 */
                                NULL},                  /* entrada 2 */
{{NULL},{terrq_pdu,terrq_aci_rf,term_pd,NULL},          /* lista proc. #3.1 */
                                NULL},                  /* entrada 3 */
{{NULL},{f_tercf_pr,closed,NULL},                       /* lista proc. #4.1 */
                            NULL},                      /* entrada 4 */
{{NULL},{f_pabin_pr,ac5_rf,closed,NULL},                /* lista proc. #5.1 */
                                NULL},                  /* entrada 5 */
{{NULL},{f_uabin_pr,closed,NULL},                       /* lista proc. #6.1 */
                            NULL},                      /* entrada 6 */
{{NULL},{f_pabin_pr,closed,NULL},                       /* lista proc. #7.1 */
                            NULL},                      /* entrada 7 */
{{NULL},{uabrq_pdu,uabrq_aci_rf,closed,NULL},                   /* lista proc. #8.1 */
                                NULL},                  /* entrada 8 */
/* Gerencia do Regime FTAM - Entidade Respondendo */

{{NULL},{f_iniin_pr,ac3_rf,f_init_pd,NULL},             /* lista proc. #1.1 */
 {inirp6_pdu,inirp6_aci_rf,closed,NULL},                        /* lista proc. #1.2 */
                                    NULL},              /* entrada 9 */
{{NULL},{inirp_pdu,inirp_aci_rf,init,NULL},                     /* lista proc. #2.1 */
 {inirp6_pdu,inirp6_aci_rf,closed,NULL},                        /* lista proc. #2.2 */
                                    NULL},              /* entrada 10 */
{{NULL},{f_terin_pr,f_term_pd,NULL},                    /* lista proc. #3.1 */
                                NULL},                  /* entrada 11 */
{{NULL},{terrp_pdu,terrp_aci_rf,closed,NULL},                   /* lista proc. #4.1 */
                                NULL},                  /* entrada 12 */

/* Gerencia do Regime de Arquivo - Entidade Iniciando */

{{NULL},{selrq2_pdu,select_pd,NULL},                    /* lista proc. #1.1 */
                                    NULL},              /* entrada 13 */
{{NULL},{f_selcf_pr,select,NULL},                       /* lista proc. #2.1 */
 {f8_selcf_pr,init,NULL},                               /* lista proc. #2.2 */
                                NULL},                  /* entrada 14 */
{{NULL},{desrq2_pdu,desel_pd,NULL},                     /* lista proc. #3.1 */
                                NULL},                  /* entrada 15 */
```

```
{{NULL},{f_descf_pr,init,NULL},                    /* lista proc. #4.1 */
                        NULL},                     /* entrada 16 */
{{NULL},{crerq2_pdu,create_pd,NULL},               /* lista proc. #5.1 */
                                    NULL},         /* entrada 17 */
{{NULL},{f_crecf_pr,select,NULL},                  /* lista proc. #6.1 */
 {f8_crecf_pr,init,NULL},                          /* lista proc. #6.2 */
                                 NULL},            /* entrada 18 */
{{NULL},{delrq2_pdu,delete_pd,NULL},               /* lista proc. #7.1 */
                                    NULL},         /* entrada 19 */
{{NULL},{f_delcf_pr,init,NULL},                    /* lista proc. #8.1 */
                        NULL},                     /* entrada 20 */
{{NULL},{ratrq2_pdu,NULL},                         /* lista proc. #9.1 */
                         NULL},                    /* entrada 21 */
{{NULL},{f_ratcf_pr,select,NULL},                  /* lista proc. #10.1 */
                         NULL},                    /* entrada 22 */
{{NULL},{catrq2_pdu,chg_att_pd,NULL},              /* lista proc. #11.1 */
                                    NULL},         /* entrada 23 */
{{NULL},{f_catcf_pr,select,NULL},                  /* lista proc. #12.1 */
                         NULL},                    /* entrada 24 */
{{NULL},{opnrq2_pdu,ac31_ra,open_pd,NULL},         /* lista proc. #13.1 */
                                       NULL},      /* entrada 25 */
{{NULL},{f8_opncf_pr,select,NULL},                 /* lista proc. #14.1 */
 {f_opncf_pr,dxfridle,NULL},                       /* lista proc. #14.2 */
 {ac15_ra,p_altin_pd,NULL},                        /* lista proc. #14.3 */
                          NULL},                   /* entrada 26 */
{{NULL},{p_altrp_pr,ac16_ra,NULL},                 /* lista proc. #15.1 */
 {f_opncf_pr,dxfridle,NULL},                       /* lista proc. #15.2 */
                        NULL},                     /* entrada 27 */
{{NULL},{clorq2_pdu,close_pd,NULL},                /* lista proc. #17.1 */
                                    NULL},         /* entrada 28 */
{{NULL},{f_clocf_pr,select,NULL},                  /* lista proc. #18.1 */
                         NULL},                    /* entrada 29 */
{{NULL},{recrq2_pdu,ac31_ra,recover_pd,NULL},      /* lista proc. #19.1 */
                                       NULL},      /* entrada 30 */
{{NULL},{f8_reccf_pr,init,NULL},                   /* lista proc. #20.1 */
 {f_reccf_pr,dxfridle_rec,NULL},                   /* lista proc. #20.2 */
 {ac15_ra,p_altin_rec_pd,NULL},                    /* lista proc. #20.3 */
                                 NULL},            /* entrada 31 */
{{NULL},{p_altrp_pr,ac16_ra,NULL},                 /* lista proc. #21.1 */
 {f_reccf_pr,dxfridle_rec,NULL},                   /* lista proc. #21.2 */
                                  NULL},           /* entrada 32 */
{{NULL},{locrq2_pdu,locate_pd,NULL},               /* lista proc. #23.1 */
                          NULL},                   /* entrada 33 */
{{NULL},{f_loccf_pr,dxfridle,NULL},                /* lista proc. #24.1 */
                       NULL},                      /* entrada 34 */
{{NULL},{erarq2_pdu,erase_pd,NULL},                /* lista proc. #25.1 */
                                    NULL},         /* entrada 35 */
```

```
{{NULL},{f_eracf_pr,dxfridle,NULL},                    /* lista proc. #26.1 */
                                NULL},                 /* entrada 36 */
{{NULL},{ac32_ra,grouping,NULL},                       /* lista proc. #27.1 */
                               NULL},                  /* entrada 37 */
{{NULL},{ac33_ra,NULL},                                /* lista proc. #28.1 */
                    NULL},                             /* entrada 38 */
{{NULL},{ac34_ra,group_pd,NULL},                       /* lista proc. #29.1 */
                               NULL},                  /* entrada 39 */
{{NULL},{ac35_ra,NULL},                                /* lista proc. #30.1 */
 {ac36_ra,NULL},                                       /* lista proc. #30.2 */
 {ac37_ra,NULL},                                       /* lista proc. #30.3 */
 {init,NULL},                                          /* lista proc. #30.4 */
 {dxfridle,NULL},                                      /* lista proc. #30.5 */
 {ac15_ra,p_altin_rec_pd,NULL},                        /* lista proc. #30.6 */
 {select,NULL},                                        /* lista proc. #30.7 */
          NULL},                                       /* entrada 40 */

/* Gerência do Regime de Arquivo - Entidade Respondendo */

{{NULL},{f_selin_pr,f_select_pd,NULL},                 /* lista proc. #1.1 */
                                      NULL},           /* entrada 41 */
{{NULL},{selrp2_pdu,select,NULL},                      /* lista proc. #2.1 */
 {selrp82_pdu,init,NULL},                              /* lista proc. #2.2 */
                                    NULL},             /* entrada 42 */
{{NULL},{f_desin_pr,f_deselect_pd,NULL},               /* lista proc. #3.1 */
                                    NULL},             /* entrada 43 */
{{NULL},{desrp2_pdu,init,NULL},                        /* lista proc. #4.1 */
                                    NULL},             /* entrada 44 */
{{NULL},{f_crein_pr,f_create_pd,NULL},                 /* lista proc. #5.1 */
                                    NULL},             /* entrada 45 */
{{NULL},{crerp2_pdu,select,NULL},                      /* lista proc. #6.1 */
 {crerp82_pdu,init,NULL},                              /* lista proc. #6.2 */
                                    NULL},             /* entrada 46 */
{{NULL},{f_delin_pr,f_delete_pd,NULL},                 /* lista proc. #7.1 */
                                    NULL},             /* entrada 47 */
{{NULL},{delrp2_pdu,init,NULL},                        /* lista proc. #8.1 */
                                    NULL},             /* entrada 48 */
{{NULL},{f_ratin_pr,f_read_att_pd,NULL},               /* lista proc. #9.1 */
                                     NULL},            /* entrada 49 */
{{NULL},{ratrp2_pdu,select,NULL},                      /* lista proc. #10.1 */
                                    NULL},             /* entrada 50 */
{{NULL},{f_catin_pr,f_chg_att_pd,NULL},                /* lista proc. #11.1 */
                                    NULL},             /* entrada 51 */
{{NULL},{catrp2_pdu,select,NULL},                      /* lista proc. #12.1 */
                                    NULL},             /* entrada 52 */
{{NULL},{f_opnin_pr,ac31_ra,f_open_pd,NULL},           /* lista proc. #13.1 */
                                      NULL},           /* entrada 53 */
```

```
{{NULL},{opnrp82_pdu,select,NULL},                    /* lista proc. #14.1 */
 {aci0_ra,NULL},                                      /* lista proc. #14.2 */
 {opnrp2_pdu,dxfridle,NULL},                          /* lista proc. #14.3 */
 {opnrp2_pdu,p_altrq_pr,p_altcf_pd,NULL},             /* lista proc. #14.4 */
                                          NULL},      /* entrada 54 */
{{NULL},{dxfridle,NULL},                              /* lista proc. #15.1 */
                 NULL},                               /* entrada 55 */
{{NULL},{f_cloin_pr,f_close_pd,NULL},                 /* lista proc. #17.1 */
                                   NULL},             /* entrada 56 */
{{NULL},{clorp2_pdu,select,NULL},                     /* lista proc. #18.1 */
                                  NULL},              /* entrada 57 */
{{NULL},{f_recin_pr,ac3i_ra,f_recover_pd,NULL},       /* lista proc. #19.1 */
                                          NULL},      /* entrada 58 */
{{NULL},{recrp82_pdu,init,NULL},                      /* lista proc. #20.1 */
 {aci0_ra,NULL},                                      /* lista proc. #20.2 */
 {recrp2_pdu,ac2_ra,dxfridle_rec,NULL},               /* lista proc. #20.3 */
 {recrp2_pdu,ac2_ra,p_altrq_pr,
  p_alcf_rec_pd,NULL},                                /* lista proc. #20.4 */
                       NULL},                         /* entrada 59 */
{{NULL},{dxfridle_rec,NULL},                          /* lista proc. #21.1 */
                       NULL},                         /* entrada 60 */
{{NULL},{f_locin_pr,f_locate_pd,NULL},                /* lista proc. #24.1 */
                                   NULL},             /* entrada 61 */
{{NULL},{locrp2_pdu,dxfridle,NULL},                   /* lista proc. #25.1 */
                                    NULL},            /* entrada 62 */
{{NULL},{f_erain_pr,f_erase_pd,NULL},                 /* lista proc. #26.1 */
                                   NULL},             /* entrada 63 */
{{NULL},{erarp2_pdu,dxfridle,NULL},                   /* lista proc. #27.1 */
                                    NULL},            /* entrada 64 */
{{NULL},{ac40_ra,ac43_ra,f_group_pd,NULL},            /* lista proc. #28.1 */
                                     NULL},           /* entrada 65 */
{{NULL},{ac41_ra,ac43_ra,f_group_pd,NULL},            /* lista proc. #29.1 */
                                     NULL},           /* entrada 66 */
{{NULL},{ac42_ra,ac43_ra,f_group_pd,NULL},            /* lista proc. #30.1 */
                                     NULL},           /* entrada 67 */
{{NULL},{ac38_ra,grouping,NULL},                      /* lista proc. #31.1 */
                         NULL},                       /* entrada #68 */
{{NULL},{ac38_ra,NULL},                               /* lista proc. #32.1 */
                NULL},                                /* entrada 69 */
{{NULL},{ac39_ra,NULL},                               /* lista proc. #33.1 */
 {init,NULL},                                         /* lista proc. #33.2 */
 {aci0_ra,NULL},                                      /* lista proc. #33.3 */
 {dxfridle,NULL},                                     /* lista proc. #33.4 */
 {p_altrq_pr,p_altcf_pd,NULL},                        /* lista proc. #33.5 */
                                  NULL},              /* entrada 70 */
```

```
/* Transferência de Massa de Dados - Entidade Iniciando */

((NULL),(ac44_md,rearq_pdu,ac2_md,NULL),                    /* lista proc. #1.1 */
 (ac24_md,ac48_md,rea_symin_pd,NULL),                       /* lista proc. #1.2 */
 (read,NULL),                                               /* lista proc. #1.3 */
            NULL),                                          /* entrada 71 */
((NULL),(ac44_md,wrtrq_pdu,ac2_md,NULL),                    /* lista proc. #2.1 */
 (p_symrq_pr,ac48_md,wrt_symcf_pd,NULL),                    /* lista proc. #2.2 */
 (write,NULL),                                              /* lista proc. #2.3 */
             NULL),                                         /* entrada 72 */
((NULL),(f_trecf_pr,dxfridle,NULL),                         /* lista proc. #3.1 */
                              NULL),                        /* entrada 73 */
((NULL),(pi8_datrq_pr,NULL),                                /* lista proc. #4.1 */
                                 NULL),                     /* entrada 74 */
((NULL),(daterq2_pdu,write_end,NULL),                       /* lista proc. #5.1 */
                                    NULL),                  /* entrada 75 */
((NULL),(read,NULL),                                        /* lista proc. #6.1 */
 (f_datin_pr,read,NULL),                                    /* lista proc. #6.2 */
                        NULL),                              /* entrada 76 */
((NULL),(f_daein_pr,read_end,NULL),                         /* lista proc. #8.1 */
                                 NULL),                     /* entrada 77 */
((NULL),(trerq2_pdu,r_xfr_end,NULL),                        /* lista proc. #9.1 */
                                    NULL),                  /* entrada 78 */
((NULL),(trerq2_pdu,w_xfr_end,NULL),                        /* lista proc. #10.1 */
                                    NULL),                  /* entrada 79 */
((NULL),(ac15_md,p_token_pd,NULL),                          /* lista proc. #11.1 */
 (f_trecf_pr,dxfridle,NULL),                                /* lista proc. #11.2 */
                           NULL),                           /* entrada 80 */
((NULL),(ac22_md,NULL),                                     /* lista proc. #12.1 */
 (canrq26_pdu,cancel_pd,NULL),                              /* lista proc. #12.2 */
                                    NULL),                  /* entrada 81 */
((NULL),(ac21_md,f_cancf_pr,dxfridle,NULL),                 /* lista proc. #13.1 */
                                    NULL),                  /* entrada 82 */
((NULL),(ac23_md,f_canin_pr,f_cancel_pd,NULL),              /* lista proc. #14.1 */
                                      NULL),                /* entrada 83 */
((NULL),(canrp26_pdu,dxfridle,NULL),                        /* lista proc. #16.1 */
                                   NULL),                   /* entrada 84 */
((NULL),(ac25_md,ac29_md,ac27_md,p_symrq_pr,
  write,NULL),                                              /* lista proc. #17.1 */
             NULL),                                         /* entrada 85 */
((NULL),(ac30_md,f_chkcf_pr,write,NULL),                    /* lista proc. #18.1 */
                                 NULL),                     /* entrada 86 */
((NULL),(ac29_md,f_chkin_pr,read,NULL),                     /* lista proc. #19.1 */
                                 NULL),                     /* entrada 87 */
((NULL),(ac30_md,p_symrp_pr,read,NULL),                     /* lista proc. #20.1 */
                                 NULL),                     /* entrada 88 */
((NULL),(ac22_md,ac50_md,resrq_pdu,rrestart_pd,NULL),       /* lista proc. #21.1 */
                                       NULL),               /* entrada 89 */
```

```
{{NULL},{ac50_md,resrq_pdu,wrestart_pd,NULL},          /* lista proc. #22.1 */
                                          NULL},       /* entrada 90 */
{{NULL},{ac21_md,f_rescf_pr,read,NULL},                /* lista proc. #23.1 */
                                  NULL},               /* entrada 91 */

{{NULL},{ac21_md,f_rescf_pr,write,NULL},               /* lista proc. #24.1 */
                                   NULL},              /* entrada 92 */
{{NULL},{ac23_md,f_resin_pr,f_rrestart_pd,NULL},       /* lista proc. #25.1 */
                                          NULL},       /* entrada 93 */
{{NULL},{ac23_md,f_resin_pr,f_wrestart_pd,NULL},       /* lista proc. #26.1 */
                                          NULL},       /* entrada 94 */
{{NULL},{ac50_md,resrp_pdu,read,NULL},                 /* lista proc. #27.1 */
                                   NULL},              /* entrada 95 */
{{NULL},{ac50_md,resrp_pdu,write,NULL},                /* lista proc. #28.1 */
                                    NULL},             /* entrada 96 */
{{NULL},{ac49_md,ac27_md,p_symrp_pr,read,NULL},        /* lista proc. #29.1 */
                                          NULL},       /* entrada 97 */
{{NULL},{ac47_md,ac22_md,res_symin_pd,NULL},           /* lista proc. #30.1 */
                                          NULL},       /* entrada 98 */
{{ac47_md,ac22_md,can_symin_pd,NULL},                  /* lista proc. #31.1 */
                                       NULL},          /* entrada 99 */
{{ac49_md,ac27_md,write,NULL},                         /* lista proc. #32.1 */
                                 NULL},                /* entrada 100 */
{{NULL},{ac47_md,ac22_md,res_symcf_pd,NULL},           /* lista proc. #33.1 */
                                          NULL},       /* entrada 101 */
{{NULL},{ac47_md,ac22_md,can_symcf_pd,NULL},           /* lista proc. #34.1 */
                                          NULL},       /* entrada 102 */
{{NULL},{rearq2_pdu,ac44_md,ac24_md,ac45_md,
  rea_symin_pd,NULL},                                  /* lista proc. #35.1 */
                       NULL},                          /* entrada 103 */
{{NULL},{wrtrq2_pdu,ac44_md,p_symrq_pr,
  ac45_md,wrt_symcf_pd,NULL},                          /* lista proc. #36.1 */
                              NULL},                   /* entrada 104 */
{{NULL},{ac30_md,NULL},                                /* lista proc. #37.1 */
                  NULL},                               /* entrada 105 */
{{NULL},{ac22_md,ac47_md,rest_can_pd,NULL},            /* lista proc. #38.1 */
                                          NULL},       /* entrada 106 */
{{NULL},{ac21_md,f_cancf_pr,canrp_pdu,ac26_md,
  dxfridle,NULL},                                      /* lista proc. #39.1 */
                  NULL},                               /* entrada 107 */
{{NULL},{canrq26_pdu,cancel_pd,NULL},                  /* lista proc. #40.1 */
                                       NULL},          /* entrada 108 */
```

```
/* Transferencia de Massa de Dados - Entidade Respondendo */

((NULL),(ac44_md,NULL),                                  /* lista proc. #1.1 */
 (ac15_md,ac48_md,p_token_pd,NULL),                      /* lista proc. #1.2 */
 (f_reain_pr,read,NULL),                                 /* lista proc. #1.3 */
                        NULL),                           /* entrada 109 */
((NULL),(ac44_md,f_wrtin_pr,NULL),                       /* lista proc. #2.1 */
 (ac48_md,wrt_symin_pd,NULL),                            /* lista proc. #2.2 */
 (write,NULL),                                           /* lista proc. #2.3 */
             NULL),                                      /* entrada 110 */
((NULL),(f_reain_pr,ac49_md,p_symrq_pd,
  rea_symcf_pd,NULL),                                    /* lista proc. #3.1 */
                    NULL),                               /* entrada 111 */
((NULL),(pi8_datrq_pr,NULL),                             /* lista proc. #4.1 */
                          NULL),                         /* entrada 112 */
((NULL),(daerq_pdu,ac2_md,read_end,NULL),                /* lista proc. #5.1 */
                                   NULL),                /* entrada 113 */
((NULL),(write,NULL),                                    /* lista proc. #6.1 */
 (f_datin_pr,write,NULL),                                /* lista proc. #6.2 */
                         NULL),                          /* entrada 114 */
((NULL),(f_daein_pr,write_end,NULL),                     /* lista proc. #8.1 */
                                NULL),                   /* entrada 115 */
((NULL),(f_trein_pr,f_r_xfer_end,NULL),                  /* lista proc. #9.1 */
                                  NULL),                 /* entrada 116 */
((NULL),(f_trein_pr,f_w_xfer_end,NULL),                  /* lista proc. #10.1 */
                                  NULL),                 /* entrada 117 */
((NULL),(trerp2_pdu,ac24_md,dxfridle,NULL),              /* lista proc. #11.1 */
 (trerp2_pdu,dxfridle,NULL),                             /* lista proc. #11.2 */
                                   NULL),                /* entrada 118 */
((NULL),(trerp2_pdu,dxfridle,NULL),                      /* lista proc. #11a.1 */
                                    NULL),               /* entrada 119 */
((NULL),(ac22_md,canrq_md,ac26_md,cancel_pd,NULL),       /* lista proc. #12.1 */
                                          NULL),         /* entrada 120 */
((NULL),(ac21_md,f_cancf_pr,dxfridle,NULL),              /* lista proc. #13.1 */
                                     NULL),              /* entrada 121 */
((NULL),(ac23_md,f_canin_pr,f_cancel_pd,NULL),           /* lista proc. #14.1 */
                                      NULL),             /* entrada 122 */
((NULL),(canrp26_pdu,dxfridle,NULL),                     /* lista proc. #15.1 */
                                    NULL),               /* entrada 123 */
((NULL),(ac25_md,ac29_md,ac27_md,p_symrq_pr,
  read,NULL),                                            /* lista proc. #16.1 */
            NULL),                                       /* entrada 124 */
((NULL),(ac29_md,f_chkin_pr,write,NULL),                 /* lista proc. #17.1 */
                                   NULL),                /* entrada 124 */
((NULL),(ac30_md,f_chkcf_pr,read,NULL),                  /* lista proc. #18.1 */
                                  NULL),                 /* entrada 125 */
((NULL),(ac30_md,p_symrp_pr,write,NULL),                 /* lista proc. #19.1 */
                                   NULL),                /* entrada 126 */
```

```
{{NULL},{ac22_md,ac50_md,resrq_pdu,rrestart_pd,NULL},   /* lista proc. #20.1 */
                                        NULL},          /* entrada 127 */
{{NULL},{ac22_md,ac50_md,resrq_pdu,wrestart_pd,NULL},   /* lista proc. #21.1 */
                                          NULL},        /* entrada 128 */
{{NULL},{ac21_md,f_rescf_pr,read,NULL},                 /* lista proc. #22.1 */
                                    NULL},              /* entrada 129 */
{{NULL},{ac21_md,f_rescf_pr,write,NULL},                /* lista proc. #23.1 */
                                     NULL},             /* entrada 130 */
{{NULL},{ac23_md,f_resin_pr,f_rrestart_pd,NULL},        /* lista proc. #24.1 */
                                          NULL},        /* entrada 131 */
{{NULL},{ac23_md,f_resin_pr,f_wrestart_pd,NULL},        /* lista proc. #25.1 */
                                            NULL},      /* entrada 132 */
{{NULL},{ac50_md,resrp_pdu,read,NULL},                  /* lista proc. #26.1 */
                                     NULL},             /* entrada 133 */
{{NULL},{ac50_md,resrp_pdu,write,NULL},                 /* lista proc. #27.1 */
                                    NULL},              /* entrada 134 */
{{NULL},{ac49_md,ac27_md,read,NULL},                    /* lista proc. #28.1 */
                                  NULL},                /* entrada 135 */
{{NULL},{ac47_md,res_symcf_pd,NULL},                    /* lista proc. #29.1 */
                                  NULL},                /* entrada 136 */
{{NULL},{ac47_md,can_symcf_pd,NULL},                    /* lista proc. #30.1 */
                                  NULL},                /* entrada 137 */
{{NULL},{ac49_md,ac27_md,write,NULL},                   /* lista proc. #31.1 */
                                   NULL},               /* entrada 138 */
{{NULL},{ac47_md,res_symin_pd,NULL},                    /* lista proc. #32.1 */
                                   NULL},               /* entrada 139 */
{{NULL},{ac47_md,can_symin_pd,NULL},                    /* lista proc. #33.1 */
                                  NULL},                /* entrada 140 */
{{NULL},{ac46_md,p_symrp_pr,ac22_md,resrq_pdu,
  wrestart_pd,NULL},                                    /* lista proc. #34.1 */
                       NULL},                           /* entrada 141 */
{{NULL},{ac46_md,p_symrp_pr,ac22_md,NULL},              /* lista proc. #35.1 */
 {canrq26_pdu,cancel_pd,NULL},                          /* lista proc. #35.2 */
                                      NULL},            /* entrada 142 */
{{NULL},{ac15_md,ac44_md,ac45_md,p_token_pd,NULL},      /* lista proc. #36.1 */
                                        NULL},          /* entrada 143 */
{{NULL},{f_wrtin_pr,ac44_md,ac45_md,wrt_symin_pd,NULL}, /* lista proc. #37.1 */
                                          NULL},        /* entrada 144 */
{{NULL},{ac30_md,read_end,NULL},                        /* lista proc. #38.1 */
                         NULL},                         /* entrada 145 */
{{NULL},{ac22_md,ac47_md,rest_can_pd,NULL},             /* lista proc. #39.1 */
                                        NULL},          /* entrada 146 */
{{NULL},{ac21_md,f_cancf_pr,canrp26_pdu,
  dxfridle,NULL},                                       /* lista proc. #40.1 */
                   NULL},                               /* entrada 147 */
{{NULL},{canrq26_pdu,cancel_pd,NULL},                   /* lista proc. #41.1 */
                                      NULL},            /* entrada 148 */
NULL
};
```

```
/*
 * Versão
 */

# define  VERS_1        0           /* Versão 1 */
# define  REVIS         1           /* Revisão */

/*
 * Nível de Serviço
 */

# define  CONF          1           /* Serviço Confiável */
# define  CORR_US       2           /* Serviço Corrigível pelo Usuário */

/*
 * Classe de Serviço
 */

# define  TRANS         0           /* Classe de Transferência */
# define  ACES          1           /* Classe de Acesso */
# define  GER           2           /* Classe de Gerência */
# define  TR_GR         3           /* Classe de Transf. e Gerenciamento */
# define  IRR           4           /* Classe Irrestrita */

/*
 * Unidades Funcionais
 */

# define  LEIT          2           /* Unidade de Leitura */
# define  ESCR          3           /* Unidade de Escrita */
# define  AC_AR         4           /* Unidade de Acesso a Arquivo */
# define  G_AR_LM       5           /* Unidade Gerência de Arquivo Limitada */
# define  G_AR_AM       6           /* Unidade Gerência de Arquivo Ampliada */
# define  AGR           7           /* Unidade de Agrupamento */
# define  REC           8           /* Unidade de Recuperação */
# define  REI           9           /* Unidade de Reinício de Transferência */

/*
 * Grupo de Atributos
 */

# define  ARM           0           /* Armazenamento */
# define  SEG           1           /* Segurança */
# define  PRIV          2           /* Privado */
```

```
/*
 * Disponibilidade de Rollback
 */

# define  S_RBCK        0                   /* Sem Rollback */
# define  RBCK          1                   /* Rollback disponível */

/*
 * Resultado de Estado
 */

# define  SUCS          0           /* Sucesso */
# define  FALH          1           /* Falha */

/*
 * Resultado de Ação
 */

# define  SUCC          0           /* Sucesso */
# define  E_TRS         1           /* Erro Transiente */
# define  E_PRM         2           /* Erro Permanente */

/*
 * DIAGNÓSTICO
 */

/*
 * Tipo de Diagnóstico
 */

# define  INF           0           /* Informativo */

  /* Erro transiente e erro permanente já está definido no parâmetro
     Resultado de Ação */

/*
 * Identificação do Erro
 */

# define  S_RZ          0           /* Sem razão */
# define  E_RP          1           /* Erro inespecífico do respondedor */
# define  S_PR          2           /* Sistema desligado */
# define  P_I_GF        3           /* Problema no gerenciamento do FTAM (não específico)*/
# define  GF_C          4           /* Gerenciamento do FTAM - Contabilidade ruim */
# define  GF_S          5           /* Gerenciamento do FTAM - Não passou pela segurança */
# define  ESP_E         6           /* Espera pode ser encontrada     */
# define  E_IN          7           /* Erro do Iniciador (Inespecífico)*/
```

```
# define  E_SBQ        8          /* Erro Subsequente */
# define  V_PR_CF     1000        /* Valores de parâmetros conflitando */
# define  V_PR_NS     1001        /* Valores de parâmetros não suportados */
# define  P_OB-DP     1002        /* Parâmetro obrigatório desposicionado */
# define  P_N_SUP     1003        /* Parâmetro não suportado */
# define  P-DUP       1004        /* Parâmetro duplicado */
# define  T_P_IL      1005        /* Tipo de parâmetro ilegal */
# define  T_P_NS      1006        /* Tipo de parâmetro não suportado */
# define  E_FTAM      1007        /* Erro do prot. FTAM - inespecífico */
# define  E_PCD       1008        /* Erro do prot. FTAM - erro de procedure */
# define  E_U_FUN     1009        /* Erro do prot. FTAM - erro de Unid. Func. */
# define  E_MSG       1010        /* Erro do prot. FTAM - erro de mensagem */
# define  F_N_INF     1011        /* Falha no nível inferior */
# define  E_E_NI      1012        /* Erro de endereçamento do nível inferior */
# define  TIME        1013        /* Timeout */
# define  S_SIS       1014        /* System shutdown */
# define  S_AG_IL     1015        /* Sequência de Agrupamento ilegal */
# define  V_GR        1016        /* Grouping threshold violation */
# define  A_U_NP      2000        /* Associação com usuário não permitida */
# define  N_SV_NS     2001        /* Nível de Serviço não suportado */
# define  C_SV_NS     2002        /* Classe de Serviço não suportada */
# define  U_F_NS      2003        /* Unidade Funcional não suportada */
# define  E_GR_AT     2004        /* Erro de grupo de atributo - inespecífico */
# define  G_AT_NS     2005        /* Grupo de Atributo não suportado */
# define  G_AT_NP     2006        /* Grupo de Atributo não permitido */
# define  CT_RM       2007        /* Conta ruim */
# define  E_G_AS      2008        /* Erro inespecífico no gerenciamento da associação */
# define  GA_ER       2009        /* Gerenciamento da Associação - endereço ruim */
# define  GA_CR       2010        /* Gerenciamento da Associação - conta ruim */
# define  E_JC-L      2011        /* Erro Janela de Checkpoint - too large */
# define  E_JC_P      2012        /* Erro Janela de Checkpoint - too small */
# define  E_JC_IN     2013        /* Erro Janela de Checkpoint - inespecífico */
# define  CO-NS       2014        /* Comunicação não suportada QoS     */
# define  ID_I_IN     2015        /* Identidade do Iniciador inaceitável */
# define  CO_G_R      2016        /* Gerência de Contexto refused      */
# define  RBC-ND      2017        /* Rollback não disponível */
# define  LTC_RP      2018
# define  LTC_SA      2019
# define  S_F_INV     2020        /* Senha do filestore inválida */
# define  NAQ_NE      3000        /* Nome do arquivo não encontrado */
# define  AT_S_NC     3001        /* Atributos de Seleção não casam */
# define  AT_I_NP     3002        /* Atributos iniciais não possíveis */
# define  N_AT_RM     3003        /* Nome de atributo ruim */
# define  AQ_NE       3004        /* Arquivo não existente */
# define  AQ_EX       3005        /* Arquivo já existe */
# define  AQ_N_CR     3006        /* Arquivo não pode ser criado */
# define  AQ_N_DL     3007        /* Arquivo não pode ser deletado */
```

```
# define  CC_N_DP     3008        /* Controle de concorrência não disponível */
# define  CC_N_SP     3009        /* Controle de concorrência não suportado */
# define  CC_N_PS     3010        /* Controle de concorrência não possível */
# define  BL_M_RT     3011        /* Bloqueio mais restritivo */
# define  AQ_OC       3012        /* Arquivo ocupado */
# define  AQ_N-DP     3013        /* Arquivo não disponível */
# define  CA_N_DP     3014        /* Controle de Acesso não disponível */
# define  CA_N_SP     3015        /* Controle de Acesso não suportado */
# define  CA_INC      3016        /* Controle de Acesso inconsistente */
# define  NARQ_T      3017        /* Nome de arquivo truncado */
# define  AT_IN_A     3018        /* Atributos iniciais alterados */
# define  CT_RUIM     3019        /* Conta ruim */
# define  SL_EX       3020        /* Superposição - seleciona arquivo existente */
# define  DC_AT_V     3021        /* Superposição - deleta e recria arquivo com
                                     atributos velhos */
# define  DC_AT_N     3022        /* Superposição - deleta e recria arquivo com
                                     atributos novos */
# define  SP_NP       3023        /* Superposição não possível */
# define  ES_AQ_A     3024        /* Especificação de arquivo ambígua */
# define  SN_CR_I     3025        /* Senha de criação inválida */
# define  SN_DL_I     3026        /* Senha de deleção inválida na superposição */
# define  VL_AT_R     3027        /* Valor de atributo ruim */
# define  AT_N_EX     4000        /* Atributo não existente */
# define  AT_N_LD     4001        /* Atributo não pode ser lido */
# define  AT_N_MD     4002        /* Atributo não pode ser mudado */
# define  AT_N_SP     4003        /* Atributo não suportado */
# define  NAT_RM      4004        /* Nome de atributo ruim */
# define  VAT_RM      4005        /* Valor de atributo ruim */
# define  FA_R_IN     5000        /* FADU ruim - inespecífico */
# define  FA_R_TM     5001        /* FADU ruim - erro de tamanho */
# define  FA_R_TI     5002        /* FADU ruim - erro de tipo */
# define  FA_R_       5003        /* FADU ruim   */
# define  FA_R_LC     5004        /* FADU ruim - localização ruim */
# define  FA_N_EX     5005        /* FADU não existe */
# define  FA_N_DP     5006        /* FADU não disponível - inespecífico */
# define  FA_N_LT     5007        /* FADU não disponível para leitura */
# define  FA_N_EC     5008        /* FADU não disponível para escrita */
# define  FA_N_LC     5009        /* FADU não disponível para localização */
# define  FA_N_AP     5010        /* FADU não disponível para apagamento */
# define  FA_N_IN     5011        /* FADU não pode ser inserida */
# define  FA_N_SB     5012        /* FADU não pode ser substituída */
# define  FA_N_LZ     5013        /* FADU não pode ser localizada */
# define  EL-D-RM     5014        /* Tipo elemento de dado ruim */
# define  OP_N_DP     5015        /* Operação não disponível */
# define  OP_N_SP     5016        /* Operação não suportada */
# define  OP_INC      5017        /* Operação inconsistente */
# define  CC_NDP      5018        /* Controle de Concorrencia não disponível */
```

```
# define  CC_NSP      5019            /* Controle de Concorrência não suportado */
# define  CC_INC      5020            /* Controle de Concorrência inconsistente */
# define  MP_N_DP     5021            /* Modo de Processamento não disponível */
# define  MP_N_SP     5022            /* Modo de Processamento não suportado */
# define  MP_INC      5023            /* Modo de Processamento inconsistente */
# define  CA_NDP      5024            /* Contexto de Acesso não disponível */
# define  CA_NSP      5025            /* Contexto de Acesso não suportado */
# define  ES_RM       5026            /* Escrita ruim - inespecífico */
# define  LT_RM       5027            /* Leitura ruim - inespecífico */
# define  FLH_LC      5028            /* Falha Local - inespecífica */
# define  ESP_AQ      5029            /* Falha Local - espaço de arquivo (?) */
# define  DD-COR      5030            /* Falha Local */
# define  FL_DIS      5031            /* Falha Local _ falha de dispositivo */
# define  TF-EXC      5032            /* Tamanho futuro de arquivo excedido */
# define  TF_INC      5033            /* Tamanho futuro de arquivo incrementado */
# define  UF_I_MP     5034            /* Unidade Funcional inválida no modo de processamento */
# define  TC_INC      5035            /* Tipo de conteúdo inconsistente */
# define  TC_SPL      5036            /* Tipo de conteúdo simplificado   */

/*
 * BLOQUEIOS (Controle de Concorrência)
 */

# define  COMP          0             /* Compartilhado */
# define  EXCL          1             /* Exclusivo */
# define  N_PD          2             /* Não Pedido */
# define  S_AC          3             /* Sem acesso */

/*
 * PEDIDO DE ACESSO / AÇÕES PERMITIDAS
 */

# define  LT            0             /* Acesso para leitura */
# define  INS           1             /* Acesso para inserção */
# define  SBT           2             /* Acesso para substituição */
# define  APG           3             /* Acesso para apagamento */
# define  ACR           4             /* Acesso para acréscimo */
# define  L_ATB         5             /* Acesso para leitura de atributo */
# define  M_ATB         6             /* Acesso para mudança de atributo */
# define  D_ARQ         7             /* Acesso para deleção de arquivo */
```

```
/*
 * GRUPOS DISPONÍVEIS DE IDENTIDADE DE UDAA
 */

# define  TRV           8           /* Transversal */
# define  TRV_RV        9           /* Transversal reverso */
# define  OR_AL        10           /* Ordem Aleatória */

/*
 * SUPERPOSIÇÃO
 */

# define  F_CR          0           /* Falha na criação */
# define  S_AQ          1           /* Selecionar arquivo já existente */
# define  AT_V          2           /* Deletar e criar com os atributos do
                                       arquivo já existente */
# define  AT_N          3           /* Deletar e criar com novos atributos */

/*
 * DISPONIBILIDADE DO ARQUIVO
 */

# define  DSP_I         0           /* Disponibilidade Imediata */
# define  DSP_A         1           /* Disponibilidade Adiada */

/*
 * NOMES DOS ATRIBUTOS PARA LEITURA
 */

# define  N_ARQ         0           /* Nome do arquivo */
# define  T_CTD         1           /* Tipo de conteúdo */
# define  CONT          2           /* Contabilidade de Armazenamento */
# define  DT_CR         3           /* Data e Hora de criação */
# define  DT_MD         4           /* Data e Hora da última modificação */
# define  DT_LT         5           /* Data e Hora do último acesso de leitura */
# define  DT_M_AT       6           /* Data e Hora da última modificação de atributo */
# define  ID_CR         7           /* Identidade do criador */
# define  ID_MD         8           /* Identidade do último modificador */
# define  ID_LT         9           /* Identidade do último leitor */
# define  ID_M_AT      10           /* Identidade do último modificador de atributo */
# define  DS_ARQ       11           /* Disponibilidade do arquivo */
# define  A_PMT        12           /* Ações Permitidas */
# define  T_ARQ        13           /* Tamanho do arquivo */
# define  T_F_ARQ      14           /* Tamanho futuro do arquivo */
# define  C_ACSS       15           /* Controle de Acesso */
# define  N_CIF        16           /* Nome de Cifragem */
# define  Q_LEG        17           /* Qualificações Legais */
# define  U_PVT        18           /* Uso Privativo */
```

```
/*
 * MODO DE PROCESSAMENTO
 */

# define  LOC           0           /* Localizar */
# define  LER           1           /* Ler */

/*
 * MODO DE RECUPERAÇÃO
 */

# define  NEN           0           /* Nenhum */
# define  I_ARQ         1           /* No início do arquivo */
# define  Q_P_CK        2           /* Qualquer ponto de checagem ativo */

/*
 * REFERÊNCIA A UDAA
 */

/* Primeiro ou último */

# define  PRIM          0           /* Primeiro nodo */
# define  ULT           1           /* Último nodo */

/* Relativo */

# define  ANT           0           /* Nodo anterior */
# define  CORR          1           /* Nodo corrente */
# define  PROX          2           /* Próximo nodo */

/* Inicial ou Final */

# define  IN            0           /* Nodo inicial */
# define  FN            1           /* Nodo final */

/*
 * CONTEXTO DE ACESSO
 */

# define  HIER          0           /* Hierárquico */
# define  N_HIER        1           /* Não hierárquico */
# define  PLN           2           /* Plano */
# define  N_PLN         3           /* Não Plano */
# define  PL_U          4
# define  DST           5           /* Desestruturado */
# define  DST_U         6
```

```
/*
 * OPERAÇÃO COM FADU NA ESCRITA
 */

# define  ISR           0           /* Inserir */
# define  SUB           1           /* Substituir */
# define  ACRS          2           /* Acrescentar */

/*
 * ESTADOS DO PROTOCOLO - listados na ordem apresentada na tabela
 *                        de estados - cap. 4 - protocolo FTAM
 */

# define  CLOSED        0           /* Fechado */
# define  INIT_PD       1           /* Inicialização Pendente - espera PDU initialize response */
# define  INIT          2           /* Inicializado */
# define  TERM_PD       3           /* Terminação Pendente - espera PDU terminate response */
# define  F_INIT_PD     4           /* Inicialização Pendente - espera prim. F-Initialize response */
# define  F_TERM_PD     5           /* Terminação Pendente - espera prim. F-Terminate response */
# define  SELECT_PD     6           /* Seleção Pendente - espera PDU Select response */
# define  SELECT        7           /* Selecionado */
# define  DESEL_PD      8           /* Desseleção Pendente - espera PDU Deselect response */
# define  CREATE_PD     9           /* Criação Pendente - espera PDU Create response */
# define  DELETE_PD     10          /* Deleção Pendente - espera PDU Delete response */
# define  READ_ATT_PD   11          /* Leitura de Atrib. Pendente - espera PDU Read-Attrib response */
# define  CHG_ATT_PD    12          /* Mudança de Atrib. Pendente - espera PDU Change-Attrib response */
# define  OPEN_PD       13          /* Abertura Pendente - espera PDU Open-Response */
# define  P_ALTIN_PD    14          /* Alteração do Contexto de apresentação pendente - espera prim.
                                       P-ALTER-CONTEXT indication */
# define  DXFRIDLE      15          /* Transferência de Dados Ociosa */
# define  REC_DXFRIDLE  16          /* Transferência de Dados Ociosa durante recuperação */
# define  CLOSE_PD      17          /* Fechamento Pendente - espera PDU Close-response */
# define  RECOVER_PD    18          /* Recuperação Pendente - espera PDU Recover-response */
# define  RC_ALTIN_PD   19          /* Alteração do Contexto de Apresentação Pendente -
                                       P-ALTER-CONTEXT indication durante recuperação */

# define  LOCATE_PD     20          /* Localizacao Pendente - Espera PDU Locate Response */
# define  ERASE_PD      21          /* Apagamento Pendente - espera PDU Erase-response */
# define  GROUPING      22          /* Agrupando PDU's */
# define  GROUP_PD      23          /* Agrupamento Pendente - espera PDU */
# define  F_SELECT_PD   24          /* Seleção Pendente - espera prim. F-Select response */
# define  F_DESEL_PD    25          /* Desselecao Pendente - Espera prim. F-Deselect response */
# define  F_CREATE_PD   26          /* Criação Pendente - espera prim. F-Create response */
# define  F_DELETE_PD   27          /* Deleção Pendente - espera prim. F-Delete response */
# define  F_READ_ATT_PD 28          /* Leitura de Atrib. Pendente - espera prim. F-Read-Attrib response */
# define  F_CHG_ATT_PD  29          /* Mudança de Atrib. Pendente - espera prim. F-Change-Attrib response */
# define  F_OPEN_PD     30          /* Abertura Pendente - espera prim. F-Open response */
```

```
# define  P_ALTCF_PD     31          /* Alteração do contexto de apresentação pendente - espera prim.
                                       P-ALTER-CONTEXT confirm */
# define  F_CLOSE_PD     32          /* Fechamento Pendente - espera prim. F-Close response */
# define  F_RECOVER_PD   33          /* Recuperação Pendente - espera prim. F-Recover-response */
# define  RC_ALTCF_PD    34          /* Alteração do Contexto de Apresentação Pendente - espera prim.
                                       P-ALTER-CONTEXT confirm durante a recuperação */
# define  F_LOCATE_PD    35          /* Localização Pendente - espera prim. F-Locate response */
# define  F_ERASE_PD     36          /* Apagamento Pendente - espera prim. F-Erase response */
# define  F_GROUP_PD     37          /* Agrupamento Pendente - espera prim. F-Begin-Group response */
# define  REA_SYMIN_PD   38          /* Espera prim. P-SYNC-MINOR indication após leitura */
# define  RES_SYMIN_PD   39          /* Espera prim. P-SYNC-MINOR indication - reinício solicitado */
# define  CAN_SYMIN_PD   40          /* Espera prim. P-SYNC-MINOR indication - cancelamento solicitado */
# define  READ           41          /* Leitura */
# define  READ_END       42          /* Terminando Leitura - espera PDU Transfer-End request */
# define  R_XFER_END     43          /* Terminando Transferência após leitura - espera PDU
                                       Transfer-End response */
# define  P_TOKEN_PD     44          /* Espera Token Sync-minor */
# define  WRT_SYMCF_PD   45          /* Espera prim. P-SYNC-MINOR confirm após escrita */
# define  RES_SYMCF_PD   46          /* Espera prim. P-SYNC-MINOR confirm - reinicio solicitado */
# define  CAN_SYMCF_PD   47          /* Espera prim. P-SYNC-MINOR confirm - cancelamento solicitado */
# define  WRITE          48          /* Escrita */
# define  WRITE_END      49          /* Terminando Escrita - espera PDU Transfer-End request */
# define  W_XFER_END     50          /* Terminando Transferência após escrita - espera PDU
                                       Transfer-End response */
# define  CANCEL_PD      51          /* Cancelamento Pendente - espera PDU Cancel response */
# define  F_CANCEL_PD    52          /* Cancelamento Pendente - espera prim. F-Cancel response */
# define  RRESTART_PD    53          /* Reinício de Leitura Pendente - espera PDU Restart response
                                       na operação de leitura */
# define  WRESTART_PD    54          /* Reinício de Escrita Pendente - espera PDU Restart-response
                                       na operação de escrita */
# define  F_RRESTART_PD  55          /* Reinicio de Leitura Pendente - espera prim. F-Restart-response
                                       na operação de leitura */
# define  F_WRESTART_PD  56          /* Reinício de Escrita Pendente - espera prim. F-Restart response
                                       na operação de escrita */
# define  REST_CAN_PD    57          /* Reinicio de cancelamento pendente */
# define  REA_SYNCF_PD   58          /* Espera prim. P-SYNC-MINOR confirm após leitura */
# define  F_R_XFR_END    59          /* Terminando Transferência após leitura - espera prim.
                                       F-Transfer-End response */
# define  WRT_SYMIN_PD   60          /* Espera prim. P-SYNC-MINOR indication após escrita */
# define  F_W_XFR_END    61          /* Terminando Transferência  após escrita - espera prim.
                                       F-Transfer-End response */
```

```
/*
 * DEFINIÇÃO DE EVENTOS
 */

/* Eventos do Gerenciamento do Regime FTAM */

# define  F_INIRQ       0           /* Primitiva F-Initialize request */
# define  ASSC_CF       1           /* Prim A-Assc-Cf carregando PDU Initialize response */
# define  F_TERRQ       2           /* Primitiva F-Terminate request */
# define  TERRP         3           /* PDU Terminate response */
# define  A_PABIN       4           /* Primitiva A-P-Abort indication */
# define  A_UABIN       5           /* Primitiva A-U-Abort indication */
# define  UABRQ         6           /* PDU U-Abort request */
# define  PABRQ         7           /* PDU P-Abort request */
# define  F_UABRQ       8           /* Primitiva F-U-Abort request */
# define  ASSC_IN       9           /* Prim A-Assc-Ind carregando PDU Initialize request */
# define  F_INIRP      10           /* Primitiva F-Initialize response */
# define  TERRQ        11           /* PDU Terminate request */
# define  F_TERRP      12           /* Primitiva F-Terminate response */


/* Eventos do Gerenciamento do regime de arquivo */

# define  F_SELRQ      13           /* Primitiva F-Select request */
# define  SELRP        14           /* PDU Select response */
# define  F_DESRQ      15           /* Primitiva F-Deselect request */
# define  DESRP        16           /* PDU Deselect response */
# define  F_CRERQ      17           /* Primitiva F-Create request */
# define  CRERP        18           /* PDU Create response */
# define  F_DELRQ      19           /* Primitiva F-Delete request */
# define  DELRP        20           /* PDU Delete response */
# define  F_RATRQ      21           /* Primitiva F-Read-Attrib request */
# define  RATRP        22           /* PDU Read Attrib response */
# define  F_CATRQ      23           /* Primitiva F-Change-Attrib request */
# define  CATRP        24           /* PDU Change-Attrib response */
# define  F_OPNRQ      25           /* Primitiva F-Open request */
# define  OPNRP        26           /* PDU Open response */
# define  P_ALTIN      27           /* Primitiva P-Alter-Context indication */
# define  F_CLORQ      28           /* Primitiva F-Close request */
# define  CLORP        29           /* PDU Close response */
# define  F_RECRQ      30           /* Primitiva F-Recover request */
# define  RECRP        31           /* PDU Recover response */
# define  F_LOCRQ      32           /* Primitiva F-Locate request */
# define  LOCRP        33           /* PDU Locate response */
# define  F_ERARQ      34           /* Primitiva F-Erase request */
# define  ERARP        35           /* PDU Erase response */
```

```
# define  F_BGPRQ       36          /* Primitiva F-Begin-Group request */
# define  F_EGPRQ       37          /* Primitiva F-End-Group request */
# define  GRPRP         38          /* Uma sequencia de PDU's response (PDU's grupo) */
# define  SELRQ         39          /* PDU Select-request */
# define  F_SELRP       40          /* Primitiva F-Select-response */
# define  DESRQ         41          /* PDU Deselect request */
# define  F_DESRP       42          /* Primitiva F-Deselect response */
# define  CRERQ         43          /* PDU Create request */
# define  F_CRERP       44          /* Primitiva F-Create response */
# define  DELRQ         45          /* PDU Delete request */
# define  F_DELRP       46          /* Primitiva F-Delete response */
# define  RATRQ         47          /* PDU Read-Attrib request */
# define  F_RATRP       48          /* Primitiva F-Read-Attrib response */
# define  CATRQ         49          /* PDU Change Attrib request */
# define  F_CATRP       50          /* Primitiva F-Change-Attrib response */
# define  OPNRQ         51          /* PDU Open request */
# define  F_OPNRP       52          /* Primitiva F-Open response */
# define  P_ALTCF       53          /* Primitiva P-Alter-Context confirm */
# define  CLORQ         54          /* PDU Close request */
# define  F_CLORP       55          /* Primitiva F-Close response */
# define  RECRQ         56          /* PDU Recover request */
# define  F_RECRP       57          /* Primitiva F-Recover response */
# define  LOCRQ         58          /* PDU Locate request */
# define  F_LOCRP       59          /* Primitiva F-Locate response */
# define  ERARQ         60          /* PDU Erase request */
# define  F_ERARP       61          /* Primitiva F-Erase response */
# define  F_BGPRP       62          /* Primitiva F-Begin-Group response */
# define  F_EGPRP       63          /* Primitiva F-End-Group response */
# define  GRPRQ         64          /* Sequencia de PDU's request (PDU's grupo) */

/* Eventos do regime de Transferência de Massa de Dados */

# define  F_REARQ       65          /* Primitiva F-Read request */
# define  F_WRTRQ       66          /* Primitiva F-Write request */
# define  P_TOKIN       67          /* Primitiva P-Token-Give indication */
# define  F_DATRQ       68          /* Primitiva F-Data request */
# define  F_DAERQ       69          /* Primitiva F-Data-End request */
# define  DATIN         70          /* Valor de dado no contexto do usuário */
# define  DAERQ         71          /* PDU Data-End request */
# define  F_TRERQ       72          /* Primitiva F-Transfer-End request */
# define  TRERP         73          /* PDU Transfer-End response */
# define  F_CANRQ       74          /* Primitiva F-Cancel request */
# define  CANRP         75          /* PDU Cancel response */
# define  CANRQ         76          /* PDU Cancel request */
# define  F_CANRP       77          /* Primitiva F-Cancel response */
# define  F_CHKRQ       78          /* Primitiva F-Check request */
# define  P_SYMCF       79          /* Primitiva P-Sync-Minor confirm */
```

```
# define  P_SYMIN       80              /* Primitiva P-Sync-Minor indication */
# define  F_CHKRP       81              /* Primitiva F-Check response */
# define  F_RESRQ       82              /* Primitiva F-Restart request */
# define  RESRP         83              /* PDU Restart response */
# define  RESRQ         84              /* PDU Restart request */
# define  F_RESRP       85              /* Primitiva F-Restart response */
# define  REARQ         86              /* PDU Read request */
# define  WRTRQ         87              /* PDU Write request */
# define  TRERQ         88              /* PDU Transfer-End-Request */
# define  F_TRERP       89              /* Primitiva F-Transfer-End response */

/* Eventos que são devolvidos aos usuários do FTAM - Gerencia do Regime FTAM :
 * (Eventos Saindo)
 */

# define  F_INICF       90              /* Primitiva F_initialize_confirm */
# define  F_TERCF       91              /* Primitiva F_Terminate_confirm */
# define  F_PABIN       92              /* Primitiva F_P_Abort_indication */
# define  F_UABIN       93              /* Primitiva F_U_Abort_indication */
# define  F_INIIN       94              /* Primitiva F_Initialize_indication */
# define  F_TERIN       95              /* Primitiva F_Terminate_indication */

/* Eventos que são devolvidos aos usuários do FTAM - Gerencia do Regime de
 * Arquivo - (Eventos Saindo)
 */

# define  F_SELCF       96              /* Primitiva F_Select_confirm */
# define  F_DESCF       97              /* Primitiva F_Deselect_confirm */
# define  F_CRECF       98              /* Primitiva F_Create_confirm */
# define  F_DELCF       99              /* Primitiva F_Delete_confirm */
# define  F_RATCF       100             /* Primitiva F_Read_Attrib_confirm */
# define  F_CATCF       101             /* Primitiva F_Change_Attrib_confirm */
# define  F_OPNCF       102             /* Primitiva F_Open_confirm */
# define  F_CLOCF       103             /* Primitiva F_Close_confirm */
# define  F_RECCF       104             /* Primitiva F_Recover_confirm */
# define  F_LOCCF       105             /* Primitiva F_Locate_confirm */
# define  F_ERACF       106             /* Primitiva F_Erase_confirm */
# define  F_SELIN       107             /* Primitiva F_Select_indication */
# define  F_DESIN       108             /* Primitiva F_Deselect_indication */
# define  F_CREIN       109             /* Primitiva F_Create_indication */
# define  F_DELIN       110             /* Primitiva F_Delete_indication */
# define  F_RATIN       111             /* Primitiva F_Read_Attrib_indication */
# define  F_CATIN       112             /* Primitiva F-Change_Attrib_indication */
# define  F_OPNIN       113             /* Primitiva F_Open_indication */
# define  F_CLOIN       114             /* Primitiva F_Close_indication */
# define  F_RECIN       115             /* Primitiva F_Recover_indication */
# define  F_LOCIN       116             /* Primitiva F_Locate_indication */
# define  F_ERAIN       117             /* Primitiva F_Erase-indication */
```

```
/* Eventos que são devolvidos aos usuários do FTAM - Transferência de Massa
 * de Dados (Eventos Saindo)
 */

# define  F_TRECF       118             /* Primitiva F_Transfer_End_confirm */
# define  F_DATIN       119             /* Primitiva F_Data_indication */
# define  F_DAEIN       120             /* Primitiva F_Data_End_indication */
# define  F_CANCF .     121             /* Primitiva F_Cancel_confirm */
# define  F_CANIN       122             /* Primitiva F_Cancel_indication */
# define  F_CHKCF       123             /* Primitiva F_Check_confirm */
# define  F_CHKIN       124             /* Primitiva F_Check_indication */
# define  F_RESCF       125             /* Primitiva F_Restart_confirm */
# define  F_RESIN       126             /* Primitiva F_Restart_indication */
# define  F_REAIN       127             /* Primitiva F_Read_indication */
# define  F_WRTIN       128             /* Primitiva F_Write_indication */
# define  F_TREIN       129             /* Primitiva F_Transf_End_indication */

/* Substitui os eventos INIRQ e INIRP por ASSC_IN e ASSC_CF mas vou precisar
 * deles para testar as pdus, então vou defini-los agora */

# define  INIRQ         130
# define  INIRP         131
```

```
/*
 * TIPOS DE DADOS
 */

 typedef int  EXTERNAL;
 typedef int  BITSTRING;
 typedef char BOOLEAN;
 typedef char G_STRING [85];
 typedef char O_STRING;

/*
 * PARÂMETROS
 */

#define  VERD          1
#define  FALSO         0
#define  NPROC         5
#define  APL          13
#define  TAM_MSG    1020              /* Tamanho das mensagens */
#define  F_E_SUP1    (key_t)1         /* Chave para Fila de Entrada Superior PGIS1 */
#define  F_S_SUP1    (key_t)2         /* Chave para Fila de Saída Superior PGIS1 */
#define  F_E_INF1    (key_t)3         /* Chave para Fila de Entrada Inferior PGIS1 */
#define  F_S_INF1    (key_t)4         /* Chave para Fila de Saída Inferior PGIS1 */
#define  F_E_SUP2    (key_t)5         /* Chave para Fila de Entrada Superior PGIS2 */
#define  F_S_SUP2    (key_t)6         /* Chave para Fila de Saída Superior PGIS2 */
#define  F_E_INF2    (key_t)7         /* Chave para Fila de Entrada Inferior PGIS2 */
#define  F_S_INF2    (key_t)8         /* Chave para Fila de Saída Inferior PGIS2 */

/* Estrutura da Mensagem */

struct  t_msg {
        long    tip_msg;              /* tipo da mensagem */
        char    txt_msg [TAM_MSG];    /* texto da mensagem */
};

/*
 * DEFINIÇÕES DOS EVENTOS DO CASE
 */

#define    ASSC_RQ      1
#define    ASSC_RP      3
#define    REL_RQ       5
#define    REL_IN       6
#define    REL_RP       7
#define    REL_CF       8
/* ASSC_IN e ASSC_CF estão definidos nos eventos do ftam em f_param.h */
```

[1] Kirner, Cláudio & Mendes, Suely B. T. (1988). **Sistemas Operacionais Distribuídos, Aspectos Gerais e Análise de sua Estrutura** , Campus, Capítulo 1.

[2] Lowe, H., (1983). "OSI Virtual Terminal Service", IEEE Network, v.71, nº 12, p. 1408 - 1413.

[3] Carvalho, Tereza C. M. B., (1988). "Serviços de Transferência de Arquivos em Redes Locais", **Dissertação de Mestrado** - Escola Politécnica da USP, Capítulo 3.

[4] ISO, (1984). "Information Processing Systems - Open Systems Interconnection - Basic Reference Model", ISO 7498.

[5] ISO, (1987). "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 1: General Introduction", **ISO/DIS 8571/1**.

[6] ISO, (1987). "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 2: The Virtual Filestore Definition", **ISO/DIS 8571/2**.

[7] ISO, (1987). "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 3: The File Service Definition", ISO/DIS 8571/3.

[8] ISO, (1987). "Information Processing Systems - Open Systems Interconnection - File Transfer, Access and Management - Part 4: The File Protocol Specification", ISO/DIS 8571/4.

[9] Coffin, Stephen, (1988). UNIX The Complet Reference, McGraw-Hill.

[10] Sampaio, Marcus C. et. al., (1988). UNIX - Guia do Usuário, McGraw-Hill, Capítulo 2.

[11] Kernighan, Brian W. & Mashey, John R., (1981). "The Unix Programming Enviroment", Computer, p. 12-22.

[12] Kernighan, Brian W. & Ritchie, Dennis M., (1986). "C" A Linguagem de Programação, Campus.

[13] Moura, J. Antão et. al., (1986). Redes Locais de Computadores - Protocolos de Alto Nível e Avaliação de Desempenho, McGraw-Hill, Capítulo 6, p.187-199.

[14] ISO, (1985). "Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)", ISO/DIS 8824.

[15] ISO, (1985). "Information Processing Systems - Open Systems Interconnection - Specification of Basic Rules or (?) Abstract Syntax Notation One (ASN.1)" , ISO/DIS 8825.

[16] ISO, (1986). "Information Processing Systems - Open Systems Interconnection - Service Definition for Common Application Service Elements. Part 2: Association Control", ISO/DIS 8649/2.

[17] ISO, (1986). "Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition, ISO/DIS 8822..

[18] Thomas, Rebecca et. al., (1986). Advanced Programmer's Guide to UNIX System V, McGraw-Hill, Capítulo 7, p. 261-350..

[19] Joberto, S. B. M. & Lima, Clizenit P. A., (1990). "Protocolo de Transferência de Arquivos em Redes Industriais: Considerações sobre Implementação e Desempenho", Artigo submetido ao 8º Congresso Brasileiro de Automática - 10 a 14 de setembro de 1990, Belém/Pa.

[20] Gomaa, Hassan, (1989). "A Software Design Method for Distributed Real-Time Applications", The Journal of Systems and Software, Vol.9, p. 81-94.

[21] Rockkind, Marc J., (1985). Advance Unix Programming, Prentice-Hall.

[22] Lima, Clizenit P. A. & Joberto, S. B. M., (1990). "Estruturação e Implementação do Protocolo FTAM", 8º Simpósio Brasileiro de Redes de Computadores, Campinas/SP.

[23] Halsall Fred, (1988). Data Communications, Computer Networks and OSI, Addison - Wesley , 2nd Edition, Capítulo 9, p. 402-427.

[24] Liu, Ming T., (1989). "Protocol Engineering", Advances in Computer, Vol. 29, p.79 - 195.

[25] EDISA, (1989). Manual de Referência - EDIX 5, Volumes I e II.