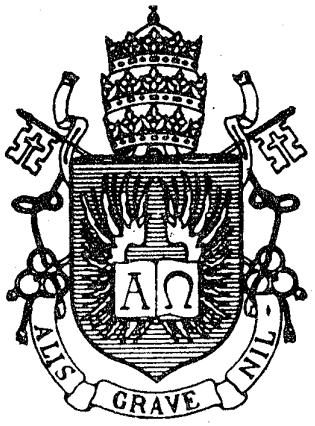

PUC

Série: Monografias em Ciência da Computação, Nº 04/84

MANUAL DE IMPLEMENTAÇÃO DO PROTOCOLO DE TRANSFERÊNCIA DE ARQUIVOS DA REDPUC

por

Orivaldo de Lira Tavares

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453

RIO DE JANEIRO – BRASIL

PUC/RJ - DEPARTAMENTO DE INFORMÁTICA

Série: Monografias em Ciência da Computação, Nº 04/84
Editor: Antônio L. Furtado Outubro, 1984

MANUAL DE IMPLEMENTAÇÃO DO PROTOCOLO DE TRANSFERÊNCIA DE ARQUIVOS DA REDPUC*

por

Orivaldo de Lira Tavares

* Trabalho parcialmente financiado pela FINEP.

AGRADECIMENTOS

Ao Prof. Daniel Schwabe que orientou o presente trabalho.

Aos demais professores do Depto. de Informática da PUC/RJ pelos conhecimentos e experiências que me transmitiram.

Aos colegas da pós-graduação do Depto. de Informática da PUC/RJ pelo apoio constante e amizade sincera.

Aos colegas do Depto. de Informática e Estatística e do Serviço de Estatística e Computação da Universidade Federal do Pará pelo incentivo que acelerou a conclusão deste trabalho.

Ao CAPES/PICD e à UFPa que apoiaram financeiramente a realização do meu curso de pós-graduação.

RESUMO

Este relatório apresenta o código PASCAL do PROTOCOLO DE TRANSFERENCIA DE ARQUIVOS DA REDPUC (PTA - REDPUC) e complementa as informações contidas na referência [1] para facilitar a implementação do PTA-REDPUC.

Palavras-chaves: Protocolo de transferência de arquivos, protocolo de alto nível, REDPUC.

ABSTRACT

This report presents the PASCAL code of the REDPUC FILE TRANSFER PROTOCOL (PTA - REDPUC) and gives complementary informations to reference [1] to make easier the PTA-REDPUC implementation.

Keywords: File transfer protocol, high level protocol, REDPUC.

CONTEÚDO PAGINA

1. REQUISITOS PARA A INSTALAÇÃO DO PTA.

Os requisitos para a instação do PTA da REDPUC são:

- Processador INTEL 8086 com 256 Kbytes de memória principal.
- Dois disquetes de face e densidade simples com o seguinte conteúdo:

  Primeiro disquete:

  - CONSTANT.PTA (Constantes do PTA).
  - TIPOS.PTA (Tipos globais do PTA).
  - PTA.PAS (Módulos inicializadores do PTA).
  - ABRETRA.PAS (Módulo que abre o contexto de transferência).
  - CRIAARQ.PAS (Módulo que cria o arquivo virtual).
  - ENCERTRA.PAS (Módulo que encerra o contexto de transferência).
  - MSGENVIA.PAS (Módulo emissor de mensagens do nível de aplicação).
  - SESSAO.PAS (Principais módulos do nível de sessão).
  - PROCMARC.PAS (Módulos de processamento das marcas de reinício).

  Segundo disquete:

  - COPIA.PAS (Módulo que faz a cópia dos arquivos).
  - NUCLEO60.CMD (Núcleo de comunicação da REDPUC).
  - CHAMANUC.R86 (Módulo de chamada do núcleo de comunicação da REDPUC).

- Dois disquetes com o PASCAL MT+86:

  O primeiro deve conter:

  - PASCAL.CMD (Compilador Pascal MT+86).
  - MT+86.000 (Programa raiz).
  - MT+86.001 \
  - MT+86.002 > (Overlays para o compilador)
  - MT+86.003 /
  - MT+86.004 /

  O segundo deve conter:

  - LINKMT.CMD (linkeditor Pascal MT+86).
  - LINKMT.001
  - LINKMT.002
  - FULLHEAP.R86 (Módulo que trata de alocação dinâmica de memória).
  - PASLIB.R86 (Rotinas da biblioteca de tempo de execução do Pascal MT+86).

2. AJUSTE DOS ATRIBUTOS DO PTA.

O PTA possue duas estruturas onde são armazenados os atributos do arquivo virtual e da operação de transferência. Além disso, possue dez tabelas que são usadas para a crítica dos atributos do arquivo virtual e da operação de transferência.

Nas duas seções seguintes descreve-se como possíveis alterações e/ou ajustes desses atributos podem ser realizados.

2.1 ATRIBUTOS DO ARQUIVO VIRTUAL.

A estrutura do PTA que armazena os atributos do arquivo virtual é denominada ATRIB_ARQUIVO. Assim, qualquer alteração

e/ou ajuste nos atributos do arquivo virtual exige um reexame desta estrutura.

Existem ainda seis tabelas usadas para crítica dos atributos do arquivo virtual. Nessas tabelas são inicializados os diversos atributos do arquivo virtual admissíveis no nodo onde o PTA está sendo implementado. Assim, elas são objetos de revisão toda vez que o PTA for implantado em um nodo. As tabelas são as seguintes:

- TAB_ESTRUTURA (Tabela de tipos de estruturas de arquivos).
- TAB_UNI (Tabela com as formas de unidades dos arquivos).
- TAB_ORDU (Tabela de tipos de ordenações das unidades dos arquivos).
- TAB_ACES (Tabela com as formas de acesso às unidades dos arquivos).
- TAB_COD_D (Tabela com os tipos de códigos de dados dos arquivos).
- TAB_CONT (Tabela com os tipos de códigos de controle que podem ser inseridos nos arquivos).

A inicialização dessas tabelas é feita na procedure INIC_PG_TRANSFS.

Essas tabelas são usadas para crítica na procedure CRITICA_ARQUIVO_ATRIBUTOS.

2.2 ATRIBUTOS DA OPERAÇÃO.

A estrutura do PTA que armazena os atributos da operação é denominada ATRIB_TRANSFERENCIA. Qualquer alteração e/ou ajuste nos atributos da operação exige um reexame dessa estrutura.

Além dessa, existem quatro tabelas usadas para a crítica dos

atributos da operação. Essas tabelas comportam os diversos atributos da operação admissíveis no nodo onde o PTA está implementado. Elas são objeto de revisão toda vez que o PTA for implantado em um nodo. As tabelas são as seguintes:

- TAB_OPERACAO (Tabela com os modos de operação do PTA quanto ao arquivo destino).
- TAB_UNIDADE (Tabela com as unidades de transferência admissíveis).
- TAB_MODO_TRANSF (Tabela com os modos de transferência possíveis).
- TAB_FUNCAO (Tabela com as funções que o PTA pode ter).

A inicialização dessas tabelas é feita na procedure INIC_PG_TRANSFS.

Essas tabelas são usadas para crítica na procedure CRIT_ATRIB_OPERACAO.

3. INCLUSAO DE NOVAS ROTINAS DE ACESSO A ARQUIVO.

Novas rotinas de acesso a arquivo podem ser incluídas no módulo COPIA.

A chamada dessas novas rotinas devem ser incluídas nas procedures COPIA_SECUNDARIO e CONF_POSITIVA_OPERACAO, nos pontos apropriadamente indicados através de comentários.

A rotina de leitura, no lado emissor do arquivo, deve ser ativada por um processo semelhante aos processos EMITE1 e EMITE2. Nesse caso, o que deve ser incluído nas procedures COPIA_SECUNDARIO e CONF_POSITIVA_OPERACAO é a atualização do nome do processo na página de transferências (PAG_TRANSFS) e a criação

dele, do mesmo modo que são feitas para os processo EMITE1 e EMITE2.

4. INTERRELAÇAO ENTRE AS ROTINAS DO PTA.

O corpo do programa principal do PTA é constituído por uma chamada da procedure INICIALIZADOR e pela criação dos processos RECEPCAO_MSG, CONTROLE_COMUNICACAO e TESTADOR.

A procedure INICIALIZADOR aloca as áreas (buffers) de trabalho do PTA e chama as procedures INIC_PG_SESSOES e INIC_PG_TRANSFS.

A procedure INIC_PG_SESSOES inicializa a página de sessões (PAG_SESSOES) e os "arrays" de últimas marcas pendentes e últimas marcas confirmadas, respectivamente ULTIMAS_MARCAS e MARCAS_CONFIRMADAS.

A procedure INIC_PG_TRANSFS inicializa a página de transferências (PAG_TRANSFS) e as tabelas de atributos da operação e do arquivo virtual, conforme visto na seção 2.

O processo TESTADOR simula um processo usuário interagindo com o PTA para a transferência de arquivos, através de envio e recepção de mensagens para/do PTA, respectivamente.

O processo CONTROLE_COMUNICACAO recebe e distribui todas as mensagens destinadas ao nível de aplicação do PTA.

O processo RECEPCAO_MSG recebe e distribui todas as mensagens destinadas ao nível de sessão do PTA.

4.1 NIVEL DE APLICAÇAO.

Esse nível está definido em detalhes em [1].

O processo CONTROLE_COMUNICACAO distribui as mensagens

recebidas para uma das seguintes procedures:

- ABRE_TRANSFERENCIA (Abre o contexto de transferência).
- CRIA_ARQUIVO (Cria o arquivo virtual).
- COPIA (Transfere o arquivo para o destino).
- ABORTA_TRANSFERENCIA (Encerra prematuramente o contexto de transferência).
- ABORTA_OPERACAO (Encerra prematuramente a operação de transferência atual).
- ENCERRA_TRANSFERENCIA (Encerra o contexto de transferência).
- RECUPERA_ATRIBUTOS (Recupera e envia para o usuário todos os atributos atuais em um contexto de transferência).

4.2 NIVEL DE SESSAO.

Esse nível está definido em detalhes em [1].

O processo RECEPCAO_MSG distribui as mensagens recebidas para uma das seguintes procedures:

- ABRE_SESSAO (Abre uma sessão de comunicação).
- PROCESSA_MARCA (Envia marcas de reinício).
- REINICIA (Reinicia a sessão de comunicação e avisa ao nível de aplicação do PTA).
- ABORTA_SESSAO (Encerra prematuramente a sessão de comunicação).
- ENCER_SESSAO (Encerra a sessão de comunicação).
- PREPARA_MSG (Prepara e envia as mensagens de/para o nível de aplicação do PTA).

5. LISTAGEM DOS MODULOS DO PTA.

```
(*CONSTANTE DO NIVEL DE SESSAO *)

   NUM_SESSOES = 2;

{ CONSTANTES DO NUCLEO DE COMUNICACAO }

   PORTA_ST    = 4; { Porta na qual o processo sessao recebe msgs do n.transp.}
   PORTA_SES   = 5; { Porta na qual o processo sessao recebe msgs do n.aplic.}
   PORTA_APL   = 7; { Porta na qual o processo aplicacao recebe mensagens }
   PORTA_USU   = 8; { Porta na qual o processo usuario recebe mensagens }
   PORTA_EMI   = 9; { Porta na qual o processo emissor recebe mensagens }

   TAM_PILHA   = 12288;  { Tamanho da pilha de registros de ativacao }
   PRIORIDADE  = 10000; { Prioridade estatica dos processos }
   TAMANHO     = 2048;  { Espaco da pilha reservado para o registro de ativacao
                          de cada processo }

     ERROR                  = -1;
     ERRO_CRITICA           = -2;
     EXITO                  =  0;

     CHARZERO               = '0';
     ESPACO                 = ' ';
     INICIAL                = -1;
     VAZIA                  = -1;
     PROC_NULO              = '       ';
     XIVBRANCOS             = '              ';
     SETEBRANCOS            = '       ';

     MATRBARQ               = 10;
     MATRBOPR               = 13;
     NUM_MAX_TRANSFERENCIAS = 2;
     CONTA_TAM              = 15;
     TAM_NOME_ARQ           = 15;
     TAM_PASSWORD           = 8;
     RETARDO_MEDIO          = 3;
     TOLERANCIA             = 1;
     MAIOR_MENSAGEM         = 200;
     TAM_ARQUIVO_MAX        = 10;
     TAM_UNIDADE_MAX        = 126;
     TAM_BLOCO_MAX          = 126;
     TAM_JANELA_MAX         = 10;
     ID_APLICACAO           = 7;


     { TABELA DE ERROS }
     MAQNATRANSF            = 200;
     SEC_MACK_TRANSF        = 210;
     ERRO_ALOCACAO_MEMORIA  = 220;

     ERRO_ESTRUTURA         = 300;
     ERRO_TAM_ARQUIVO       = 310;
     ERRO_FORMA_UNIDADES    = 320;
     ERRO_UNIDADE_TAM_MAX   = 330;
```

```
    ERRO_ORDEM_UNIDADES    = 340;
    ERRO_POSICAO_CHAVE     = 350;
    ERR_FORMA_ACESSO       = 360;
    ERRO_CODIGO_DADOS      = 370;
    ERRO_CONTROL_CODE      = 380;
    ABERTURA_ARQUIVO       = 390;

    ERRO_OPERACAO          = 550;
    ERRO_FUNCAO_SECUNDARIO = 560;
    ERRO_BYTE_TRANSF       = 570;
    ERRO_TRANSF_UNIDADE    = 580;
    ERR_TAM_UNIDADE        = 590;
    ERRO_MODO_TRANSF       = 600;
    ERRO_JANELA            = 620;
    ERRO_INTEVALO_MARCAS   = 630;

    MOTIVO_EXTERNO         = 750;

    NAOABRESESSAO          = 1005;
    JANELAESGOTADA         = 1040;

{ TAMANHOS DAS TABELAS DE ATRIBUTOS DA TRANSFERENCIA E ATRIBUTOS DO ARQUIVO
  VIRTUAL }
  NELEOP                   = 2;
  NELEUNI                  = 4;
  NELETRANS                = 3;
  NELFUNC                  = 2;
  NELESTRU                 = 3;
  NELUNI                   = 2;
  NELORDU                  = 2;
  NELACES                  = 3;
  NELCOD                   = 3;
  NELCONT                  = 5;
```

```
(*  TIPOS.PTA *)
(*-------------------------------------------------------------*)

   { Tipos do nucleo de comunicacao }
   str7          = string[7];
   strmsg        = record
                     tam : byte;
                     pta : ^mensagem
                   end;

   dsc_processo  = packed array [0..0] of char;
   ptr_dsc_prc   = ^dsc_processo;
   processo      = packed array [0..0] of char;
   ptr_proc      = ^processo;
   pilha         = packed array [0..TAM_PILHA] of char;
   ptr_pilha     = ^pilha;
   ptr_none      = ^str7;

   func_nucleo   = (DEVOLVE, ENVIA, ENVIA_B, RECEBE, RECEBE_B, APAGA_MSG,
                    LIGA_INT, DESL_INT, LIGA_TEMPO, DESLIGA_TEMPO,
                    ENTRADA_SAIDA, INICIA_PROCESSO, ACABA_PROCESSO,
                    BUSCA_PROCESSO, INIC2_PROCESSO);
   bloco_nucleo  = record
                    retcode         : integer;
                    case funcao     : func_nucleo of
                      DEVOLVE       : ();
                      ENVIA_B,
                      RECEBE,
                      RECEBE_B,
                      BUSCA_PROCESSO:
                                    (msg_ptr     : ^strmsg;
                                     proc        : ptr_dsc_prc;
                                     porta       : byte);
                      INICIA_PROCESSO:
                                    (nom_proc    : ptr_none;
                                     ender_proc  : ptr_dsc_prc;
                                     pilha_proc  : ptr_pilha;
                                     prioridade  : integer);
                      ACABA_PROCESSO : ()
                   end;
   ptr_bloco_nuc         = ^bloco_nucleo;

{_____________________________________________________________________}

     tipo_cod_erro       = array [1..2] of integer;
     desc_processo       = str7;
     aptmsg              = ^mensagem;       (*teste*)

     ptr_atributos_transf = ^atrib_transferencia;
     ptr_atr_arquivo     = ^atrib_arquivo;

     estados             = (LIVRE, INICIANDO, INICIADO, EM_ANDAMENTO,
                            FINALIZANDO);

     tipo_conta          = string [CONTA_TAM];
```

```
tipo_password_conta  = string [TAM_PASSWORD];
tipo_arquivo_nome    = string [TAM_NOME_ARQ];
tipo_file_password   = string [TAM_PASSWORD];

tipo_bloco_dados     = string [TAM_BLOCO_MAX];

ccondcriarq          = array [1..NATRBARQ] of integer;
ccondopr             = array [1..NATRBOPR] of integer;


{ DEFINICAO DAS ESTRUTURAS DE DADOS }

{ Tipos de mensagens trocadas entre os PTAs_APLICACAO }

{ 010:Mensagem de abertura da transferencia }
{ 020:Confirmacao positiva da abertura da transferencia }
{ 021:Confirmacao negativa da abertura da transferencia }
{ 030:Pedido de criacao de uma instancia do arquivo virtual}
{ 040:Confirmacao positiva da criacao da instancia do arquivo virtual }
{ 041:Confirmacao condicional da criacao da instancia do arq. virtual}
{ 042:Confirmacao negativa da criacao da instancia do arquivo virtual }
{ 050:Mensagem com bloco(s) de dados do arquivo virtual }
{ 060:Mensagem de pedido de operacao}
{ 070:Mensagem de confirmacao positiva do pedido de operacao }
{ 071:Mensagem de confirmacao condicional do pedido de operacao }
{ 072:Mensagem de recusa do pedido de operacao }
{ 080:Mensagem de aborto do contexto de transferencia}
{ 081:Mensagem de aborto da operacao de transferencia }
{ 090:Mensagem de fim de dados do arquivo }
{ 100:Confirmacao do recebimento da mensagem de fim de dados }
{ 110:Mensagem de encerramento do contexto de transferencia }


{ TIPOS DE MENSAGENS NA INTERFACE PTA_APLICACAO / PTA_SESSAO }

{ 200:Mensagem pedindo a abertura de uma sessao de comunicacao }
{ Resposta do protocolo de sessao a um pedido de abertura de uma sessao
  de comunicacao}
{ 210:Confirmacao positiva }
{ 211:Confirmacao condicional }
{ 212:Confirmacao negativa (rejeicao do pedido) }
(*220:Mensagem com o tamanho da janela de marcas pendentes. *)
{ 240:Pedido para o protocolo de sessao enviar uma marca de reinicio }
{ 250:Pedido para o protocolo de sessao reiniciar a transferencia }
{ 260:Mensagem com a identificacao de reinicio p/ o protocolo de
      aplicacao }
{ 270:Pedido para abortar a sessao de comunicacao }
{ 280:Pedido para encerrar a sessao de comunicacao }.

{ TIPOS DE MENSAGENS NA INTERFACE USUARIO/PTA }

{ 400:Pedido de abertura de um contexto de transferencia }
{ 410:Resposta positiva ao pedido de abertura do contexto de
      transferencia }
{ 411:Resposta negativa ao pedido de abertura do contexto de
```

```
     transferencia }
{ 420:Pedido para a criacao de um arquivo virtual }
{ 430:Confirmacao positiva da criacao do arquivo virtual }
{ 431:Confirmacao condicional da criacao de um arquivo virtual }
{ 432:Recusa da criacao do arquivo virtual }
{ 440:Pedido de operacao dentro de um contexto de transferencia }
{ 450:Mensagem de confirmacao de aceitacao do pedido de operacao }
{ 451:Mensagem de confirmacao condicional do pedido de operacao }
{ 452:Recusa do pedido de operacao }
{ 455:Aviso de encerramento da operacao atual de transferencia }
{ 460:Pedido de aborto do contexto de transferencia }
{ 470:Pedido de aborto da operacao atual do contexto de transferencia }
{ 480:Confirmacao do aborto da operacao }
{ 490:Mensagem de encerramento do contexto de transferencia }
{ 491:Confirmacao do encerramento da transferencia }
{ 500:Solicitacao dos atributos da transferencia. }
{ 510:Mensagem com os atributos atuais da transferencia. }
{ 520:Solicitacao dos atributos do arquivo virtual. }
{ 530:Mensagem com os atributos do arquivo virtual. }
{ 540:Solicitacao dos atributos da operacao atual. }
{ 550:Mensagem com os atributos da operacao atual. }
{ 555:Estrutura dos atributos solicitados nao existe ou o estado da
     transferencia é improprio. }


reg010 = record
           protocolo              : integer;
           local_sessao           : integer;
           id_trsf_origem         : integer;
           sec_conta              : tipo_conta;
           sec_pass_conta         : tipo_password_conta
         end;

reg060 = record
           { atributos da operacao }
           op_conta                : tipo_conta;
           op_pass_conta           : tipo_password_conta;
           op_arquivo              : tipo_arquivo_nome;
           op_arq_pass             : tipo_file_password;
           op_operacao_modo        : char;
           op_funcao_secundario    : char;
           op_tam_byte_transf      : integer;
           op_unidade              : char;
           op_tam_unidade          : integer;
           op_modo_transf          : char;
           op_reinicio             : boolean;
           op_tam_janela_marcas    : integer;
           op_intervalo_entre_marcas: integer
         end;

reg200 = record
           ses_eld_remoto          : integer;
           ses_max_tam_msg         : integer;
           ses_tempo_espera        : integer
         end;
```

```
    reg400 = record
               u_processo              : str7;
               p_conta                 : tipo_conta;
               p_pass_conta            : tipo_password_conta;
               s_conta                 : tipo_conta;
               s_pass_conta            : tipo_password_conta;
               u_etd_remoto            : integer;
               u_tam_bloco_dados       : integer
             end;

    reg420 = record
               { atributos do arquivo virtual }
               u_estrutura             : char;
               u_tam_max               : integer;
               u_unidades              : char;
               u_tam_unidade           : integer;
               u_ordem_unidades        : char;
               u_forma_acesso          : char;
               u_posicao_chave         : integer;
               u_tamanho_chave         : integer;
               u_codigo_dados          : char;
               u_control_code          : char
             end;

    reg440 = record
               user_process            : str7;
               { atributos da operacao }
               u_p_conta               : tipo_conta;
               u_p_pass_conta          : tipo_password_conta;
               u_p_arquivo             : tipo_arquivo_nome;
               u_arqp_pass             : tipo_file_password;
               u_s_conta               : tipo_conta;
               u_s_pass_conta          : tipo_password_conta;
               u_s_arquivo             : tipo_arquivo_nome;
               u_arqs_pass             : tipo_file_password;
               u_modo_operacao         : char;
               u_funcao_primario       : char;
               u_tam_byte_transf       : integer;
               u_unidade               : char;
               u_tam_unidade           : integer;
               u_modo_transf           : char;
               u_reinicio              : boolean;
               u_tam_janela_marcas     : integer;
               u_intervalo_entre_marcas: integer
             end;

{   reg510 = record
               trsf_processo           : str7;
               trsf_conta              : tipo_conta;
               trsf_pass_conta         : tipo_password_conta;
               trsf_estado             : estados
             end;

    reg550 = record
```

```
                trsf_funcao_primario    : char;
                trsf_arq_local          : tipo_arquivo_nome;
                trsf_chav_arq           : tipo_file_password;
                trsf_modo_operacao      : char;
                trsf_tam_byte           : integer;
                trsf_unidade            : char;
                trsf_comp_unidade       : integer;
                modo_trsf               : char;
                trsf_reinicio           : boolean;
                trsf_interv_entre_marcas: integer
              end;
}

  mensagem = record
               case boolean of
                TRUE : (grande: string [MAIOR_MENSAGEM]);
                FALSE:
                (
                 tipo_msg     : integer;
                 case tipo0   : integer of

(*-------------------------------------------------------------------------*)

                 (* MENSAGENS TROCADAS ENTRE OS PTAs_SESSAO *)

                 710, 740, 750, 760, 761, 770,
                 (*780, *)790:
                    (protocolo      : integer;
                     id_sessao      : integer;
                     case tipo10    : integer of
                        710:
                           (porta_entrada    : integer;
                            eld_origem       : integer;
                            tam_mensag_max   : integer;
                            temp_espera      : integer);

                        740, 750, 760, 761, 770:
                           (marca_reinicio   : integer);
{*
                        780:
                           (motivo           : integer);
*}
                        790:
                           ()
                    );

                 720, (*721, *)722:
                    (ids_protocolo          : integer;
                     sess_origem            : integer;
                     sess_destino           : integer;
                     case tipo20            : integer of
                        720, 722:
                           ()      (*;
                        721:
                           (tam_msg_max      : integer;
```

```
                time_out        : integer)
*)
            );

(*-------------------------------------------------------------------*)
            { MENSAGENS TROCADAS ENTRE OS PTAs_APLICACAO }
            010:
               (corpo010            : reg010
               );
            020, 021, 030, 040, (*041, *)042,
            050, 060, 070, (*071, *)072, (*080,
            081, *)090, 100, 110:
               (id_protocolo        : integer;
                loc_sessao          : integer;
                transf_destino      : integer;
                case tipo_1         : integer of
                   020:
                      (transf_origem : integer);
                   021, 042, 072(*, 080, 081*):
                      (causa_msg    : integer);
                   030:
                       (corpo030    : reg420);
(*
                   041:
                      (atrbarq      : ccondcriarq);
*)
                   050:
                      (id_bloco     : integer;
                       comp_dados   : integer;
                       bloco_dados  : tipo_bloco_dados);
                   060:
                      (corpo060     : reg060 );
(*
                   071:
                      (atrbopr      : ccondopr);
*)
                   040, 070, 090, 100, 110:
                      ()
               );

            { MENSAGENS NA INTERFACE PTA_APLICACAO/PTA_SESSAO }
            200, 210, (*211, *)212, 220, 240, 250, 260(, 270), 280:
               (id_transferencia    : integer;
                case tipo_2         :integer of
                   200:
                      (corpo200     : reg200);
                   210, 250, (*270, *)280:
                      (sessao       : integer);
(*
                   211:
                      (neg_tam_max_msg    : integer;
                       neg_max_tempo_espera: integer);
*)
                   212:
```

```
                    (ses_motivo     : integer);
                  220:
                    (ses            :integer;
                     janela_marcas  : integer);
                  240:
                    (sess           :integer;
                     pt_reinicio    : integer);
                  260:
                    (id_reinicio.   : integer)

                );

              { MENSAGENS NA INTERFACE USUARIO/PTA_APLICACAO }
              400:
                (corpo400           : reg400
                );
              411:
                (causa              : integer
                );
              410, 420, 430, (*431, *)432, 440,
              450, (*451, *)452, 455, (*460, 470,
              480, *)490, 491{, 500, 510, 520,
              530, 540, 550, 555):
                (ident_transf       : integer;
                 case tipo_4        : integer of
                   410, 430, 450, 455, 480, 491{,
                         500, 520, 540):
                     ();
                   420{, 530):
                     (corpo420      : reg420);
(*                 431:
                     (u_atrbarq     : ccondcriarq);
*)
                   432, 452:
                     (u_motivo      : integer);
                   440:
                     (corpo440      : reg440);

(*                 451:
                     (u_atrbopr     : ccondopr);
*)
                   (*460, 470, *)490:
                     (u_proc_usuario : str7)(;
                   510:
                     (corpo510      : reg510);
                   550:
                     (corpo550      : reg550);
                   555:
                     (t_estado      : estados)
}             )
            )
          end;
(*-------------------------------------------------------------------*)

  (* ESTRUTURAS DE CONTROLE DO PROTOCOLO DE SESSAO DO PTA *)
```

```
(* SESSAO *)
sessao = record
           transferencia              : integer;
           etd_rem                    : integer;
           porta_local                : integer;
           id_rem_sessao              : integer;
           msg_max_tam                : integer;
           tempo_espera               : integer;
           tam_janela                 : integer;
           case        boolean of
                       TRUE:  (ptr_marcas_pendentes: ^marca_pendente);
                       FALSE: (prox_sessao        : integer)
         end;

(* PAGINA DE SESSOES *)
pagina_sessoes = record
                   ptr_sessoes_livres    : integer;
                   sessoes               : array [1..NUM_SESSOES] of sessao
                 end;

(* ESTRUTURAS PARA ARMAZENAR AS MARCAS DE REINICIO *)

(* ELEMENTO DO ARRAY DE ULTIMAS MARCAS CONFIRMADAS *)
marca = record
          marc                         : integer;
          ident_reinicio               : integer
        end;

(* NODO DA LISTA DE MARCAS PENDENTES *)
marca_pendente = record
                   marc_reinicio        : integer;
                   ponto_reinicio       : integer;
                   prox_marca           : ^marca_pendente
                 end;


(*-------------------------------------------------------------------*)

  { ESTRUTURAS DE CONTROLE DO PROTOCOLO DE APLICACAO DO PTA }

  { Transferencia }
  transferencia = record
                    conta                  : tipo_conta;
                    password_conta         : tipo_password_conta;
                    proc_usuario           : str7;
                    proc_emissor           : str7;
                    estado                 : estados;
                    id_remota_transferencia : integer;
                    id_local_sessao        : integer;
                    atributos_transferencia : ptr_atributos_transf;
                    case status            : estados of
                        LIVRE:       (prox_transf   : integer);
                        INICIANDO,
                        INICIADO,
```

```
                    EM_ANDAMENTO,
                    FINALIZANDO:  (atribs_arquivo:  ptr_atr_arquivo)
                end;

{ Pagina de transferencias }
pagina_transferencia = record
                        ptr_transfs_livres: integer;
                        transfs: array [1..NUM_MAX_TRANSFERENCIAS]
                                    of transferencia
                       end;

{ Estrutura de atributos da transferencia }
atrib_transferencia = record
                        sentido_transf     : char; {( E=EMISSOR,
                                                      R=RECEPTOR);}
                        nome_arq_local     : tipo_arquivo_nome;
                        arquivo_password   : tipo_file_password;
                        modo_operacao      : char; {(A=APPEND,
                                                     D=DESTRUTIVA)}
                        tam_byte           : integer;
                        unidade            : char; {(B=BYTE, p=PALAVRA,
                                                  R=REGISTRO, P=PAGINA);}
                        tam_unidade        : integer;
                        modo_transferencia : char; {(b=BYTE, B=BLOCO,
                                                    C=BLOCO_COMPRIMIDO)}
                        reinicio           : boolean;
                        intervalo_entre_marcas: integer
                       end;

{ Estrutura de atributos do arquivo virtual }
atrib_arquivo = record
                  estrutura                : char; {(N=nao_estruturado,
                                                     R=por_registro,
                                                     P=por_pagina)}
                  tam_max                  : integer;
                  forma_unidades           : char; {(F=FIXA, V=VARIAVEL)}
                  tam_unidade              : integer;
                  ordem_unidades           : char; {(P=POR_POSICAO,
                                                     C=POR_CHAVE)}
                  forma_acesso             : char; {(S=SEQUENCIAL,
                                                     P=POR_POSICAO,
                                                     C=POR_CHAVE)}
                  posicao_chave            : integer;
                  tamanho_chave            : integer;
                  codigo_dados             : char; {(7=ISO_7, 8=ISO_8,
                                                     B=BINARIO)}
                  control_code             : char {(N=NENHUM, T=TELNET,
                                                    A=ASA,   b=BIT,
                                                    B=BYTE)}
                end;
```

```
program pta;

const

(*$I CONSTANT.PTA      *)

type

(*$I TIPOS.PTA         *)

(* VARIAVEIS GLOBAIS   *)

 var

    { VARIAVEIS GLOBAIS P/ USAR O NUCLEO DE COMUICACAO }
    pilha_proc            : pilha;
    posicao               : integer;
    ptr_nuc               : ^mensagem; { Ponteiro para a mensagem manipulada
                                         pelo nucleo }
    {--------------------------------------}

    { VARIAVEIS GLOBAIS DO PTA }
      tam_msg_max         : integer;

      (* TABELAS DE ATRIBUTOS DA OPERACAO *)
      tab_operacao        : array [1..NELEOP] of char;
      tab_unidade         : array [1..NELEUNI] of char;
      tab_modo_transf     : array [1..NELETRANS] of char;
      tab_funcao          : array [1..NELFUNC] of char;

      (* TABELAS DE ATRIBUTOS DO ARQUIVO VIRTUAL *)
      tab_estrutura       : array [1..NELESTRU] of char;
      tab_uni             : array [1..NELUNI] of char;
      tab_ordu            : array [1..NELORDU] of char;
      tab_aces            : array [1..NELACES] of char;
      tab_cod_d           : array [1..NELCOD] of char;
      tab_cont            : array [1..NELCONT] of char;

 (*-------------------------------------------------------------------*)

 (* PONTEIROS PARA BUFFERS DO NIVEL DE SESSAO *)

 (* Ponteiro para o buffer de recepcao de mensagens *)
 ptrs_msg               : ^mensagem;

 (* Ponteiros para os buffers de emissao de mensagens nas interfaces
    pta_sessao / pta_aplicacao  e pta_sessao / pta_sessao *)
 ptrs_apl               : ^mensagem;
 ptrs_ses               : ^mensagem;

 (* Estruturas de controle *)

 pag_sessoes            : pagina_sessoes;
```

```
  (* ARRAY DE ULTIMAS MARCAS PENDENTES POR SESSAO *)
  ultimas_marcas          : array [1..NUM_SESSOES] of integer;

  (* ARRAY DE ULTIMAS MARCAS CONFIRMADAS *)
  marcas_confirmadas      : array [1..NUM_SESSOES] of marca;

(*-------------------------------------------------------------------------*)

    (* PONTEIROS PARA BUFFERS DO NIVEL DE APLICACAO *)

     { Buffer para recepcao de mensagens de qualquer interface }
     ptr_msg              : ^mensagem;

     (* Pagina de transferencias *)
     pag_transfs          : pagina_transferencias;

     (* Buffer para emissao de mensagens entre os PTAs_APLICACAO *)
     ptr_apl              : ^mensagem;

     (* ARRAY COM O DESCRITOR DOS ARQUIVOS OPERADOS EM CADA TRANSFERENCIA*)
     arq                  : array [1..NUM_MAX_TRANSFERENCIAS] of text;

     (* ARRAY COM A IDENTIFICACAO DO ULTIMO BLOCO RECEBIDO / TRANSFERENCIA*)
     ult_bloco_recebido   : array [1..NUM_MAX_TRANSFERENCIAS] of integer;

{ ------------------------------------------------ }
external procedure chama_nucleo (para: ptr_bloco_nuc);

external procedure copia;
external procedure recepcao_msg;
external procedure envia_msg   (tipo: integer; var cod_retorno: integer);
external procedure enviar_msg  (var cod_retorno: integer);

(*-----------------------------------------------ENVIABL-------------*)
procedure enviabl(nome : str7;        (*nome do processo receptor da msg *)
                  porta: integer);    (*porta de recebimento da msg *)
(* DESCRICAO: Envia uma msg e devolve o controle ao nucleo (envio bloqueado).*)

var bloco       : bloco_nucleo;

begin
   bloco.funcao         := BUSCA_PROCESSO;
   bloco.msg_ptr        := addr (nome);
   chama_nucleo         (addr (bloco));
   bloco.funcao         := ENVIA_B;
   new                  (bloco.msg_ptr);
   bloco.msg_ptr^.tam   := 4;
   new                  (bloco.msg_ptr^.ptm);
   bloco.msg_ptr^.ptm^  := ptr_nuc^;
   bloco.porta          := porta;
   chama_nucleo         (addr (bloco));
   dispose              (bloco.msg_ptr);
   devolve_controle
end;
```

```
(*-------------------------------------------RECEBER-------------*)
procedure receber (porta : integer); (* porta de recepcao da msg *)
(* DESCRICAO: Recepcao bloqueada de uma msg. *)
var
    bloco     : bloco_nucleo;
    ptr_rec   : ^mensagem;
begin
   ptr_rec             := ptr_nuc;
   bloco.funcao        := RECEBE_B;
   new                 (bloco.msg_ptr);
   bloco.msg_ptr^.tam  := 4;
   bloco.porta         := porta;
   chama_nucleo        (addr (bloco));
   ptr_rec^            := bloco.msg_ptr^.ptm^;
   dispose             (bloco.msg_ptr^.ptm);
   dispose             (bloco.msg_ptr)
end;

(*-------------------------------------------RECEBE-------------*)
procedure recebe (porta      : integer;   (*porta de recepcao da msg*)
            var cod_retorno: integer);  (*indica se houve ou nao recepcao*)
(* DESCRICAO: Recepcao de  mensagem. *)
var bloco       : bloco_nucleo;
begin
   bloco.funcao        := RECEBE;
   new                 (bloco.msg_ptr);
   bloco.msg_ptr^.tam  := 4;
   bloco.porta         := porta;
   chama_nucleo        (addr (bloco));
   cod_retorno         := bloco.retcode;
   if cod_retorno       = EXITO
     then begin
            ptr_nuc^   := bloco.msg_ptr^.ptm^;
            dispose    (bloco.msg_ptr^.ptm)
          end;
   dispose             (bloco.msg_ptr)
end;

(*-------------------------------------INICIAR_PROCESSO----*)
procedure iniciar_processo (nome     : str7;      (* processo a ativar *)
                            pont_proc: ptr_proc); (* endereco do codigo *)
(* DESCRICAO: Inicia um processo *)
var
    bloco          : bloco_nucleo;
begin
   posicao         := posicao + TAMANHO;
   bloco.funcao    := INICIA_PROCESSO;
   bloco.pilha_proc:= addr (pilha_proc [posicao]);
   bloco.nom_proc  := addr (nome);
   bloco.ender_proc:= pont_proc;
   bloco.prioridade:= PRIORIDADE;
   chama_nucleo    (addr (bloco))
end;
```

```
(*-----------------------------------------DEVOLVE_CONTROLE--------*)
procedure devolve_controle;
(* DESCRICAO: Devolve o controle ao nucleo de comunicacao. *)
var
    bloco       : bloco_nucleo;
begin
   bloco.funcao := DEVOLVE;
   chama_nucleo (addr (bloco))
end;

(*-----------------------------------------DESTROI_PROCESSO---*)
procedure destroi_processo;
(* DESCRICAO: Termina a execucao de um processo. *)
var
    bloco       : bloco_nucleo;
begin
   bloco.funcao := ACABA_PROCESSO;
   chama_nucleo (addr (bloco));
   posicao      := posicao - TAMANHO
end;

(*_____________________________________LIBERA_TRANSF_____*)

procedure libera_transf (id_transf: integer);
{
 Descricao: Esta procedure inclui a transferencia id_transf, da pagina de
            transferencias, na lista de transferencias disponiveis, bem como,
            libera as estruturas de dados alocadas para esta transferencia.
                                                                          }
var i, cod_retorno              : integer;

begin      { Corpo da procedure libera_transf }

   writeln (' INICIO LIBERA_TRANSF');

   with pag_transfs.transfs[id_transf] do
      begin  { inicializa a transferencia }

         conta                  := XIVBRANCOS;
         password_conta         := SETEBRANCOS;

         proc_emissor           := PROC_NULO;
         id_remota_transferencia := INICIAL;
         id_local_sessao        := INICIAL;

         { Liberacao das estruturas de dados alocadas para a transferencia }
         if estado <> LIVRE
           then begin
{                  dispose ( atributos_transferencia );
                   dispose ( atribs_arquivo );}
                   estado := LIVRE;
                   if proc_usuario <> PROC_NULO
                     then begin  { Este nodo é o primario }
                            { Confirmacao, ao usuario, do encerramento da
                              transferencia}
```

```
                        ptr_apl^.ident_transf     := id_transf;
                        msg_envia (491, cod_retorno)
                      end
              end;

        proc_usuario            := PROC_NULO
      end;

      with pag_transfs do
        begin  { Liberacao da transferencia }
          transfs[id_transf]. prox_transfs := ptr_transfs_livres;
          ptr_transfs_livres               := id_transf
        end;

   writeln (' FIM LIBERA_TRANSF')

end;  { Fim da procedure libera_transf }

(*_____________________________________________FIM LIBERA_TRANSF__*)


{-------------------------------------ARQUIVOS 'A INCLUIR----------}
(*$I msgenvia.pas*)
{$I abretra.pas}
{$I criaarq.pas}
{$I encertra.pas}
{$I procmarc.pas}

{-------------------------------------------------FIM DA INCLUSAO----}

(*_____________________________________________CONTCOM.PAS______*)

procedure controle_comunicacao;
{
  Descricao: Esta procedure recebe e distribui todas as mensagens destinadas
             ao protocolo de aplicacao do PTA.
                                                                          }
var transf, cod_retorno              : integer;

begin   { Corpo da procedure controle_comunicacao }

  writeln      (' INICIO CONTROLE_COMUNICACAO');

  while TRUE do
   begin       { Inicio do while eterno }
      ptr_nuc                        := ptr_msg;
      receber  (PORTA_APL);

      case ptr_msg^.tipo_msg of

         010, { Pedido, do primario, de abertura de um contexto de
                transferencia. }
         020, { Confirmacao, do secundario, da abertura de um contexto de
                transferencia. }
         021, { Recusa, do secundario, da abertura de um contexto de
```

```
        transferencia. }

      210, (* Confirmacao positiva da abertura de uma sessao de
              comunicacao. *)

(*    211,   Confirmacao condicional da abertura de uma sessao de
              comunicacao. *)
      212, (* Recusa da abertura de uma sessao de comunicacao. *)

      400: { Pedido, do usuario, de abertura de um contexto de
                                                transferencia. }

        abre_transferencia;

      030, { Pedido, do primario, de criacao de uma instancia de arquivo
             virtual. }
      040, { Confirmacao positiva, do secundario, da criacao da instancia
             do arquivo virtual. }
(*    041,*)(Confirmacao condicional, do secundario, da criacao do
             arquivo virtual. }
      042, { Recusa, do secundario, da criacao da instancia do arquivo
             virtual. }
      420: { Pedido, do usuario, de criacao de um arquivo virtual. }

        cria_arquivo;

      050, { Mensagem de dados do emissor }
      090: { Mensagem, do emissor, de fim de dados }
        begin
          transf                := ptr_msg^.transf_destino;
          if (pag_transfs.transfs[transf].
                          estado  = EM_ANDAMENTO)
            then copia
        end;

      060,   { Pedido, do primario, de operacao }
      070,   { Confirmacao positiva, do secundario, do pedido de operacao }
(*    071,*) { Confirmacao condicional, do secundario, do pedido de
               operacao }
      072,   { Recusa, do secundario, do pedido de operacao }
      440:   { Pedido de operacao dentro de um contexto de transferencia }

        copia;

(*    080, *) { Pedido, do primario, de aborto do contexto de transf. }
(*    460: *) { Pedido, do usuario, de aborto do contexto de transferencia }

(*      aborta_transferencia;
*)
(*    081,*) { Pedido, do primario, de aborto da operacao de transfe. }
(*    470:*) { Pedido, do usuario, de aborto da operacao atual do contexto
               de transferência. }

(*      aborta_operacao;
*)
```

```
        100:    { Confirmacao do recebimento da mensagem de fim de dados }

          copia;

        110,    { Mensagem, do primario, de encerramento do contexto de
                  transferencia }
        490:    { Pedido, do usuario, para encerrar o contexto de transf. }

          encer_transferencia;

        260:    { Mensagem, para o emissor, com a identificacao de reinicio }
          begin
            ptr_nuc   := ptr_msg;
            transf    := ptr_msg^.id_transferencia;
            with pag_transfs.transfs[transf] do
            if (estado = EM_ANDAMENTO) AND
               (proc_emissor <> PROC_NULO)
              then enviabl (proc_emissor, PORTA_ENI)
          end;

(*      500, *) { Mensagem, do usuario, solicitando os atributos da
                  transferencia }

(*      520, *) { Mensagem, do usuario, solicitando os atributos do arquivo
                  virtual }

(*      540: *) { Mensagem, do usuario, solicitando os atributos da operacao
                  atual }
(*
          recupera_atributos;
*)

    end (* Fim do case *)

  end    { Fim do while }

end;     { Fim da procedure controle_comunicacao }

(*_____________________________________FIM CONTCOM.PAS___*)

(*_____________________________________TESTADOR.PAS______*)

procedure simulamsg (transf: integer);
(*
  Descricao: Esta procedure simula as mensagens manipuladas pelo pta para
             efeito de teste.
                                                                          *)
var opcao: char;

begin   (* Corpo da procedure simulamsg *)

  writeln (' INICIO SIMULAMSG');

  case ptr_msg^.tipo_msg of            (* Case 1 *)
```

```
400:
  with ptr_msg^.corpo400 do
    begin
       u_processo        := 'TESTA00';
       p_conta           := '00000';
       p_pass_conta      := 'teste';
       s_conta           := '00000';
       s_pass_conta      := 'teste';
       u_etd_remoto      := 10;
       u_tam_bloco_dados:= 126
    end;

420;
 begin
   ptr_msg^.ident_transf:= transf;
   with ptr_msg^.corpo420 do
     begin
        u_estrutura     := 'R';
        u_tam_max       := 10;
        u_unidades      := 'F';
        u_tam_unidade   := 80;
        u_ordem_unidades:= 'P';
        u_forma_acesso  := 'S';
        u_posicao_chave := 0;
        u_tamanho_chave := 0;
        u_codigo_dados  := '7';
        u_control_code  := 'N'
     end
 end;

440:
 begin
   ptr_msg^.ident_transf := transf;
   with ptr_msg^.corpo440 do
     begin
        user_processo    := pag_transfs.transfs[ptr_msg^.ident_transf].
                            proc_usuario;
        u_p_conta        := '00000';
        u_p_pass_conta   := 'teste';
        writeln (' Digite o nome do arquivo primario: ');
        readln (u_p_arquivo);
        u_arqp_pass      := 'teste';
        u_s_conta        := '00000';
        u_s_pass_conta   := 'teste';
        writeln (' Digite o nome do arquivo secundario: ');
        readln (u_s_arquivo);
        u_arqs_pass      := 'teste';
        u_modo_operacao  := 'A';
        writeln(' Digite a funcao do primario (E:emissor, R:receptor):');
        readln (u_funcao_primario);
        u_tam_byte_transf:= 7;
        u_unidade        := 'R';
        u_tam_unidade    := 80;
        u_modo_transf    := '0';
        writeln (' Digite a opcao de reinicio (S:sim, N:nao):');
```

```
            readln  (opcao);
            if opcao         = 'S'
              then begin
                     u_reinicio             := TRUE;
                     u_tam_janela_marcas    := 1;
                     u_intervalo_entre_marcas:= 1
                   end
              else begin
                     u_reinicio             := FALSE;
                     u_tam_janela_marcas    := 0;
                     u_intervalo_entre_marcas:= 0
                   end
          end
       end;

      490:
         begin
           ptr_msg^.ident_transf             := transf;
           ptr_msg^.u_proc_usuario           := 'TESTADO'
         end

   end;  (* Fim do case *)
   writeln (' FIM SIMULAMSG')

end; (* Fim da procedure simulamsg *)

procedure testador;
(*
  Descricao: Esta procedure realiza as funcoes de um programa usuario para
             efeito de teste do pta.
                                                                          *)
var i, codigo, trsf, cod_retorno      : integer;
    fim                               : boolean;
    nome_proc                         : str7;
    func_prim                         : char;

procedure recresp;
(*
  Descricao: Esta procedure espera e recebe a resposta do pedido do usuario
             (TESTADOR) ao PTA.
                                                                          *)
begin    (* Inicio da recresp *)

   ptr_nuc          := ptr_msg;
   receber          (PORTA_USU);

   case ptr_msg^.tipo_msg of

      411:
         begin
           writeln (' Recusa abertura contexto de transferencia');
           writeln (' MOTIVO: ', ptr_msg^.causa)
         end;

      410, 430, 432, 450, 452, 455, 491:
```

```
        begin
           writeln (' MSG do tipo: ', ptr_msg^.tipo_msg);
           trsf            :=      ptr_msg^.ident_transf;
           writeln (' Transf. local: ',ptr_msg^.ident_transf);
           if      (ptr_msg^.tipo_msg = 432) OR
                   (ptr_msg^.tipo_msg = 452)
             then writeln (' Motivo MSG: ',ptr_msg^.u_motivo)
        end
   end;       (* case *)

   devolve_controle

end;          (* Fim do recresp *)

begin (* Corpo da procedure testador *)

   writeln (' INICIO TESTADOR ');

   (* Inicializacao da procedure *)
   nome_proc:= 'CONTROL';
   fim      := FALSE;

   (* Inicio do laco de interacao com o usuario *)
   while not fim do
     begin  (* Inicio do while *)
        ptr_nuc := ptr_msg;
        writeln (' Relacao de funcoes:');
        writeln (' 1: ABRE CONTEXTO DE TRANSFERENCIA.');
        writeln (' 2: CRIA ARQUIVO VIRTUAL.');
        writeln (' 3: COPIA.');
        writeln (' 4: ENCERRA TRANSFERENCIA.');
        writeln (' 6: ACABA PROCESSO');
        writeln (' 7: LISTAR ESTRUTURAS DE CONTROLE');
        writeln (' 8: DEVOLVE CONTROLE');

        writeln (' Digite o codigo da funcao desejada: ');
        readln  (codigo);

        case codigo of   (* case 1 *)

           1: (* Abertura de um contexto de transferencia *)
             begin
                writeln (' Vou abrir um contexto de transferencia');
                ptr_msg^.tipo_msg:=400;
                simulamsg (trsf);
                enviabl  (nome_proc, PORTA_APL);
                recresp
             end;

           2: (* Criacao de um arquivo virtual *)
             begin
                writeln (' Vou criar um arquivo virtual');
                ptr_msg^.tipo_msg:=420;
                simulamsg (trsf);
                enviabl  (nome_proc, PORTA_APL);
```

```
            recresp
          end;

        3: (* Cópia do arquivo a ser transferido *)
          begin
            writeln (' Vou copiar um arquivo para/do secundario');
            ptr_msg^.tipo_msg:=440;
            simulamsg (trsf);
            func_prim       := ptr_msg^.corpo440.u_funcao_primario;
            enviabl (nome_proc, PORTA_APL);
            recresp;
            recresp;
            if func_prim = 'R'
              then devolve_controle
          end;

        4: (* Encerramento de um contexto de transferencia *)
          begin
            writeln (' Vou encerrar o contexto de transferencia');
            ptr_msg^.tipo_msg:=490;
            simulamsg (trsf);
            enviabl  (nome_proc, PORTA_APL);
            recresp
          end;

        6: (* Destruicao do processo *)
          destroi_processo;

        7: { Listagem das estruturas de controle do pta }
           begin
             writeln (' DADOS DA PAG_TRANSFERENCIAS');
             for i:=1 to NUM_MAX_TRANSFERENCIAS do
               with pag_transfs.transfs[i] do
                 begin
                   writeln (' TRANSF:',i,' TRANSF_REMOT:',
                                        id_remota_transferenc);
                   writeln (' SESSAO_LOCAL:',id_local_sessao)
                 end;

             writeln (' DADOS DA PAG_SESSOES');
             for i:=1 to NUM_SESSOES do
               with pag_sessoes.sessoes[i] do
                 begin
                   writeln (' TRANSF:',transferencia,' ETD_REM:',etd_rem);
                   writeln (' ID_REM_SESSAO:',id_rem_sessao)
                 end
           end;

        8: (* Devolvendo o controle *)
           devolve_controle

      end   (* case 1  *)
    end;    (* while  *)

  writeln   (' FIM TESTADOR')
```

```
end;          (* Fim da procedure testador *)

(*_____FIM TESTADOR.PAS_____*)

procedure inic_sessao (session: integer);
(*
  Descricao: Esta procedure inicaliza os campos de uma sessao (session)
             da pagina de sessoes.
                                                                        *)
begin   (* Corpo da procedure inic_sessao *)

   writeln (' INICIO INIC_SESSAO ');

   with pag_sessoes.sessoes [session] do
      begin
         (* Inicializa os campos da sessao *)
         transferencia := INICIAL;
         etd_rea       := INICIAL;
         porta_local   := INICIAL;
         id_rea_sessao := INICIAL;
         msg_max_tam   := INICIAL;
         tempo_espera  := INICIAL;
         tam_janela    := INICIAL;

         (* Retorna a sessao para a lista de sessoes livres *)
         prox_sessao                  := pag_sessoes.ptr_sessoes_livres;
         pag_sessoes.ptr_sessoes_livres := session
      end;

   writeln (' FIM INIC_SESSAO ')

end;  (* Fim da procedure inic_sessao *)

procedure inic_pg_sessoes;
(*
   Descricao: Esta procedure inicializa a pagina de sessoes e os arrays de
              ultimas marcas pendentes e de ultimas marcas confirmadas.
                                                                        *)
var session: integer;

begin  (* Corpo da procedure inic_pg_sessoes *)

   writeln (' INICIO INIC_PG_SESSOES');

   (* Inicializacao do header da lista de sessoes livres *)
   pag_sessoes.ptr_sessoes_livres := 1;

   (* Encadeamento dos demais nodos da lista de sessoes livres *)
   for session := 1 to NUM_SESSOES do
    begin
      inic_sessao (session);
      ultimas_marcas     [session]              := INICIAL;
      marcas_confirmadas [session].marc         := INICIAL;
      marcas_confirmadas [session].ident_reinicio := INICIAL
```

```
    end;

   (* Indicacao do final da lista *)
   pag_sessoes.sessoes   [1].prox_sessao            := VAZIA;

   writeln (' FIM INIC_PG_SESSOES')

end;  (* Fim da procedure inic_pg_sessoes *)
(*-------------------------------------------------------*)

procedure inic_pg_transfs;
{
  Descricao: Esta  procedure  inicializa  a  pagina de  transferencias  e
             as tabelas locais de atributos da transferencia e do arquivo
             virtual.
                                                                          }
var transf: integer;

begin   { Corpo da procedure inic_pg_transfs }

   writeln (' INICIO INIC_PG_TRANSFS');

   { Inicializacao do ponteiro para a lista de transferencias livres }
   pag_transfs.ptr_transfs_livres         := 1;

   { Inicializacao da lista de transferencias livres }
   for transf := 1 to NUM_MAX_TRANSFERENCIAS do
      begin
         with pag_transfs.transfs[transf] do
           begin
              estado                     := LIVRE;
              atributos_transferencia    := NIL;
              atribs_arquivo             := NIL;
              proc_usuario               := PROC_NULO
           end;
         libera_transf (transf)
      end;
   { Marcacao do final da lista }
   pag_transfs.transfs      [1].prox_transf:= VAZIA;

   (* Inicializacao da tabela local de atributos da transferencia e a
      tabela local de atributos do arquivo virtual. *)

   (* INICIALIZACAO DA TABELA DE ATRIBUTOS DA OPERACAO. *)
   tab_operacao    [1] := 'A';
   tab_operacao    [2] := 'D';

   tab_unidade     [1] := 'B';
   tab_unidade     [2] := 'p';
   tab_unidade     [3] := 'R';
   tab_unidade     [4] := 'P';

   tab_modo_transf [1] := 'b';
   tab_modo_transf [2] := 'B';
   tab_modo_transf [3] := 'C';
```

```
   tab_funcao      [1] := 'E';
   tab_funcao      [2] := 'R';

   (* INICIALIZACAO DA TABELA DE ATRIBUTOS DO ARQUIVO VIRTUAL. *)
   tab_estrutura   [1] := 'M';
   tab_estrutura   [2] := 'R';
   tab_estrutura   [3] := 'P';

   tab_uni         [1] := 'F';
   tab_uni         [2] := 'V';

   tab_ordu        [1] := 'P';
   tab_ordu        [2] := 'C';

   tab_aces        [1] := 'S';
   tab_aces        [2] := 'P';
   tab_aces        [3] := 'C';

   tab_cod_d       [1] := '7';
   tab_cod_d       [2] := '8';
   tab_cod_d       [3] := '8';

   tab_cont        [1] := 'N';
   tab_cont        [2] := 'T';
   tab_cont        [3] := 'A';
   tab_cont        [4] := 'D';
   tab_cont        [5] := 'B';

   (*-------------------------------------------------------------------*)

   writeln (' FIM INIC_PG_TRANSFS')

end;  { Fim da procedure inic_pg_transfs }

procedure inicializador;
{
 Descricao: Esta procedure  inicializa os recursos necessarios para  a
            atividade do PTA_SESSAO e do PTA_APLICACAO.              }

begin { Corpo da procedure inicializador }

   writeln (' INICIO INICIALIZADOR');

   new (ptrs_asg);
   new (ptrs_ses);
   new (ptrs_apl);

   inic_pg_sessoes;

   new (ptr_asg);
   new (ptr_apl);

   inic_pg_transfs;
```

```
   writeln (' FIM INICIALIZADOR');

end;  { Fim da procedure inicializador }

begin (* INICIO DO MODULO PTA *)

   writeln (' INICIO DO PTA');

   inicializador;

   iniciar_processo ('RECEPCA', addr (recepcao));
   iniciar_processo ('CONTROL', addr (controle));
   iniciar_processo ('TESTADO', addr (testador));

   writeln (' FIM DO PTA');

   { Incremento da posicao para nao destruir a pilha de processos }
   posicao         := posicao + TAMANHO;

   destroi_processo

end.   { Fim do programa PTA }
```

```
procedure abre_transferencia;
{
 Descricao: Esta procedure abre um contexto de transferencia a partir de uma
            solicitacao de um usuario local ou de um pta remoto.
                                                                          }

procedure aloca_transferencia (var id_transferencia, cod_retorno: integer);
{
 Descricao: Esta procedure deve alocar uma transferencia da pagina de
            transferencias.
                                                                          }
begin  { Corpo da procedure aloca_transferencia }
(*
 writeln (' INICIO ALOCA_TRANSFERENCIA');
*)
 with pag_transfs do

   if ptr_transfs_livres <> VAZIA
     then begin
            cod_retorno          := EXITO;
            { Alocacao de uma transferencia da pagina de transferencias. }
            id_transferencia     := ptr_transfs_livres;
            ptr_transfs_livres   := transfs[id_transferencia].prox_transf;

            { Inicializacao do estado da transferencia alocada }
            transfs[id_transferencia].estado := INICIANDO
          end

     else cod_retorno            := NAOHATRANSF(*;*)
(*
 writeln (' FIM ALOCA_TRANSFERENCIA')
*)
end;  { Fim da procedure aloca_transferencia }

procedure atualiza_transf (tipo_msg, id_transferencia: integer);
{
 Descricao: Esta procedure deve  atualizar uma  transferencia da  pagina de
            transferencias com as informacoes recebidas na mensagem do tipo
            "tipo_msg".
                                                                          }
begin  { Corpo da procedure atualiza_transf }
(*
   writeln (' INICIO ATUALIZA_TRANSFERENCIA');
*)
   case tipo_msg of

      010:
         with pag_transfs.transfs[id_transferencia], ptr_msg^.corpo010 do
           begin
              id_remota_transferencia       := id_trsf_origem;
              id_local_sessao               := local_sessao;
              conta                         := sec_conta;
              password_conta                := sec_pass_conta
           end;
```

```
      020:
         pag_transfs.transfs[id_transferencia].
                  id_remota_transferencia := ptr_msg^.transf_origem;

      210:
         pag_transfs.transfs[id_transferencia].
                  id_local_sessao         := ptr_msg^.sessao;

      400:
         with pag_transfs.transfs[id_transferencia], ptr_msg^.corpo400 do
            begin
               proc_usuario               := u_processo;
               conta                      := p_conta;
               password_conta             := p_pass_conta
            end
   end(*;*)   { Fim do case }
(*
  writeln (' FIM ATUALIZA_TRANSF')
*)
end;   { Fim da procedure atualiza_transf }

procedure abre_secund_transf;
{
 Descricao: Esta procedure deve alocar uma transferencia da pagina de
            transferencias, atualizar os campos desta transferencia e
            enviar uma confirmacao ao primario.
                                                                        }
var id_transferencia, cod_retorno          : integer;

begin { Corpo da procedure abre_secund_transf }
(*
   writeln (' INICIO ABRE_SECUND_TRANSF');
*)
   aloca_transferencia (id_transferencia, cod_retorno);
   if cod_retorno = EXITO
     then begin
             ptr_apl^.transf_origem        := id_transferencia;
             atualiza_transf (010, id_transferencia);
             msg_envia       (020, cod_retorno)
          end
     else msg_envia          (021, cod_retorno)(*;*)
(*
   writeln (' FIM ABRE_SECUND_TRANSF')
*)
end;  { Fim da procedure abre_secund_transf }

procedure c_posit_transf;
{
 Descricao: Esta procedure recebe uma confirmacao positiva da abertura de um
            contexto de transferencia (020) e atualiza a identificacao da
            da transferencia, no secundario, na transferencia correspondente
            da  pagina  de  transferencias,  e  comunica  a  abertura  da
            transferencia ao usuario.
                                                                        }
```

```
var id_transferencia, cod_retorno      : integer;

begin   { Corpo da procedure c_posit_transf }
(*
   writeln (' INICIO C_POSIT_TRANSF');
*)
   id_transferencia              := ptr_msg^.transf_destino;
   if pag_transfs.transfs[id_transferencia].
      estado = INICIANDO
     then begin
            atualiza_transf (020, id_transferencia);
            msg_envia       (410, cod_retorno)
          end(*;*)
(*
   writeln (' FIM C_POSIT_TRANSF')
*)
end;  { Fim da procedure c_posit_transf }

procedure c_negat_transf;
{
 Descricao: Esta procedure recebe uma confirmacao negativa sobre a abertura
            de um contexto de transferencia (021), libera os recursos
            alocados localmente para esta transferencia e avisa ao processo
            usuario que o contexto de transferencia nao foi aberto.
                                                                          }
var id_transferencia, cod_retorno      : integer;

begin  { Corpo da procedure c_negat_transf }
(*
   writeln (' INICIO C_NEGAT_TRANSF');
*)
   id_transferencia                   := ptr_msg^.transf_destino;

   if pag_transfs.transfs[id_transferencia].
      estado = INICIANDO
     then begin
            libera_transf (id_transferencia);
            msg_envia     (411, cod_retorno)
          end(*;*)
(*
   writeln (' FIM C_NEGAT_TRANSF')
*)
end; { Fim da procedure c_negat_transf }

procedure abre_primar_transf;
{
 Descricao: Esta procedure recebe um pedido de abertura de um contexto de
            transferencia do usuario, aloca uma transferencia da pagina de
            transferencias, atualiza-a com as informacoes recebidas do usuario,
            e prepara e envia um pedido de abertura de transferencia ao
            secundario.
                                                                          }
var  id_transf, cod_retorno            : integer;

begin   { Corpo da procedure abre_primar_transf }
```

```
(*
  writeln (' INICIO ABRE_PRIMAR_TRANSF');
*)
  aloca_transferencia (id_transf, cod_retorno);
  if cod_retorno = EXITO
    then begin
           (* Envio de um pedido, ao pta_sessao,  de um pedido de abertura de
              uma sessao de comunicacao *)
           ptr_apl^.id_transferencia  :=id_transf;
           atualiza_transf (400, id_transf);
           msg_envia       (200, cod_retorno)
         end
    else msg_envia         (411, cod_retorno)(*;*)
(*
   writeln (' FIM ABRE_PRIMAR_TRANSF')
*)
end;  { Fim da procedure abre_primar_transf }

procedure conf_sessao;
(*
  Descricao: Esta procedure recebe a confirmacao do pedido de abertura de uma
             sessao de comunicacao e toma as providencias cabiveis.
                                                                          *)
var transf, cod_retorno              : integer;

begin  (* Corpo da procedure conf_sessao *)
(*
   writeln (' INICIO CONF_SESSAO');
*)
   transf                                  := ptr_msg^.id_transferencia;

   case ptr_msg^.tipo_msg of

      210:  (* Confirmacao positiva da sessao de comunicacao *)
        begin
           atualiza_transf (210, transf);
           (* Envio do pedido de abertura de um contexto de transferencia *)
           ptr_apl^. corpo010.id_trsf_origem := transf;
           msg_envia       (010, cod_retorno)
        end;

      211,  (* Confirmacao condicional da sessao de comunicacao *)
      212:  (* Recusa a abertura da sessao de comunicacao *)

         begin
           libera_transf   (transf);
           cod_retorno                     := NAOABRESESSAO;
           ptr_msg^.u_proc_usuario         := pag_transfs.transfs[transf].
                                                        proc_usuario;
           msg_envia       (411, cod_retorno)
         end
    end(*;*)
(*
   writeln (' FIM CONF_SESSAO')
*)
```

```
end;    (* Fim da procedure conf_sessao *)

begin   { Corpo da procedure abre_transferencia }
(*
  writeln (' INICIO ABRE_TRANSFERENCIA');
*)
  case ptr_asg^.tipo_asg of

     810:                    { Pedido de abertura de um contexto de
                               transferencia de um pto remoto. }
        abre_secund_transf;  { Deve ser aberto um contexto secundario de
                               transferencia. }

     820:                    { Confirmacao positiva de um pedido de abertura
                               de um contexto secundario de transferencia. }
        c_posit_transf;      { Deve ser aceita a confirmacao e comunicada ao
                               usuario. }

     821:                    { Confirmacao negativa do pedido de abertura
                               de um contexto de transferencia. }
        c_negat_transf;      { Deve ser comunicada ao usuario. }

     210,                    (* Confirmacao positiva da abertura de uma
                               sessao de comunicacao.*)
     211,                    (* Confirmacao condicional da abertura de
                               uma sessao de comunicacao.*)
     212:                    (* Recusa da abertura de uma sessao de
                               comunicacao. *)
        conf_sessao;

     400:                    { Pedido de abertura de um contexto de
                               transferencia vindo do usuario. }
        abre_primar_transf;  { Deve ser aberto um contexto primario de
                               transferencia. }
  end(*;*)   { Fim do case. }
(*
  writeln (' FIM ABRE_TRANSFERENCIA')
*)
end;    { Fim da procedure abre_transferencia. }
```

```
procedure cria_arquivo;
{
 Descricao: Esta procedure cria um arquivo virtual, alocando uma estrutura de
            dados onde devem ser guardados os atributos do arquivo virtual.
            Conforme for o caso, pode enviar ou receber um pedido de criacao
            de uma instancia do arquivo virtual no secundario; enviar ou
            receber uma confirmacao positiva, condicional ou negativa da
            criacao de um arquivo virtual.
                                                                        }

procedure critica_arquivo_atributos (tipo           : integer;
                                     var cod_erro   : ccondcriarq;
                                     var cod_retorno : integer);
{
 Descricao: Esta procedure critica os atributos do arquivo virtual para
            verificar se sao compativeis com as caracteristicas do nodo local.
                                                                        }
var i                           : integer;
    k                           : boolean;

begin { Corpo da procedure critica_arquivo_atributos }
(*
   writeln (' INICIO CRITICA_ARQUIVO_ATRIBUTOS');
*)
   { Inicializacao do vetor de indicacao de erros }
   for i                := 1 to NATRBARQ do
      cod_erro [i]                      := EXITO;

   case tipo of

      830:              { Pedido do primario para a criacao de uma instancia
                          de  um  arquivo virtual. }

         with ptr_msg^.corpo830 do

            begin
               k                        := false;
               for i                    := 1 to NELESTRU do
                   if (tab_estrutura[i]     = u_estrutura)
                      then k                := true;
                   if (not k)
                      then  cod_erro[1]     := ERRO_ESTRUTURA;

               if u_tam_max > TAM_ARQUIVO_MAX
                  then cod_erro     [2]     := ERRO_TAM_ARQUIVO;

               k                        := false;
               for i                    := 1 to NELEUNI  do
                   if (tab_uni      [i]     = u_unidades)
                      then k                := true ;
               if (not k)
                  then cod_erro     [3]     := ERRO_FORMA_UNIDADES;

               if u_tam_unidade > TAM_UNIDADE_MAX
```

```
        then cod_erro      [4]     := ERRO_UNIDADE_TAM_MAX;

      k                            := false;
      for i                        := 1 to NELORDU  do
          if (tab_ordu     [i]      = u_ordem_unidade)
             then  k               := true;
      if (not k)
         then  cod_erro    [5]     := ERRO_ORDEM_UNIDADES;

      k                            := false;
      for i                        := 1 to NELACES  do
          if ( tab_aces    [i]      = u_forma_acesso )
             then  k               := true ;
      if (not k )
         then  cod_erro    [6]     := ERR_FORMA_ACESSO;

      if (u_posicao_chave + u_tamanho_chave) > u_tam_unidade
        then cod_erro      [7]     := ERRO_POSICAO_CHAVE;

      k                            := false;
      for i                        := 1 to NELCOD  do
          if (tab_cod_d    [i]      = u_codigo_dados )
             then  k               := true;
      if (not k )
         then cod_erro     [8]     := ERRO_CODIGO_DADOS;

      k                            := false;
      for i                        := 1 to NELCONT  do
          if (tab_cont     [i]      = u_control_code)
             then  k               := true;
      if ( not k)
         then  cod_erro    [9]     :=  ERRO_CONTROL_CODE

end;


420:  { Pedido do usuario para a criacao de uma instancia
        de um arquivo virtual. }

  with ptr_msg^.corpo420 do

    begin
      k                            := false;
      for i                        := 1 to NELESTRU do
          if (tab_estrutura [i]     = u_estrutura)
             then k                := true;
          if (not k)
             then  cod_erro [1]    := ERRO_ESTRUTURA;

      if u_tam_max > TAM_ARQUIVO_MAX
        then cod_erro       [2]    := ERRO_TAM_ARQUIVO;

      k                            := false;
      for i                        := 1 to NELEUNI  do
          if (tab_uni      [i]      = u_unidades)
```

```
                then k                  := true ;
            if (not k)
              then cod_erro      [3]    := ERRO_FORMA_UNIDADES;

            if u_tam_unidade > TAM_UNIDADE_MAX
              then cod_erro      [4]    := ERRO_UNIDADE_TAM_MAX;

            k                           := false;
            for i                       := 1 to NELORDU  do
                if (tab_ordu     [i]     = u_ordem_unidade)
                  then  k               := true;
            if (not k)
              then  cod_erro     [5]    := ERRO_ORDEM_UNIDADES;

            k                           := false;
            for i                       := 1 to NELACES  do
                if ( tab_aces    [i]     = u_forma_acesso )
                  then  k               := true ;
            if (not k )
              then  cod_erro     [6]    := ERR_FORMA_ACESSO;

            if (u_posicao_chave + u_tamanho_chave) > u_tam_unidade
              then cod_erro      [7]    := ERRO_POSICAO_CHAVE;

            k                           := false;
            for i                       := 1 to NELCOD  do
                if (tab_cod_d    [i]     = u_codigo_dados )
                  then  k.              := true;
            if (not k )
              then cod_erro      [8]    := ERRO_CODIGO_DADOS;

            k                           := false;
            for i                       := 1 to NELCONT  do
                if (tab_cont     [i]     = u_control_code)
                  then  k               := true ;
            if ( not k)
              then  cod_erro     [9]    :=  ERRO_CONTROL_CODE

          end

   end;   { Fim do case de tipos de mensagens }

   i                                    := 0;
   cod_retorno                          := EXITO;
   repeat
      i                                 := i + 1;
      if cod_erro               [i]     <> EXITO
        then cod_retorno                := ERRO_CRITICA
   until (cod_retorno = ERRO_CRITICA)
      OR (i           = MATRBARQ)(*;*)
(*
   writeln (' FIM CRITICA_ARQUIVO_ATRIBUTOS')
*)
end;       { Fim da procedure critica_arquivo_atributos }
```

```
procedure aloca_arquivo_estrutura (id_transf       : integer;
                                   var cod_retorno : integer);
{
 Descricao: Esta procedure aloca uma estrutura de dados onde possam ser
            armazenados os atributos de um arquivo virtual.
                                                                        }
begin      { Corpo da procedure aloca_arquivo_estrutura  }
(*
   writeln (' INICIO ALOCA_ARQUIVO_ESTRUTURA');
*)
   with pag_transfs.transfs[id_transf] do
      begin
         if     atribs_arquivo    = NIL
           then new    (atribs_arquivo);
         if     atribs_arquivo    = NIL
           then cod_retorno      := ERRO_ALOCACAO_MEMORIA
           else cod_retorno      := EXITO
      end(*;*)
(*
   writeln (' FIM ALOCA_ARQUIVO_ESTRUTURA')
*)
end;       { Fim da procedure aloca_arquivo_estrutura }

procedure atual_arquivo_estrut (tipo_msg: integer);
{
 Descricao: Esta procedure atualiza as informacoes da estrutura de atributos
            do arquivo virtual.
                                                                        }
var ptr_arquivo                  : ptr_atr_arquivo;

begin      { Corpo da procedure atual_arquivo_estrut }
(*
   writeln (' INICIO ATUAL_ARQUIVO_ESTRUT');
*)
   case tipo_msg of

     420:      { Chegou um pedido do usuario para a criacao de um arquivo
                 virtual }
         begin
            ptr_arquivo          := pag_transfs.
                                       transfs [ptr_msg^.ident_transf].
                                       atribs_arquivo;
            with ptr_arquivo^, ptr_msg^.corpo420 do
               begin
                  estrutura      := u_estrutura;
                  tam_max        := u_tam_max;
                  forma_unidades := u_unidades;
                  ordem_unidades := u_ordem_unidades;
                  forma_acesso   := u_forma_acesso;
                  posicao_chave  := u_posicao_chave;
                  tamanho_chave  := u_tamanho_chave;
                  codigo_dados   := u_codigo_dados;
                  control_code   := u_control_code
               end
         end;
```

```
      030:     { Chegou um pedido do primario para a criacao de uma instancia
                 de um  arquivo  virtual.  Deve_se  atualizar as informacoes
                 recebidas  nesta  mensagem  na  estrutura  de  atributos  do
                 arquivo. }
          begin
             ptr_arquivo          := pag_transfs.
                                        transfs [ptr_msg^.transf_destino].
                                        atribs_arquivo;
             with   ptr_arquivo^,     ptr_msg^.corpo030   do
                begin
                   estrutura       := u_estrutura;
                   tam_max         := u_tam_max;
                   forma_unidades  := u_unidades;
                   ordem_unidades  := u_ordem_unidades;
                   forma_acesso    := u_forma_acesso;
                   posicao_chave   := u_posicao_chave;
                   tamanho_chave   := u_tamanho_chave;
                   codigo_dados    := u_codigo_dados;
                   control_code    := u_control_code

                end
          end

   end(*;*)     { Fim do case de tipos de mensagens }
(*
  writeln    (' FIM ATUAL_ARQUIVO_ESTRUT')
*)
end;  { Fim da procedure atual_arquivo_estrut }

procedure prim_cria_arquivo;
{
 Descricao: Chegou um pedido (420) de um usuario para a criacao de um
            arquivo virtual.
                                                                          }
var  cod_retorno, id_transf, i    : integer;
     cod_erro                     : ccondcriarq;

begin       { Corpo da procedure prim_cria_arquivo }
(*
   writeln  (' INICIO PRIM_CRIA_ARQUIVO');
*)
   id_transf                     := ptr_msg^.ident_transf;
   critica_arquivo_atributos        (420, cod_erro, cod_retorno);
   if cod_retorno                 = EXITO
     then begin
            aloca_arquivo_estrutura (id_transf, cod_retorno);
            if cod_retorno        = EXITO
              then begin
                     atual_arquivo_estrut (420);
                     msg_envia           (030, cod_retorno)
                   end
          end
     else if cod_retorno          = ERRO_CRITICA
            then begin
```

```
                ptr_apl^.ident_transf          := id_transf;
{               for i                          := 1 to NATRBARQ do
                  ptr_apl^.u_atrbarq  [i]      := cod_erro [i];
                msg_envia              (431, cod_retorno)
}               msg_envia              (432, cod_retorno)
              end;
   if     (cod_retorno <> EXITO)
    AND  (cod_retorno <> ERRO_CRITICA)
    then begin
           ptr_apl^.ident_transf               := id_transf;
           msg_envia                  (432, cod_retorno)
         end(*;*)
(*
   writeln    (' FIM PRIM_CRIA_ARQUIVO')
*)
end;          { Fim da procedure pria_cria_arquivo }

procedure sec_cria_arquivo;
{
 Descricao: Chegou um pedido (030) de um nodo primario para a criacao de uma
            instancia de um arquivo virtual.
                                                                          }
var cod_retorno, id_transf, i     : integer;
    cod_erro                      : ccondcriarq;

begin  {  Corpo da procedure sec_cria_arquivo  }
(*
   WRITELN (' INICIO SEC_CRIA_ARQUIVO');
*)
   id_transf                      := ptr_msg^.transf_destino;

   critica_arquivo_atributos      (030, cod_erro, cod_retorno);

   if cod_retorno                 = EXITO

     then begin                   { Cria_se a instancia do arquivo virtual. }

            aloca_arquivo_estrutura       (id_transf, cod_retorno);

            if cod_retorno        = EXITO
              then begin
                     atual_arquivo_estrut (030);
                     msg_envia            (040, cod_retorno)
                   end
         end

     else if cod_retorno          = ERRO_CRITICA

           then begin             { Faz_se a aceitacao condicional da criacao
                                    de uma instancia do arquivo virtual. }
                  ptr_apl^.transf_destino := pag_transfs.transfs[id_transf].
                                             id_remota_transferencia;
(*                 for i                   := 1 to NATRBARQ do
                    ptr_apl^.atrbarq [i] := cod_erro[i];
                  msg_envia               (041, cod_retorno)
```

```
*)
                msg_envia              (042, cod_retorno)
              end;

    if    (cod_retorno <> EXITO)
      AND (cod_retorno <> ERRO_CRITICA)

      then begin           { Recusa do pedido de criacao da instancia do
                             arquivo virtual. }
             ptr_apl^.transf_destino      := pag_transfs.transfs[id_transf].
                                             id_remota_transferencia;
             msg_envia                    (042, cod_retorno)
           end(*;*)
(*
   writeln (' FIM SEC_CRIA_ARQUIVO')
*)
end;       { Fim da procedure sec_cria_arquivo }

procedure c_pos_cria_arquivo;
{
 Descricao: Esta procedure recebe uma confirmacao positiva da criacao de uma
            instancia  de  um  arquivo virtual,  do secundario, (040) e a
            comunica ao usuario.
                                                                          }
var cod_retorno              : integer;

begin         { Corpo da procedure c_pos_cria_arquivo }
(*
   writeln    (' INICIO C_POS_CRIA_ARQUIVO');
*)
   msg_envia  (438, cod_retorno)(*;*)
(*
   writeln    (' FIM C_POS_CRIA_ARQUIVO')
*)
end;          { Fim da procedure c_pos_cria_arquivo }
{
procedure cond_cria_arquivo;

 Descricao: Chegou uma  confirmacao condicional  da criacao de uma instancia de
            um arquivo virtual, do secundario, (041) que deve ser comunicada
            ao usuario.
                                                                          }
(*
var transf, i, cod_retorno   : integer;

begin  *)     { Corpo da procedure cond_cria_arquivo }
(*
   writeln    (' INICIO C_COND_CRIA_ARQUIVO');

   transf                       := ptr_msg^.transf_destino;
   ptr_usuario^.ident_transf    := transf;

   for i                        := 1 to MATRBARQ do
      ptr_usuario^. u_atrbarq[i] := ptr_msg^.atrbopr[i];
```

```
   msg_envia   (431, cod_retorno);
*)
   { Liberacao da estrutura de atributos do arquivo virtual, no primario }
   { dispose   (pag_transfs. transfs[transf].atribs_arquivo); }
(*   pag_transfs.transfs[transf].atribs_arquivo := NIL

end;  *) { Fim da procedure cond_cria_arquivo }


procedure c_neg_cria_arquivo;
{
 Descricao: Chegou uma recusa (confirmacao negativa), do secundario, para a
            criacao de uma instancia de um arquivo virtual (842) que deve
            ser comunicada ao usuario.
                                                                          }
var cod_retorno                  : integer;

begin      { Corpo da procedure c_neg_cria_arquivo }
(*
   writeln (' INICIO C_NEG_CRIA_ARQUIVO');
*)
   ptr_apl^.ident_transf         := ptr_msg^.transf_destino;
   msg_envia                        (432, cod_retorno)(*;*)
(*
   writeln (' FIM C_NEG_CRIA_ARQUIVO')
*)
end;       { Fim da procedure c_neg_cria_arquivo }

begin  { Corpo da procedure cria_arquivo }
(*
   writeln (' INICIO CRIA_ARQUIVO');
*)
   case ptr_msg^.tipo_msg of

      830:     { Chegou um pedido do primario solicitando a criacao de uma
                 instancia do arquivo virtual }
         sec_cria_arquivo;

      840:     { Chegou uma confirmacao positiva do secundario da criacao de
                 uma instancia do arquivo virtual }
         c_pos_cria_arquivo;

(*    841:*)   { Chegou uma confirmacao condicional, do secundario, da criacao
                 de uma instancia do arquivo virtual }
(*       cond_cria_arquivo;
*)

      842:     { Chegou uma confirmacao negativa, do secundario, da criacao de
                 uma instancia do arquivo virtual }
         c_neg_cria_arquivo;

      420:     { Chegou um pedido de um usuario para a criacao de um arquivo
                 virtual }
         prim_cria_arquivo
```

```
   end(*;*)        { Fim do case }
(*
   writeln     (' FIM CRIA_ARQUIVO')
*)
end;           { Fim da procedure cria_arquivo }
```

```
MODULE ptacopia;

{$E-}

const

(*$I CONSTANT.PTA*)


type

(*$I TIPOS.PTA   *)


var

(*_________________VARIAVEIS GLOBAIS P/ USAR O NUCLEO DE COMUNICACAO_____*)
   pilha_proc                    : external pilha;
   posicao                       : external integer;
   ptr_nuc                       : external ^mensagem;

(*_________________VARIAVEIS GLOBAIS DO PTA_____________________________*)

(* TABELAS DE ATRIBUTOS DA OPERACAO                                     *)

   tab_operacao                  : external array [1..MELEOP]    of char;

   tab_unidade                   : external array [1..MELEUNI]   of char;

   tab_modo_transf               : external array [1..MELETRANS] of char;

   tab_funcao                    : external array [1..MELFUNC]   of char;

   ptr_msg ,
   ptr_apl                       : external ^mensagem;

   pag_transfs                   : external pagina_transferencias;

   arq                           : external array [1..NUM_MAX_TRANSFERENCIAS]
                                            of text;

   ult_bloco_recebido            : external array [1..NUM_MAX_TRANSFERENCIAS]
                                            of integer;

(*___________VARIAVEIS GLOBAIS DO MODULO COPIA_____________________________*)

   emissores                     :          array [1..NUM_MAX_TRANSFERENCIA]
                                            of str7;
(*          enderecos dos processos emissores                             *)
   endemi                        :          array [1..NUM_MAX_TRANSFERENCIA]
                                            of ptr_proc;

(*          array de indicacao de reinicio                                *)
   retomada                      :          array [1..NUM_MAX_TRANSFERENCIA]
                                            of boolean;
```

```
(*-------------------------------------------------------------------------*)

external procedure libera_transf (    id_transf  : integer      );
external procedure chama_nucleo  (    parm       : ptr_bloco_nuc);
external procedure testador;
external procedure
        iniciar_processo        (    nome       : str7;
                                     pont_proc  : ptr_proc     );
external procedure enviabl      (    nome       : str7;
                                     porta      : integer      );
external procedure recebe       (    porta      : integer;
                                var cod_retorno: integer      );
external procedure destroi_processo;
external procedure devolve_controle;
(*-------------------------------------------------------------------------*)

procedure conclui_operacao      (transf         : integer      );
{
  Descricao: Esta procdure muda o estado da transferencia para INICIANDO e,
             caso o nodo seja o primario,  avisa ao usuario que a operacao
             foi concluida.
                                                                          }
var cod_retorno                                 : integer;

begin      { Corpo da procedure conclui_operacao }

   writeln (' INICIO CONCLUI_OPERACAO');

   with pag_transfs.transfs[transf] do
      begin
         estado                         := INICIANDO;
         dispose                        (atributos_transferencia);
         if proc_usuario                <> PROC_NULO
           then begin
                  ptr_apl^.ident_transf:= transf;
                  msgc_envia            (455, cod_retorno)
                end
      end;

  writeln (' FIM CONCLUI_OPERACAO')

end;       { Fim da procedure conclui_operacao }


procedure crit_atrib_operacao (    tipo         : integer;
                               var cod_erro     : ccondopr;
                               var cod_retorno  : integer   );
{
 Descricao: Esta procedure critica os atributos da operacao para verificar se
            sao compativeis com as caracteristicas do nodo local.
                                                                          }
var j, i                                        : integer;
    k                                           : boolean;
```

```
begin      { Corpo da procedure crit_atrib_operacao                    }

   writeln (' INICIO CRIT_ATRIB_OPERACAO');

{  Inicializacao do vetor de indicacao de erros                        }

   for i                          := 1 to MATRBOPR do
      cod_erro[i]                 := EXITO;

   j                              := 5;   { Inicio do elemento do
                                            vetor cod_erro a ser criticado   }

{  Nao sao feitas criticas dos seguintes itens:
                                                    op_conta;
                                                    op_pass_conta;
                                                    op_arquivo; e
                                                    op_arq_pass.          }

   case tipo of

      060: { Pedido de operacao proveniente do primario                 }

        with ptr_msg^.corpo060 do

         begin

            k                        := FALSE;
            for i                    := 1 to NELEOP  do
                if (tab_operacao[i]   = op_operacao )
                   then k            := TRUE;

            if     ( NOT k )
               then  cod_erro   [j]  := ERRO_OPERACAO;

            j                        := j + 1;
            k                        := FALSE;
            for i                    := 1 to NELFUNC  do
                if (tab_funcao  [i]   = op_funcao_secundario )
                   then  k           := TRUE;

            if (NOT k)
               then cod_erro    [j]  := ERRO_FUNCAO;

            j                        := j + 1;
            if op_tam_byte_transf     < 1
              then cod_erro     [j]  := ERRO_BYTE_TRANSF;

            j                        := j + 1;
            k                        := FALSE;
            for i                    := 1 to NELEUNI  do
                if ( tab_unidade[i]   = op_unidade )
                   then  k           := TRUE;

            if (NOT k)
               then  cod_erro   [j]  := ERRO_TRANSF_UNIDADE;
```

```
      j                          := j + 1;
      if (op_tam_unidade          < 1) OR
         (op_tam_unidade          > TAM_UNIDADE_MAX)
        then cod_erro     [j]    := ERR_TAM_UNIDADE;

      j                          := j + 1;
      k                          := FALSE;
      for i                      := 1 to NELETRANS  do
         if (tab_modo_transf[i] = op_modo_transf )
             then k              := TRUE;

      if (NOT k)
         then  cod_erro   [j]    := ERRO_MODO_TRANSF;

      j                          := j + 1;
      if (op_tam_janela_marcas  < 0) OR
         (op_tam_janela_marcas  > TAM_JANELA_MAX)
        then cod_erro     [j]    := ERRO_JANELA;

      j                          := j + 1;
      if (op_intervalo_entre_marcas < 0) OR
         (op_intervalo_entre_marcas > (TAM_ARQUIVO_MAX * 1000))
        then cod_erro     [j]    := ERRO_INTERVALO_MARCAS
   end;

440: { PEDIDO DE OPERACAO DO USUARIO }

  with ptr_msg^.corpo440 do
   begin
      k                          := FALSE;
      for i                      := 1 to NELEOP do
          if ( tab_operacao[i]  = u_modo_operacao )
             then  k             := TRUE;

      if (NOT k)
         then  cod_erro    [j]   := ERRO_OPERACAO;

      j                          := j + 1;
      k                          := FALSE;
      for i                      := 1 to NELFUNC do
          if ( tab_funcao  [i]  = u_funcao_primario )
             then k              := TRUE;

      if ( NOT k)
         then cod_erro     [j]   := ERRO_FUNCAO;

      j                          := j + 1;
      if u_tam_byte_transf        < 1
       then cod_erro      [j]    := ERRO_BYTE_TRANSF;

      j                          := j + 1;
      k                          := FALSE;
      for i                      := 1 to NELEUNI do
          if (tab_unidade  [i]  = u_unidade )
```

```
            then k          := TRUE;

        if ( NOT k)
           then cod_erro    [j] := ERRO_TRANSF_UNIDADE;

        j                       := j + 1;
        if (u_tam_unidade        < 1) OR
           (u_tam_unidade        > TAM_UNIDADE_MAX)
          then cod_erro     [j] := ERR_TAM_UNIDADE;

        j                       := j + 1;
        k                       := FALSE;
        for i                   := 1 to NELETRANS do
            if ( tab_modo_transf  [i]  = u_modo_transf )
               then  k          := TRUE;

        if (NOT k)
           then  cod_erro   [j] := ERRO_MODO_TRANSF;

        j                       := j + 1;
        if (u_tam_janela_marcas  < 0) OR
           (u_tam_janela_marcas  > TAM_JANELA_MAX)
          then cod_erro     [j] := ERRO_JANELA;

        j                       := j + 1;
        if (u_intervalo_entre_marcas  < 0) OR
           (u_intervalo_entre_marcas  >  (TAM_ARQUIVO_MAX * 1000))
          then cod_erro     [j] := ERRO_INTERVALO_MARCAS

      end

  end;    { Fim do case de tipos de mensagens }

  i                             := 0;
  cod_retorno                   := EXITO;

  repeat
    i                           := i + 1;

    if cod_erro                [i] <> EXITO
      then cod_retorno              := ERRO_CRITICA

  until   (cod_retorno               = ERRO_CRITICA)
    OR    ( i                        = MATRBOPR    );

  writeln (' FIM CRIT_ATRIB_OPERACAO')

end;      { Fim da procedure crit_atrib_operacao }

procedure aloca_estrut_atrib_operacao (    id_transf    : integer;
                                       var cod_retorno  : integer);

{
 Descricao: Esta procedure aloca uma estrutura de dados onde possam ser
            armazenados os atributos da operacao.
                                                                        }
```

```
begin      { Corpo da procedure aloca_estrut_atrib_operacao }

   writeln (' INICIO ALOCA_ESTRUT_ATRIB_OPERACAO');

   with pag_transfs. transfs [id_transf]  do

      begin
         if atributos_transferencia = NIL
          then NEW                 (atributos_transferencia);
         if atributos_transferencia = NIL
          then cod_retorno        := ERRO_ALOCACAO_MEMORIA
          else cod_retorno        := EXITO

      end;

  writeln (' FIM ALOCA_ESTRUT_ATRIB_OPERACAO')

end;       { Fim da procedure aloca_estrut_atrib_operacao }

procedure atual_atrib_operacao (tipo : integer);
{
 Descricao: Esta procedure atualiza as informacoes da estrutura de atributos
            da transferencia (operacao).
                                                                          }
var ptr_operacao                  : ptr_atributos_transf;

begin      { Corpo da procedure atual_atrib_operacao }

   writeln  (' INICIO ATUAL_ATRIB_OPERACAO');

   case tipo of

      448:  { Chegou um pedido de operacao do usuario }

         begin
            ptr_operacao              := pag_transfs.
                                            transfs [ptr_msg^.ident_transf].
                                            atributos_transferencia;
            with ptr_operacao^,
                 ptr_msg^.corpo448  do
               begin
                  sentido_transf      := u_funcao_primario;
                  nome_arq_local      := u_p_arquivo;
                  arquivo_password    := u_argp_pass;
                  modo_operacao       := u_modo_operacao;
                  tam_byte            := u_tam_byte_transf;
                  unidade             := u_unidade;
                  tam_unidade         := u_tam_unidade;
                  modo_transferencia  := u_modo_transf;
                  reinicio            := u_reinicio;
                  intervalo_entre_marcas:= u_intervalo_entre_marcas
               end
         end;

      868:  { Chegou um pedido de operacao do primario                    }
```

```
        begin
          ptr_operacao              := pag_transfs.
                                         transfs [ptr_msg^.
                                                     transf_destino].
                                         atributos_transferencia;
          with ptr_operacao^,
               ptr_msg^.corpo860 do
            begin
              sentido_transf        := op_funcao_secundario;
              nome_arq_local        := op_arquivo;
              arquivo_password      := op_arq_pass;
              modo_operacao         := op_operacao;
              tam_byte              := op_tam_byte_transf;
              unidade               := op_unidade;
              tam_unidade           := op_tam_unidade;
              modo_transferencia    := op_modo_transf;
              reinicio              := op_reinicio;
              intervalo_entre_marcas:= op_intervalo_entre_marcas
            end
        end

    end;    { Fim do case de tipos de mensagens                                }

    writeln (' FIM ATUAL_ATRIB_OPERACAO')

end;        { Fim da procedure atual_atrib_operacao                            }

procedure arq_abre (    tran          : integer;
                        como          : char;
                    var cod_ret       : integer);
(*
  Descricao: Esta procedure abre o arquivo local do contexto de transferencia
             "tran" para leitura ou escrita sequencial, conforme o valor de
             "como".
                                                                              *)
(* Descricao das variaveis:
     tran   : Identificacao da transferencia local.
     como   : Como o nodo local vai trabalhar (emissor ou receptor).
     cod_ret: Codigo de retorno.
     nomarq : Nome do arquivo local.                                          *)

var nomarq                          : tipo_arquivo_nome;

begin       (* Inicio da procedure arq_abre                                  *)

    writeln (' INICIO ARQ_ABRE');

    nomarq                        := pag_transfs.transfs[tran].
                                     atributos_transferencia^.nome_arq_local;
    if como                       = 'E'          (* emissor                 *)
      then  (* Deve-se abrir o arquivo para leitura                          *)
        begin
          ASSIGN                  (arq [tran], nomarq);
          RESET                   (arq [tran]);
```

```
        if IORESULT              = 255
          then                   (* Houve erro na abertura do arquivo     *)
            cod_ret              := ERROR
          else
            cod_ret              := EXITO
      end;
   if comp                       = 'R'          (* receptor               *)
    then  (* Deve-se abrir o arquivo para escrita                          *)
      begin
        ASSIGN                   (arq [tran], nomarq);
        REWRITE                  (arq [tran]);
        if IORESULT              = 255
          then                   (* Houve erro na abertura do arquivo     *)
            cod_ret              := ERROR
          else
            cod_ret              := EXITO
      end;
   writeln (' FIM ARQ_ABRE')

end;       (* Fim da procedure arq_abre                                    *)

procedure copia_primario;
   {
     Descricao: Esta  procedure  implementa os  procedimentos do  primario para
                executar uma operacao de transferencia, a partir do recebimento
                de uma mensagem de um usuario (440).
                                                                            }
   var  pri_transf, i, cod_retorno     : integer;
        cod_erro                       : ccondopr;

   begin       { Corpo da procedure copia_primario                         }

     writeln  (' INICIO COPIA_PRIMARIO');
     pri_transf                          := ptr_msg^.ident_transf;
     pag_transfs.transfs[pri_transf].estado := INICIADO;

     crit_atrib_operacao                  (440, cod_erro, cod_retorno);
     if cod_retorno                       = EXITO
      then begin
             aloca_estrut_atrib_operacao  (pri_transf, cod_retorno);
             if cod_retorno               = EXITO
               then begin
(*                     Envia uma 220 com o tamanho da janela ao
                       pta_sessao                                           *)
                    ptr_apl^.
                     id_transferencia     := pri_transf;
                    ptr_apl^.ses          := pag_transfs.
                                             transfs[pri_transf].
                                             id_local_sessao;
                    ptr_apl^.
                        janela_marcas     := ptr_msg^.corpo440.
                                                u_tam_janela_marcas;
                    msgc_envia            (220, cod_retorno);

(*                    Atualiza a estrutura de atributos da operacao     *)
```

```
                    atual_atrib_operacao(440          );

(*                    Envia uma 060 com ua pedido de operacao ao
                      pta_aplicacao parceiro da transferencia      *)
                    msgc_envia        (060, cod_retorno)
                  end
              else pag_transfs.
                     transfs[pri_transf].
                     estado             := INICIANDO
          end

      else if cod_retorno                  = ERRO_CRITICA
             then begin
                    ptr_apl^.ident_tranf  := pri_transf;
(*                  for i                 := 1 to MATRBOPR do
                     ptr_apl^.
                        u_atrbopr  [i]    := cod_erro[i];
                    msgc_envia             ( 451, cod_retorno)
*)
                    msgc_envia             ( 452, cod_retorno)
                  end;
    if    (cod_retorno                     <> EXITO          )
      AND (cod_retorno                     <> ERRO_CRITICA   )
      then begin
             ptr_apl^.ident_transf       := pri_transf;
             msgc_envia                     ( 452, cod_retorno)
           end;

     writeln (' FIM COPIA_PRIMARIO')

  end;       { Fim da procedure copia_primario }

  procedure emite_arquivo (id_transf          : integer);
  {
   Descricao: Esta procedure emite um arquivo determinado para um nodo
              receptor,  conforme as  especificacoes dos  atributos da
              transferencia e dos atributos do arquivo virtual.
                                                                      }

  var i, numbloco,
      interv, iores,
      cod_retorno                          : integer;
      bloco                                : tipo_bloco_dados;

   procedure posiciona_leitura (posicao    : integer);
   {
     Descricao: Esta procedure deve fazer o reposicionar a leitura do arquivo
                em transferencia para a unidade de endereco "posicao-1".
                                                                      }
    var ponto                              : integer;

    begin      { Corpo da procedure posiciona_leitura }

      writeln (' INICIO POSICIONA_LEITURA');
```

```
      CLOSE                          (arq [id_transf], cod_retorno);
      if cod_retorno                 = EXITO
        then begin
               ASSIGN                (arq [id_transf],
                                      pag_transfs.transfs      [id_transf].
                                                  atributos_transferencia^.
                                                  nome_arq_local);
               RESET                 (arq [id_transf]);
               if posicao            > 0
                 then for  ponto     := 0 to posicao  do
                        readln             (arq [id_transf], bloco)
             end;
      writeln (' FIM POSICIONA_LEITURA')

    end;       { Fim da procedure posiciona_leitura }

    procedure virtualiza (id_trnsf              : integer);
    {
     Descricao: Esta procedure recebe um bloco de dados do arquivo real e a
                transforma num bloco de dados do arquivo virtual.
                                                                          }
    begin   { Corpo da procedure virtualiza }

      writeln (' INICIO VIRTUALIZA');

      numbloco                       := numbloco + 1;
      with ptr_apl^ do
        begin
    {         Preparacao do bloco de dados                                }
          transf_destino             := pag_transfs.transfs[id_trnsf].
                                                  id_remota_transferencia;
          id_bloco                   := numbloco;
          comp_dados                 := length (bloco);
    {       Inicializacao do bloco de dados da mensagem                   }
          bloco_dados                := bloco;
          writeln                    (' COMP_DADOS:', comp_dados)
        end;
{       Envio da mensagem de dados                                        }
      msgc_envia                     (650, cod_retorno);

      writeln                        (' FIM VIRTUALIZA')

    end;                             { Fim da procedure virtualiza        }

  begin                              { Corpo da procedure emite_arquivo   }

    writeln                          (' INICIO EMITE_ARQUIVO');
    interv                           := pag_transfs.transfs [id_transf].
                                                  atributos_transferencia^.
                                                  intervalo_entre_marcas;
    numbloco                         := -1;

    if pag_transfs. transfs [id_transf]. atributos_transferencia^. reinicio
      then
  {       Laco de emissao de dados antes de cada nova marca               }
```

```
        while not EOF              (arq [id_transf] )  do
          begin
            i                   := 0;
            while  ( i           < interv)
              AND ( not EOF      (arq [id_transf]))
              AND ( pag_transfs.
                        transfs      [id_transf].
                        estado  =  EM_ANDAMENTO  )
            do begin
                 readln          (arq[id_transf], bloco);
{                SALVAR O IORESULT                                          }
                 iores          := IORESULT;
                 virtualiza      (id_transf);
{                Verifica se nao chegou nenhuma mensagem de controle        }
                 ptr_nuc        := ptr_msg;
                 recebe          (PORTA_EMI, cod_retorno);
                 if cod_retorno = EXITO
                   then
                        case ptr_msg^.tipo_msg of
                          260: { Mensagem, do protocolo de sessao,
                                 com a identificacao de reinicio
                                 da operacao                               }
                            begin
                              writeln (' MSG PEDIDO REINICIO',
                                       ', ID_REIN: ',
                                          ptr_msg^.id_reinicio);
                              numbloco  := ptr_msg^.id_reinicio;
                              i         := 0;
                              posiciona_leitura(ptr_msg^.
                                                id_reinicio);
                            end
                        end
                   else  devolve_controle;

                 i             := i + 1
               end;

{              Envio de uma nova marca ( se for o caso )                   }
              if    ( iores       = EXITO )
                AND ( i          >= interv)
                then begin { Protocolo de sessao deve enviar uma marca     }
                       ptr_apl^.id_transferencia :=  id_transf;
                       ptr_apl^.sessao           := pag_transfs.
                                                    transfs[id_transf].
                                                    id_local_sessao;
                       ptr_apl^.pt_reinicio      := numbloco;
                       msgc_envia                 (240, cod_retorno)
                     end
          end

      else begin
             readln   (arq               [id_transf], bloco);
             while    (not  EOF ( arq    [id_transf]      ))
                AND   (pag_transfs.transfs[id_transf].estado = EM_ANDAMENTO)
```

```
                do begin
                     writeln    (' BLOCO_DADOS:',bloco  );
                     writeln    (' NUM_BLOCO: ',numbloco);
                     virtualiza (     id_transf         );
                     readln     (arq [id_transf], bloco )
                   end;
    {           Envio do ultimo bloco de dados para o receptor                 }
                virtualiza       (     id_transf         )
              end;
      with pag_transfs.transfs        [id_transf]  do
        if estado                           = EM_ANDAMENTO
          then begin
                 estado                  := FINALIZANDO;
                 ptr_apl^.transf_destino := id_remota_transferencia;
                 ptr_apl^.loc_sessao     := id_local_sessao;
                 msgc_envia               (090, cod_retorno)
               end
          else if estado                 = FINALIZANDO
                then { Ocorreu o fechamento do contexto de transferencia    }
                     libera_transf       (id_transf       );

      writeln (' FIM EMITE_ARQUIVO')

  end;       { Fim da procedure emite_arquivo                                 }

procedure emite1;
{ DESCRICAO: Esta procedure cria uma instancia da procedure emite_arquivo.   }
var  id_transf                      : integer;

begin      { Inicio da procedure emite1                                       }

   id_transf                        := 1;
   retomada   [id_transf]           := FALSE;
   emite_arquivo                     (id_transf);
   while TRUE do destroi_processo

end;       { Fim da procedure emite1                                          }

procedure emite2;
{ DESCRICAO: Esta procedure cria uma instancia da procedure emite_arquivo.   }
var  id_transf                      : integer;

begin      { Inicio da procedure emite2                                       }

   id_transf                        := 2;
   retomada   [id_transf]           := FALSE;
   emite_arquivo                     (id_transf);
   while TRUE do destroi_processo

end;       { Fim da procedure emite2                                          }

   procedure recebe_arquivo (id_transf      : integer);
   {
     Descricao: Esta procedure recebe um determinado arquivo virtual e o
                mapeia para um tipo de arquivo aceitavel pelo nodo local.
```

```
                                                                    }
var fim_dados                          : boolean;
    bloco                              : tipo_bloco_dados;
    cod_retorno, resultado             : integer;

procedure realiza (    id_transf       : integer;
                   var bloco           : tipo_bloco_dados;
                   var cod_retorno     : integer);
(*
  Descricao: Esta procedure recebe um bloco de dados do arquivo virtual
             e o transforma num bloco de dados do arquivo real.
                                                                    *)

begin  (* Corpo da procedure realiza                                *)

   writeln (' INICIO REALIZA');
{---------------------------------}
   with ptr_msg^ do
     if    (tipo_msg                    = 850          )
       AND (id_bloco                    = 3            )
       AND (pag_transfs.transfs [transf_destino].
                      atributos_transferencia^. reinicio)
       AND (retomada [transf_destino]   = FALSE        )
         then id_bloco                 := id_bloco + 1;
{---------------------------------}
   fim_dados                           := FALSE;
   cod_retorno                         := EXITO;

   case ptr_msg^. tipo_msg of

      850:  (* Chegou uma mensagem de dados                          *)
        with ptr_msg^ do
          if id_bloco       =  (ult_bloco_recebido [transf_destino] + 1)
            then
              begin
                writeln       (' BLOCO RECEBIDO:',bloco_dados);
                writeln       (' NUM. BLOCO:', id_bloco      );
(*                Indicacao do estado do reinicio                   *)
                retomada          [transf_destino] := FALSE;

(*                Inicializacao do ultimo bloco de dados e do ultimo
                  bloco recebido.                                   *)
                ult_bloco_recebido [transf_destino] := id_bloco;

(*                Mapeia o bloco de dados do arquivo virtual, recebido
                  na mensagem, para o bloco de dados real           *)
                bloco               := bloco_dados
              end
            else
              begin
                cod_retorno         := ERROR;
                if      (id_bloco    > (ult_bloco_recebido
                                        [transf_destino] + 1)      )
                  AND   (retomada       [transf_destino] = FALSE   )
                 then
```

```
                    begin  (* Deve_se pedir o reinicio da operacao      *)

                      writeln (' ULT_BLOC_RECEBIDO [',transf_destino,']:',
                               ult_bloco_recebido  [transf_destino]);
                      writeln (' NO.BLOCO(ERRADO):', id_bloco);

                      (* Indicacao do pedido de reinicio                *)
                      retomada [transf_destino]  := TRUE;

                      ptr_apl^.id_transferencia  := transf_destino;
                      ptr_apl^. sessao           := pag_transfs.
                                                    transfs
                                                        [transf_destino].
                                                    id_local_sessao;
                      msgc_envia                  (250, cod_retorno);
                      cod_retorno                := ERROR
                    end
                end;

        098:  (* Chegou uma mensagem de fim de dados                     *)
          begin
             msgc_envia              (100, cod_retorno);
             fim_dados              := TRUE
          end(* ; *)
(*
        080,
        460:       Chegou uma mensagem avisando que o contexto de
                   transferencia foi abortado.                          *)
(*         fim_dados               := TRUE
*)
      end;    (* fim do case                                            *)

      writeln (' FIM REALIZA')

  end;        (* fim da procedure realiza                               *)

  begin       (* Corpo da procedure recebe_arquivo                      *)

     writeln (' INICIO RECEBE_ARQUIVO');
     with pag_transfs.transfs [id_transf] do
        begin
          realiza          (id_transf, bloco, cod_retorno);
          if               (not fim_dados              )
            AND            (cod_retorno   = EXITO      )
            then begin
                   writeln(' BLOCO ESCRITO:',bloco          );
                   if ult_bloco_recebido [id_transf]   <> 0
                     then writeln (arq  [id_transf]        );
                   write       (arq  [id_transf], bloco  );
                   if IORESULT                         <> EXITO
                    then writeln  (' ERRO ESCRITA DO ARQUIVO')
                 end;
          if fim_dados
            then begin
                   CLOSE          (arq [id_transf], resultado);
```

```
                if estado                       = EM_ANDAMENTO
                  then conclui_operacao         (id_transf)
                  else if estado                = FINALIZANDO
                       then
(*                             Ocorreu o fechamento do
                               contexto de transferencia        *)
                          libera_transf         (id_transf)
              end
     end;

  writeln (' FIM RECEBE_ARQUIVO')

end;      (* fim da procedure recebe_arquivo                     *)

procedure copia_secundario;
(*
  Descricao: Esta procedure implementa os procedimentos do secundario para
             executar uma  operacao  de  transferencia, a partir  do
             recebimento de uma mensagem do primario (860).
                                                                  *)
var sec_transf, i, cod_retorno          : integer;
    cod_erro                            : ccondopr;
    atrb_arq                            : ptr_atr_arquivo;

begin     (* Corpo da procedure copia_secundario                 *)

   writeln (' INICIO COPIA_SECUNDARIO');

(* Inicializacao dos nomes e enderecos dos processos emissores    *)
   emissores [1]                        := 'EMITE1';
   emissores [2]                        := 'EMITE2';
   endemi    [1]                        := addr (emite1);
   endemi    [2]                        := addr (emite2);

   sec_transf                           := ptr_msg^.transf_destino;

   crit_atrib_operacao                   (860, cod_erro, cod_retorno);
   if cod_retorno                        = EXITO     { if 1          }
     then begin
            aloca_estrut_atrib_operacao       (sec_transf, cod_retorno);
            if cod_retorno              = EXITO
              then begin
                     (* Envia uma 220 com o tamanho da janela ao
                        pta_sessao                                *)
                     ptr_apl^.id_transferencia:= sec_transf;
                     ptr_apl^.ses             := pag_transfs.
                                                 transfs[sec_transf].
                                                 id_local_sessao;
                     ptr_apl^.janela_marcas   := ptr_msg^.corpo860.
                                                 op_tam_janela_marcas;
                     msgc_envia               (220, cod_retorno);

                     pag_transfs.
                         transfs[sec_transf].
                         estado               := EM_ANDAMENTO;
```

```
                    atual_atrib_operacao      (060);

                    (* abertura do arquivo secundario para a operacao *)
                    atrb_arq                  := pag_transfs.
                                                 transfs[sec_transf].
                                                 atribs_arquivo;

                    with pag_transfs.transfs[sec_transf].
                                 atributos_transferencia^ do

                      begin
(*                      PRE_CONDICAO DO CASE                          *)
                        cod_retorno       := ERROR;
                        case atrb_arq^.forma_acesso of (* case 4      *)
                          'S': (* Acesso sequencial                  *)
                             arq_abre (sec_transf, sentido_transf,
                                       cod_retorno                );
                          'P': (* Acesso por posicao                 *)
                             begin end; (* Substituir esta linha pela
                                           chamada   da   procedure
                                           apropriada                *)
                          'C': (* Acesso por chave                   *)
                             begin end (* Substituir esta linha
                                          pela chamada da procedure
                                          apropriada                 *)
                        end;   (* Fim do case 4                      *)

                        if cod_retorno     <> EXITO
                          then begin
                                 cod_retorno     := ABERTURA_ARQUIVO;
                                 writeln (' ERRO ABERTURA ARQUIVO');
                                 ptr_apl^.
                                   transf_destino:= pag_transfs.
                                            transfs[sec_transf].
                                            id_remota_transferencia;
                                 msgc_envia       (072, cod_retorno)
                               end
                          else
                            if sentido_transf     = 'R'
                             then ult_bloco_recebido
                                  [sec_transf] := -1
                      end
                  end       { fim do if 1                             }
          end;
    if cod_retorno                          = EXITO
      then begin
(*            Envia uma confirmacao positiva do pedido de
              operacao ao primario                                    *)
            ptr_apl^.transf_destino         := pag_transfs.
                                               transfs[sec_transf].
                                               id_remota_transferencia;
            msgc_envia                      (070, cod_retorno);
            if pag_transfs.
                  transfs [sec_transf].
                  atributos_transferencia^.
```

```
                    sentido_transf              = 'E'
                then begin
                      cod_retorno              := ERROR;
                      case atrb_arq^.forma_acesso of       (* case 5     *)
                        'S':          (* Acesso sequencial                *)
                           begin
                              pag_transfs.
                                  transfs [sec_transf],
                                  proc_emissor := emissores [sec_transf];
                              iniciar_processo  ( emissores [sec_transf],
                                                  endemi    [sec_transf]);
                              devolve_controle
                           end;

                        'P': (* Acesso por posicao *)
                          begin end; (* Substituir esta linha pela
                                        chamada da procedure apropriada  *)
                        'C':          (* Acesso por chave                 *)
                          begin end  (* Substituir esta linha pela
                                        chamada da procedure apropriada  *)
                      end             (* Fim do case 5                    *)
                    end
          end
       else if cod_retorno                          = ERRO_CRITICA
             then begin
                   ptr_apl^.transf_destino     := pag_transfs.
                                                      transfs[sec_transf].
                                                  id_remota_transferencia;

{                  for i                       := 1 to NATRBOPR do
                     ptr_apl^.atrbopr [i]     := cod_erro [i];
                   asgc_envia                   (071, cod_retorno)
}                  asgc_envia                   (072, cod_retorno)
                  end
             else begin
                   ptr_apl^.transf_destino  .  := pag_transfs.
                                                      transfs[sec_transf].
                                                  id_remota_transferencia;
                   asgc_envia                  (072, cod_retorno)
                  end;

      writeln (' FIM COPIA_SECUNDARIO')

  end;       (* Fim da procedure copia_secundario *)

  procedure conf_positiva_operacao;
  (*
    Descricao: Esta procedure executa no primario e processa a recepcao da
               confirmacao positiva de cada pedido de operacao  (070).
                                                                          *)
  var  transf                  : integer;
       atrb_trsfs              : ptr_atributos_transf;
       atrb_arq                : ptr_atr_arquivo;
       cod_retorno             : integer;
```

```
begin  (* Corpo da procedure conf_positiva_operacao *)

   writeln (' INICIO CONF_POSITIVA_OPERACAO');

(* Inicializacao dos nomes e enderecos dos processos emissores          *)
   emissores [1]                        := 'EMITE1';
   emissores [2]                        := 'EMITE2';
   endemi    [1]                        := addr (emite1);
   endemi    [2]                        := addr (emite2);

   transf                     := ptr_msg^. transf_destino;
   with pag_transfs.transfs [transf]  do
      begin
         atrb_trsfs           := atributos_transferencia;
         atrb_arq             := atribs_arquivo;
         estado               := EM_ANDAMENTO
      end;

   (* abertura do arquivo primario para a operacao *)
   if atrb_trsfs^.sentido_transf = 'E'
     then
      begin
         cod_retorno          := ERROR;       (* PRE_CONDICAO DO CASE   *)
         case atrb_arq^.forma_acesso of       (* case 1                 *)
            'S':                              (* Acesso sequencial      *)
              arq_abre        (transf, 'E', cod_retorno);
            'P':                              (* Acesso por posicao     *)
              begin end;                      (* Substituir esta linha
                                                 pela chamada da
                                                 procedure apropriada   *)
            'C':                              (* Acesso por chave *)
              begin end                       (* Substituir esta linha
                                                 pela chamada da
                                                 procedure apropriada   *)
         end;                                 (* Fim do case 1          *)

         if cod_retorno       <> EXITO
           then begin
                 cod_retorno  := ABERTURA_ARQUIVO;
                 writeln      (' ERRO/NOME_ARQ:',atrb_trsfs^.
                                                 nome_arq_local);
                 ptr_apl^. transf_destino    := pag_transfs.
                                                  transfs [transf].
                                               id_remota_transferencia;
                 msgc_envia   (881, cod_retorno   )
                 writeln      (' ABORTAR OPERACAO')
                end
           else begin
                  ptr_apl^.ident_transf  := transf;
                  msgc_envia   (450, cod_retorno);
                  cod_retorno               := ERROR;
                  case atrb_arq^.forma_acesso of (* case 2            *)
                    'S':                       (* Acesso sequencial   *)
                      begin
                        pag_transfs.
```

```
                    transfs [transf].
                    proc_emissor := emissores [transf];
                  iniciar_processo  ( emissores [transf],
                                      endemi    [transf]);
                  devolve_controle
                end;
              'P':                          (* Acesso por posicao   *)
                begin end;                  (* Substituir esta linha
                                               pela chamada da
                                               procedure apropriada *)
              'C':                          (* Acesso por chave     *)
                begin end                   (* Substituir esta linha
                                               pela chamada da
                                               procedure apropriada *)
            end                             (* Fim do case 2        *)
          end
      end
    else
     begin
                                            (* Escrita destrutiva   *)
       cod_retorno                  := ERROR;(*PRE_CONDICAO DO CASE  *)
       case atrb_arq^.forma_acesso of       (* case 3               *)
         'S':                               (* Acesso sequencial    *)
           arq_abre                  (transf, 'R', cod_retorno);
         'P':                               (* Acesso por posicao   *)
           begin end;                       (* Substituir esta linha
                                               pela chamada da
                                               procedure apropriada *)
         'C':                               (* Acesso por chave     *)
           begin end                        (* Substituir esta linha
                                               pela chamada da
                                               procedure apropriada *)
       end;                                 (* Fim do case 3        *)

       if cod_retorno                <> EXITO
         then begin
                cod_retorno          := ABERTURA_ARQUIVO;
                writeln               (' ERRO/NOME_ARQ:',
                                         atrb_trsfs^.nome_arq_local);
{               ptr_apl^.transf_destino:= pag_transfs.
                                             transfs[transf].
                                             id_remota_transferencia;
                osgc_envia             (881, cod_retorno)
}               writeln                (' ABORTAR OPERACAO')
              end
         else begin
                ptr_apl^.
                    ident_transf     := transf;
                osgc_envia            (450, cod_retorno);
                ult_bloco_recebido
                           [transf] := -1
              end
     end;

  writeln (' FIM CONF_POSITIVA_OPERACAO')
```

```
   end;       (* fim da procedure conf_positiva_operacao *)
(*
   procedure conf_condicional_operacao;

     Descricao: Esta procedure  recebe uma confirmacao condicional, provinda do
                secundario,  de um  pedido de  operacao  (071),  e  avisa ao
                usuario as condicoes impostas pelo secundario para a realizacao
                da operacao.
                                                                              *)
(* var i, cod_retorno, transf              : integer;

   begin    Corpo da procedure conf_condicional_operacao                     *)
(*
      transf                               := ptr_msg^.transf_destino;
      ptr_apl^.ident_transf             := transf;

      for i                                := 1 to NATRBOPR do
         ptr_apl^.u_atrbopr [i]         := ptr_msg^.atrbopr[i];

      msgc_envia                            (451, cod_retorno);
*)
(*    Liberacao da estrutura de atributos da operacao no primario           *)
(*    dispose (pag_transfs.transfs[transf]. atributos_transferencia);       *)
(*
   end;    Fim da procedure conf_condicional_operacao *)

   procedure conf_negativa_operacao;
   (*
     Descricao: Esta procedure recebe uma recusa, provinda do secundario, a um
                pedido de operacao (072) e a avisa ao usuario.
                                                                              *)
   var cod_retorno, transf                 : integer;

   begin   (* Corpo da procedure conf_negativa_operacao *)

      WRITELN (' INICIO CONF_NEGATIVA_OPERACAO');

      transf                               := ptr_msg^.transf_destino;
      ptr_apl^.ident_transf             := transf;
      msgc_envia                            (452, cod_retorno);

   (* Liberacao da estrutura de atributos da operacao no primario           *)
      dispose                               (pag_transfs.transfs[transf].
                                                atributos_transferencia);
      writeln (' FIM CONF_NEGATIVA_OPERACAO')

  end;       (* Fim da procedure conf_negativa_operacao *)

{$E+}
procedure copia;
{
 Descricao: Esta  procedure  permite  ao  usuario  executar  a  operacao  de
            transferencia de  arquivo  desejada.  A  partir de uma  mensagem
            recebida, esta procedure pode iniciar, continuar ou concluir uma
```

```
        operacao de transferencia.
                                                                    }

  begin   (* Corpo da procedure copia *)

    writeln (' INICIO COPIA');

    case ptr_msg^.tipo_msg of

      050,                          (* Chegou uma mensagem de dados       *)
      090:                          (* Chegou uma mensagem de fim de dados*)
        case pag_transfs.
               transfs[ptr_msg^.transf_destino].
               atribs_arquivo^.forma_acesso  of   (*   case  2          *)
          'S':                      (* Acesso sequencial                  *)
            recebe_arquivo  (ptr_msg^.transf_destino);
          'P':                      (* Acesso por posicao                 *)
            begin end;              (* Substituir esta linha pela
                                       chamada da procedure apropriada    *)
          'C':                      (* Acesso por chave                   *)
            begin end               (* Substituir esta linha pela
                                       chamada da procedure apropriada    *)
        end;                        (* Fim do case 2 *)

      060:                          (* Chegou um pedido de operacao de
                                       transferencia do primario.         *)
        copia_secundario;

      070:                          (* Chegou uma confirmacao positiva
                                       para um pedido de operacao.        *)
        conf_positiva_operacao;
{*
      071:                          Chegou uma confirmacao condicional
                                       de um pedido de operacao.          *)
(*        conf_condicional_operacao;
*)
      072:                          (* Chegou uma recusa para um pedido
                                       de operacao.                       *)
        conf_negativa_operacao;

(*    080: *)                       (* Mensagem de aborto do contexto
                                       de transferencia                   *)
(*      if pag_transfs.transfs     [ptr_msg^.transf_destino].
           atributos_transferencia = 'R'
         then recebe_arquivo      (ptr_msg^.transf_destino);
*)
      160:                          { Confirmacao do recebimento da
                                      mensagem de fim de dados            }
        begin
          if pag_transfs.
               transfs[ptr_msg^.transf_destino].
               estado           = FINALIZANDO
            then conclui_operacao (ptr_msg^.transf_destino)
        end;
```

```
        440:                            (* Chegou um pedido de operacao
                                           de transferencia de um usuario.   *)
          copia_primario

(*      460: *)                         (* Chegou um pedido de aborto do
                                           contexto de transferencia         *)
(*         if pag_transfs.transfs       [ptr_msg^.ident_transf].
             atributos_transferencia = 'R'
           then recebe_arquivo          (ptr_msg^.ident_transf)
*)
   end;                                 (* Fim do case de tipos de mensagens. *)

   writeln (' FIM COPIA')

end;       (* Fim da procedure copia *)
{$E-}

procedure msgc_envia (    tipo         : integer;
                      var cod_retorno  : integer);
{
 Descricao: Esta procedure prepara e envia as mensagens ao protocolo de sessao,
            de onde devem ser encaminhadas ao destino, ou ao processo  usuario,
            durante a copia do arquivo.
                                                                              }
var transf                              : integer;
   proc_sessao                          : str7;

begin  { Corpo da procedure msgc_envia }

   writeln (' INICIO MSGC_ENVIA');
   cod_retorno                          := EXITO;
   proc_sessao                          := 'RECEPCA';
   ptr_muc                              := ptr_apl;

  case tipo of

     850:      { Deve ser enviada, ao receptor, uma mensagem com dados do
                 arquivo virtual. }
       begin
         ptr_apl^.tipo_msg              := 850;
         ptr_apl^.id_protocolo          := ID_APLICACAO;

         enviabl                         (proc_sessao, PORTA_SES)

       end;

     860:      { Deve ser pedido ao secundario a aceitacao dos atributos de
                 uma operacao de transferencia.  Este procedimento deve ser
                 executado  apos a  chegada  de  um  pedido de  operacao de
                 transferencia (440) provindo do usuario. }
       begin
         transf                         := ptr_msg^.ident_transf;
         ptr_apl^.tipo_msg              := 860;
         ptr_apl^.id_protocolo          := ID_APLICACAO;
         ptr_apl^.transf_destino        := pag_transfs. transfs[transf].
```

```
                                                id_remota_transferencia;

        with ptr_apl^.corpo060,
             ptr_msg^.corpo440  do
          begin
                                { Insercao dos atributos da operacao }
            op_conta                    := u_s_conta;
            op_pass_conta               := u_s_pass_conta;
            op_arquivo                  := u_s_arquivo;
            op_arq_pass                 := u_arqs_pass;
            op_operacao_modo            := u_modo_operacao;
            if u_funcao_primario         = 'E'
              then op_funcao_secundario := 'R'
              else op_funcao_secundario := 'E';
            op_tam_byte_transf          := u_tam_byte_transf;
            op_unidade                  := u_unidade;
            op_tam_unidade              := u_tam_unidade;
            op_modo_transf              := u_modo_transf;
            op_reinicio                 := u_reinicio;
            op_tam_janela_marcas        := u_tam_janela_marcas;
            op_intervalo_entre_marcas   := u_intervalo_entre_marcas
          end;

        enviabl                     (proc_sessao, PORTA_SES)

      end;

    070:     { Deve ser enviada,  ao primario, uma confirmacao positiva do
               pedido de operacao de transferencia. Este procedimento deve
               ser  executado  apos o  processamento  bem  sucedido,  pelo
               secundario,  de um  pedido de operacao (060) provindo do
               primario. }
      begin
        with ptr_apl^ do
          begin
            tipo_msg            := 070;
            id_protocolo        := ID_APLICACAO;
          end;

        enviabl                     (proc_sessao, PORTA_SES)

      end;
(*
    071: *)  { Deve ser  enviada,  ao primario, uma  confirmacao  condicional
               do pedido de operacao de transferencia. Este procedimento deve
               ser  executado apos o  processamento com  sucesso condicional,
               pelo secundario, de um pedido de operacao (060) provindo do
               primario. }
(*
      begin
        with ptr_apl^ do
          begin
            tipo_msg            := 071;
            id_protocolo        := ID_APLICACAO;
          end;
```

```
        enviabl                    (proc_sessao, PORTA_SES)

      end;
*)
    872:    { Deve ser enviada, ao primario, uma recusa a um pedido de
              operacao de transferencia. Este procedimento deve ser
              executado apos o processamento sec sucesso, pelo secundario,
              de um pedido de operacao (860) provindo do primario. }
      begin
        with ptr_apl^ do
          begin
            tipo_msg           := 872;
            id_protocolo       := ID_APLICACAO;
          end;

        enviabl                 (proc_sessao, PORTA_SES)

      end;

    890:    (* Deve ser enviada, ao receptor, u'a mensagem de fim de dados *)
      begin
        with ptr_apl^ do
          begin
            tipo_msg           :=890;
            id_protocolo       := ID_APLICACAO
          end;

        enviabl                 (proc_sessao, PORTA_SES)

      end;

    100:    { Deve ser avisado, ao emissor, o recebimento da mensagem de
              fim de dados (890). }
      begin
        with ptr_apl^ do
          begin
            transf_destino     := pag_transfs.
                                    transfs[ptr_msg^.transf_destino].
                                    id_remota_transferencia;
            id_protocolo       := ID_APLICACAO;
            tipo_msg           := 100
          end;

        enviabl                 (proc_sessao, PORTA_SES)

      end;

    220:    (* Deve ser enviada, ao pta_sessao local, o tamanho da janela
              de marcas *)

      begin
        ptr_apl^.tipo_msg      := 220;

        enviabl                 (proc_sessao, PORTA_SES)
```

```
      end;

    240: { Deve ser enviado, ao pta_sessao local, uma ordem para ele inserir
           uma mensagem com marca de reinicio no fluxo de dados.}
      begin
        ptr_apl^.tipo_msg          :=240;

        enviabl                     (proc_sessao, PORTA_SES)

      end;

    250: { Deve ser enviado, ao protocolo de sessao local, um pedido
           de reinicio da operacao de transferencia. }

      begin
        ptr_apl^.tipo_msg          := 250;
        enviabl                     (proc_sessao, PORTA_SES)
      end;

    450:  { Deve ser avisado, ao usuario, que o pedido de operacao foi
            aceito.  Este procedimento deve ser tomado apos a recepcao
            de uma  mensagem  de  aceitacao dos  atributos da operacao
            (070), provinda do secundario. }
      begin
        transf                     := ptr_msg^. transf_destino;

        with ptr_apl^ do
          begin
            tipo_msg               := 450;
            ident_transf           := transf
          end;

        enviabl                     (pag_transfs.
                                         transfs[transf].proc_usuario,
                                     PORTA_USU                      )

      end;
(*
    451: *)  { Deve ser avisado,  ao usuario,  que o pedido de operacao foi
               aceito condicionalmente.  Este  procedimento deve ser tomado
               apos a recepcao de uma mensagem de aceitacao condicional dos
               atributos da  operacao (071),  provinda do secundario, ou
               apos o  processamento de  um  pedido de operacao, feito pelo
               usuario, por parte do primario. }
(*
      begin
        transf                     := ptr_msg^. transf_destino;

        with ptr_apl^ do
          begin
            tipo_msg               := 451;
            ident_transf           := transf
          end;
```

```
        enviabl                 (pag_transfs.
                                     transfs[transf].proc_usuario,
                                 PORTA_USU                      )

      end;
*)
    452:    { Deve ser avisado ao usuario que o pedido de operacao foi
              recusado. Este procedimento deve ser executado apos a
              recepcao de uma mensagem de recusa de uma operacao (072),
              provinda do secundario, ou apos o processamento de um pedido
              de operacao do usuario por parte do primario. }
      begin
        transf                  := ptr_msg^. transf_destino;

        with ptr_apl^ do
          begin
            tipo_msg            := 452;
            ident_transf        := transf
          end;

        enviabl                 (pag_transfs.
                                     transfs[transf].proc_usuario,
                                 PORTA_USU                      )

      end;

    455:    { Deve ser avisado, ao usuario, que a operacao atual de
              transferencia foi concluida. }
      begin
        ptr_apl^.tipo_msg       := 455;
        transf                  := ptr_apl^.ident_transf;

        enviabl                 (pag_transfs.
                                     transfs[transf].proc_usuario,
                                 PORTA_USU                      )

      end
(*
    510: *) { Deve ser enviado, ao usuario, os atributos atuais do contexto
              de transferencia. Isto deve ocorrer apos a recepcao de uma
              mensagem, do usuario, pedindo os atributos atuais do contexto
              de transferencia. }
(*
      begin
        transf                  := ptr_msg^.ident_transf;
        ptr_apl^.tipo_msg       := 510;
        ptr_apl^.ident_transf   := transf;

        with ptr_apl^.corpo510 do
          begin
            with pag_transfs. transfs[transf] do
              begin
                trsf_processo   := proc_usuario;
                trsf_conta      := conta;
                trsf_pass_conta:= password_conta;
```

```
                trsf_estado    := estado
              end;
          end;

        enviabl                    (pag_transfs.
                                       transfs[transf].proc_usuario,
                                    PORTA_USU                       )

      end;

    530: *)  { Devem ser enviados, ao usuario, os atributos do arquivo
               virtual. Isto ocorre apos a recepcao de uma mensagem
               (520), do usuario, pedindo os atributos do arquivo
               virtual.}
(*
      begin
         transf                  := ptr_msg^. ident_transf;
         ptr_apl^.tipo_msg       := 530;
         ptr_apl^.ident_transf   := transf;

         with ptr_apl^.corpo420 do
            with pag_transfs.transfs [transf]. atribs_arquivo^ do
               begin
                  u_estrutura       := estrutura;
                  u_tam_max         := tam_max;
                  u_forma_unidades  := forma_unidades;
                  u_tam_unidades    := tam_unidades;
                  u_ordem_unidades  := ordem_unidades;
                  u_forma_acesso    := forma_acesso;
                  u_posicao_chave   := posicao_chave;
                  u_tamanho_chave   := tamanho_chave;
                  u_codigo_dados    := codigo_dados;
                  u_control_code    := control_code
               end;

         enviabl                    (pag_transfs.
                                       transfs[transf].proc_usuario,
                                    PORTA_USU                       )

      end;

    550: *)  { Deve ser enviado, ao usuario, os atributos da operacao atual.
               Isto ocorre apos o recebimento de uma mensagem (540), do
               usuario, pedindo os atributos atuais da operacao. }
(*
      begin
         transf                  := ptr_msg^.ident_transf;
         ptr_apl^.tipo_msg       := 550;

         with ptr_apl^.corpo550,
              pag_transfs.
                 transfs[transf].
                 atributos_transferencia^   do
          begin
             trsf_funcao_primario     := sentido_transf;
```

```
                trsf_arq_local          := nome_arq_local;
                trsf_chav_arq           := arquivo_password;
                trsf_modo_operacao      := modo_operacao;
                trsf_tam_byte           := tam_byte;
                trsf_unidade            := unidade;
                trsf_comp_unidade       := tam_unidade;
                modo_trsf               := modo_transferencia;
                trsf_reinicio           := reinicio;
                trsf_interv_entre_marcas := intervalo_entre_marcas
             end;

        enviabl                  (pag_transfs.
                                      transfs[transf].proc_usuario,
                                  PORTA_USU                        )

      end;

    555: *)  { Deve ser enviado, ao usuario, um aviso de que os atributos
               solicitados nao existem. Isto ocorre apos o recebimento de
               uma mensagem (520 ou 540), do usuario, sem condicoes
               para o atendimento destas mensagens. }
(*
      begin
        transf                  := ptr_msg^.ident_transf;
        with ptr_apl^ do
          begin
            tipo_msg            := 555;
            ident_transf        := transf;
            t_estado            := pag_transfs.transfs[transf].estado
          end;

        enviabl                  (pag_transfs.
                                      transfs[transf].proc_usuario,
                                  PORTA_USU                        )

      end
*)
  end;    { Fim do case }

  writeln (' FIM MSGC_ENVIA')

end;      { Fim da procedure msgc_envia  }
{$E+}
MODEND.
```

```
(* Isto é o inicio do arquivo recatrib.pas *)

(* ESTA PROCEDURE AINDA NAO FOI IMPLEMENTADA *)

procedure recupera_atributos;
(*
  Descricao: Esta procedure deve reaver os atributos atuais da  transferencia
             ou do arquivo virtual ou da operacao atual ao usuario, de acordo
             com a mensagem recebida.
                                                                          *)
var cod_retorno: integer;

begin   (* Corpo da procedure recupera_atributos *)

   case ptr_msg^.tipo_msg of

      500: { Mensagem, do usuario, solicitando os atributos da
             transferencia }
         if pag_transfs.transfs[ptr_msg^.ident_transf].estado <> LIVRE
           then msg_envia (510, cod_retorno)
           else msg_envia (555, cod_retorno);

      520: { Mensagem, do usuario, solicitando os atributos do arquivo
             virtual }
         if pag_transfs.transfs[ptr_msg^.ident_transf].estado <> LIVRE
           then msg_envia (530, cod_retorno)
           else msg_envia (555, cod_retorno);

      540: { Mensagem, do usuario, solicitando os atributos da operacao
             atual }
         if pag_transfs.transfs[ptr_msg^.ident_transf].estado = EM_ANDAMENTO
           then msg_envia (550, cod_retorno)
           else msg_envia (555, cod_retorno)

   end   (* Fim do case *)

end;   (* Fim da procedure recupera_atributos *)
```

```
{ Isto é o inicio do arquivo abortopr.pas }

(* ESTA PROCEDURE AINDA NAO FOI IMPLEMENTADA *)

procedure aborta_operacao;
{
  Descricao: Esta procedure deve  abortar uma  operacao de  transferencia que
             esta em andamento em um contexto de transferencia, como resposta
             a um pedido  do  usuario (470)  ou de  um  pedido de primario
             (081).
                                                                            }
var transf, cod_retorno: integer;

begin  { Corpo da procedure aborta_operacao }

   case ptr_msg^.tipo_msg of

      470:  { Chegou um pedido de aborto da operacao atual, provindo do
                usuario. }
         begin
            transf := ptr_usuario^.ident_transf;

            with pag_transfs.transfs[transf] do
               if (estado <> LIVRE) AND (estado <> INICIANDO)
                 then begin
                        { Fechar o arquivo local }
                        { dispose (atributos_transferencia);}
                        atributos_transferencia := NIL;
                        estado := INICIANDO;
                        msg_envia (081, cod_retorno)
                      end;
            msg_envia (480, cod_retorno)
         end;

      081:  { Chegou um pedido de aborto da operacao atual, provindo do
                primario. }
         begin
            transf := ptr_apl^.transf_destino;
            with pag_transfs.transfs[transf] do
               if (estado <> LIVRE) AND (estado <> INICIANDO)
                 then begin
                        { Fechar arquivo local }
                        { dispose (atributos_transferencia);}
                        atributos_transferencia := NIL;
                        estado := INICIANDO
                      end
         end

   end   (* Fim do case *)

end;  { Fim da procedure aborta_operacao }
```

```
{ Isto é o inicio do arquivo abortran.pas }

(* ESTA PROCEDURE AINDA NAO FOI IMPLEMENTADA *)

procedure aborta_transferencia;
{
 Descricao: Esta procedure deve abortar o contexto de transferencia como
            resposta a um pedido do usuario (460), ou de um pedido do
            primario (080).
                                                                        }
var transf, cod_retorno: integer;

begin   { Corpo da procedure aborta_transferencia }

   case ptr_msg^.tipo_msg of

      460:  { Chegou um pedido de aborto do contexto de transferencia
                provindo do usuario. }
         begin
            msg_envia (080, cod_retorno);
            msg_envia (270, cod_retorno);

            transf := ptr_usuario^.ident_transf;
            with pag_transfs.transfs[transf] do
               if estado = EM_ANDAMENTO
                 then if proc_copia <> NIL
                        then estado := FINALIZANDO
                        else libera_transf (transf)
         end;

      080:  { Chegou um pedido de aborto do contexto de transferencia
                proveniente do primario. }
         begin
            transf := ptr_apl^.ident_transf;
            with pag_transfs.transfs[transf] do
               if estado <> LIVRE
                 then if proc_copia <> NIL
                        then estado := FINALIZANDO
                        else libera_transf (transf)
         end

   end  { Fim do case de tipos de mensagens }

end;  { Fim da procedure aborta_transferencia }
```

```
procedure encer_transf;
{
  Descricao: Esta procedure deve encerrar o contexto de transferencia como
             resposta a um pedido do usuario (490), ou de um pedido  do
             primario (110).
                                                                        }
var transf, cod_retorno                    : integer;

begin          { Corpo da procedure encer_transf }

   writeln     (' INICIO ENCER_TRANSF');

   case ptr_msg^.tipo_msg of

      490:     { Chegou um pedido de encerramento do contexto de transferencia
                 provindo do usuario. }
         begin
            transf                          := ptr_msg^.ident_transf;
            pag_transfs.transfs[transf].estado := FINALIZANDO;

               { Mensagem de encerramento do contexto de transferencia para o
                 secundario. }
            msg_envia         (110, cod_retorno);

               { Pedido de encerramento da sessao de comunicacao }
            msg_envia         (280, cod_retorno);

               { Liberacao dos recursos da transferencia }
            libera_transf     (transf)

         end;

      110:     { Chegou, no secundario, um aviso para encerrar o contexto de
                 transferencia. }
         libera_transf        (ptr_msg^.transf_destino)

   end;        { Fim do case }

   writeln     (' FIM ENCER_TRANSF')

end;           { Fim da procedure encer_transf }
```

```
procedure msg_envia   ( tipo_msg          : integer;
                        var cod_retorno   : integer);
{
 Descricao: Esta procedure prepara e envia as mensagens ao protocolo de sessao,
          de onde devem ser encaminhadas ao destino, ou ao processo usuario.
                                                                          }
var transf                                : integer;
   nome_proc                              : str7;

begin          { Corpo da procedure msg_envia }

   writeln     (' INICIO MSG_ENVIA');

   cod_retorno                            := EXITO;
   nome_proc                              := 'RECEPCA';  (* RECEPCAO_MSG *)
   ptr_nuc                                := ptr_apl;

   case tipo_msg of

      010:     { Deve ser enviada uma mensagem ao secundario pedindo a abertura
                 de um contexto de transferencia. Este procedimento ocorre apos
                 o recebimento de uma mensagem (400), do usuario, pedindo a
                 abertura de um contexto de transferencia. }
         begin

           ptr_apl^.tipo_msg                := 010;

           with ptr_apl^.corpo010 do
              begin
                 protocolo                  := ID_APLICACAO;
                 sec_conta                  := ptr_msg^.corpo400.s_conta;
                 sec_pass_conta             := ptr_msg^.corpo400.s_pass_conta;
                 { A identificacao da transferencia origem foi inserida na
                   mensagem na procedure abre_primar_transf. }
                 { O campo local_sessao indicara a id_sessao no secundario}
                 local_sessao               := -pag_transfs.
                                                  transfs[id_trsf_origem].
                                                  id_local_sessao;
              end;

           enviabl     (nome_proc, PORTA_SES)

         end;

      020:     { Deve ser enviada uma confirmacao positiva da abertura de um
                 contexto secundario de transferencia. Este procedimento ocorre
                 sempre que chega um pedido de abertura de um contexto
                 secundario de transferencia (010) que obtem sucesso.      }
         begin

           with ptr_apl^ do
              begin
                 tipo_msg                   := 020;
                 id_protocolo               := ID_APLICACAO;
```

```
        loc_sessao              := pag_transfs.
                                      transfs[transf_origem].
                                      id_local_sessao;
        transf_destino          := ptr_msg^.corpo010.id_trsf_origem
      end;

    enviabl     (nome_proc, PORTA_SES)

  end;

021:    { Deve ser enviada uma confirmacao negativa da abertura de um
          contexto secundario de transferencia. Este procedimento ocorre
          sempre que chega um pedido de abertura de um contexto
          secundario de transferencia (010) que nao obtem sucesso. }
  begin

    with ptr_apl^ do
      begin
        tipo_msg                := 021;
        id_protocolo            := ID_APLICACAO;
        transf_destino          := ptr_msg^.corpo010.
                                          id_trsf_origem;
        causa_msg               := cod_retorno
      end;

    enviabl     (nome_proc, PORTA_SES)

  end;

030:    { Deve ser pedido ao secundario a criacao de uma instancia de
          um arquivo virtual. Este procedimento deve ser tomado quando
          chega um pedido do usuario para a criacao de um arquivo
          virtual (420).. }
  begin

    transf                      := ptr_msg^.ident_transf;
    ptr_apl^.tipo_msg           := 030;
    ptr_apl^.id_protocolo       := ID_APLICACAO;
    ptr_apl^.loc_sessao         := pag_transfs.transfs [transf].
                                      id_local_sessao;
    ptr_apl^.transf_destino     := pag_transfs.transfs [transf].
                                      id_remota_transferencia;
    move (ptr_msg^.corpo420, ptr_apl^.corpo030, 14);

    enviabl     (nome_proc, PORTA_SES)

  end;

040:    { Deve ser confirmada, ao primario, a criacao de uma instancia
          de um arquivo virtual. Este procedimento ocorre apos a
          recepcao de um pedido de criacao de uma instancia de arquivo
          virtual (030) e seu processamento com sucesso. }
  begin
    transf                      := ptr_msg^.transf_destino;
    with ptr_apl^ do
```

```
            begin
               tipo_msg                := 040;
               id_protocolo            := ID_APLICACAO;
               loc_sessao              := pag_transfs.transfs[transf].
                                             id_local_sessao;
               transf_destino          := pag_transfs.transfs[transf].
                                             id_remota_transferencia
            end;

         enviabl     (nome_proc, PORTA_SES)

      end;

(*
   041: *) { Deve ser confirmada condicionalmente, ao primario, a criacao
             de uma  instancia de um  arquivo virtual.  Este procedimento
             ocorre  apos  a  recepcao e  o  processamento,  com  sucesso
             condicional,  de um  pedido de criacao  de uma  instancia de
             arquivo virtual.(030). }
(*       begin

         transf                        := ptr_msg^.transf_destino;

         with ptr_apl^ do
            begin
               tipo_msg                := 041;
               id_protocolo            := ID_APLICACAO;
               transf_destino          := pag_transfs.transfs[transf].
                                             id_remota_transferencia
            end;

         enviabl     (nome_proc, PORTA_SES)

      end;
*)
   042:  { Deve  ser  indicado,  ao primario,  a recusa da criacao de uma
           instancia de um arquivo virtual. Este procedimento ocorre apos
           a  recepcao  e o  processamento,  sem sucesso, de um pedido de
           criacao de uma instancia de um arquivo virtual (030).}
      begin
         transf                  := pag_transfs.
                                       transfs[ptr_msg^.transf_destino].
                                       id_remota_transferencia;
         with ptr_apl^ do
            begin
               tipo_msg          := 042;
               id_protocolo      := ID_APLICACAO;
               transf_destino    := pag_transfs.transfs[transf].
                                       id_remota_transferencia;
               causa_msg         := cod_retorno
            end;

         enviabl     (nome_proc, PORTA_SES)

      end;
```

```
(*
     080: *)  { Deve ser enviado ao secundario um pedido de aborto do contexto
                de transferencia. Anteriormente deve ter havido um pedido de
                aborto da transferencia vindo do usuario. }
(*
        begin

           transf                    := ptr_msg^. ident_transf;

           with ptr_apl^ do
              begin
                 tipo_msg            := 080;
                 id_protocolo        := ID_APLICACAO;
                 transf_destino      := pag_transfs.transfs[transf].
                                           id_remota_transferencia;
                 causa_msg           := MOTIVO_EXTERNO
              end;

           enviabl     (nome_proc, PORTA_SES)

        end;
*)
(*      081:*) { Deve ser enviado ao secundario um pedido de aborto da
                 operacao de transferencia. Anteriormente deve ter havido
                 um pedido de aborto da operacao atual vindo do usuario. }
(*         begin

           transf                    := ptr_msg^. ident_transf;

           with ptr_apl^ do
              begin
                 tipo_msg            := 081;
                 id_protocolo        := ID_APLICACAO;
                 transf_destino      := pag_transfs.transfs[transf].
                                           id_remota_transferencia;
                 causa_msg           := MOTIVO_EXTERNO
              end;

           enviabl     (nome_proc, PORTA_SES)

        end;
*)
     110:     { Deve ser avisado, ao secundario, para encerrar o contexto de
                transferencia. Isto ocorre apos o recebimento de uma mensagem
                de encerramento do contexto de transferencia, provinda do
                usuario. }
        begin
           transf                    := ptr_msg^.ident_transf;
           with ptr_apl^ do
              begin
                 tipo_msg            := 110;
                 id_protocolo        := ID_APLICACAO;
                 transf_destino      := pag_transfs.transfs[transf].
                                           id_remota_transferencia
              end;
```

```
        enviabl     (nome_proc, PORTA_SES)

      end;

    200: { Deve ser pedido, ao protocolo de sessao, a abertura de uma
           sessao de comunicacao. Este procedimento deve ser feito
           apos o recebimento de uma mensagem 400. }

      with ptr_apl^.corpo200, ptr_msg^.corpo400 do

        begin
          ptr_apl^.tipo_msg       := 200;
          ses_etd_remoto          := u_etd_remoto;
          ses_max_tam_msg         := u_tam_bloco_dados + 12;
          ses_tempo_espera        := RETARDO_MEDIO * 5;

          enviabl     (nome_proc, PORTA_SES)
        end;
(*
    270:  *) { Deve ser avisado, ao protocolo de sessao, para abortar a
               sessao de comunicacao. Isto ocorre apos o recebimento  de
               uma  mensagem,  do  usuario,  para  abortar o contexto de
               transferencia. }
(*
      begin
         transf                   := ptr_msg^.ident_transf;
         with ptr_apl^ do
           begin
             tipo_msg             := 270;
             sessao               := pag_transfs.transfs[transf].
                                       id_local_sessao
           end;

        enviabl     (nome_proc, PORTA_SES)

      end;
*)
    280:     { Deve ser avisado, ao protocolo de sessao, para encerrar a
               sessao de comunicacao. Isto ocorre apos o recebimento  de
               uma  mensagem,  do primario,  para encerrar o contexto de
               transferencia. }
      begin
         transf                   := ptr_msg^.ident_transf;
         with ptr_apl^ do
           begin
             tipo_msg             := 280;
             sessao               := pag_transfs.transfs[transf].
                                       id_local_sessao
           end;

        enviabl     (nome_proc, PORTA_SES)

      end;
```

```
410:     { Deve ser confirmada a abertura de um contexto de transferencia
           ao usuario.  Este  procedimento ocorre apos a recepcao de uma
           confirmacao positiva da abertura de um contexto secundario de
           transferencia (020). }
   begin

     transf                    := ptr_msg^.transf_destino;

     with ptr_apl^ do
        begin
          tipo_msg             := 410;
          ident_transf         := transf
        end;

     enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

   end;

411:     { Deve ser avisado ao usuario que a abertura de um contexto de
           transferencia foi rejeitada pelo secundario (021) ou pelo
           primario. }
   begin
     transf                    := ptr_msg^.id_transferencia;
     with ptr_apl^ do
        begin
          tipo_msg             := 411;
          causa                := ptr_msg^.causa_msg * 1000 +
                                  SEC_NACK_TRANSF
        end;

     enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

   end;

430:     { Deve ser avisado ao usuario que foi criado um arquivo virtual.
           Este procedimento ocorre apos a recepcao de uma confirmacao
           positiva, do secundario, sobre a criacao de uma instancia de
           um arquivo virtual (040). }
   begin
     transf                    := ptr_msg^.transf_destino;

     with ptr_apl^ do
        begin
          tipo_msg             := 430;
          ident_transf         := transf
        end;

     enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

   end;
(*
431: *)  { Deve ser avisado ao usuario que chegou, do secundario, uma
           confirmacao condicional da criacao de uma instancia de um
           arquivo virtual (041), ou o primario nao aceitou todos
           os atributos propostos pelo usuario. }
```

```
(*      begin
          transf                    := ptr_apl^.ident_transf;
          ptr_apl^.tipo_msg         := 431;

          enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

        end;
*)
     432:     { Deve ser avisado ao usuario que a criacao do arquivo virtual
                foi recusada. }
        begin

          transf                    := ptr_apl^.ident_transf;

          with ptr_apl^ do
           begin
              tipo_msg              := 432;
              u_motivo              := cod_retorno
            end;

          enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

        end;

     480:     { Deve ser confirmado ao usuario o aborto da operacao atual de
                transferencia. Isto deve ser feito apos o processamento de
                um pedido de aborto da operacao (470), vindo do usuario.}
        begin

          transf                    := ptr_msg^. ident_transf;

          with ptr_apl^ do
             begin
                tipo_msg            := 480;
                ident_transf        := transf
             end;

          enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

        end;

     491:     { Deve ser avisado, ao usuario, que o contexto de transferencia
                foi encerrado. }
        begin
           transf                   := ptr_apl^.ident_transf;
           ptr_apl^.tipo_msg        := 491;

           enviabl     (pag_transfs.transfs[transf].proc_usuario, PORTA_USU)

        end

   end;                 { Fim do case }

   writeln              (' FIM MSG_ENVIA')
```

```
end;                    { Fim da procedure msg_envia }
```

```
MODULE PTA_SESSAO;
{$E-}

const

{ ---------------------CONSTANTES P/ NUCLEO DE COMUNICACAO---- }
   PORTA_ST                = 4;  { PORTA P/ RECEPCAO DE MSGS DO N.TRANSPORTE}
   PORTA_SES               = 5;  { PORTA P/ RECEPCAO DE MSGS DO N.APLICACAO }
   PORTA_APL               = 7;  { PORTA P/ ENVIO    DE MSGS AO N.APLICACAO }

(*_____________________CONSTANTES DO PTA_SESSAO_____________*)
   PTA_SESSAO              = 5;
   ETD                     = 10;

   VAZIA                   = -1;
   EXITO                   = 0;
   ERROR                   = -1;
   INICIAL                 = -1;

   TAM_BLOCO_MAX           = 126;
   MAIOR_MENSAGEM          = 200;
   MAIOR_TEMPO_ESPERA      = 1000;
   NUM_SESSOES             = 2;
   TAM_JANELA_MAX          = 10;
   CONTA_TAM               = 15;
   TAM_PASSWORD            = 8;

   NUM_TENTATIVAS          = 3;
   TOLERANCIA              = 1;

(* TABELA DE ERROS *)

NAOABRESESSAO              = 1005;
NAOHASESSAO                = 1010;
ERRO_TAM_MSG               = 1020;
ERRO_TEMPO_ESPERA          = 1030;
JANELA_ESGOTADA            = 1040;
SEC_COND_SESSAO            = 1050;
SEC_RECUSOU_SESSAO         = 1060;
PEDIDOUSUARIO              = 1070;

(*_____________________________________________________*)

type

   STR7                    = string [7];
   aptmsg                  = ^mensagem;
   tipo_cod_erro           = array  [1..2] of integer;
   tipo_conta              = string [CONTA_TAM];
   tipo_password_conta     = string [TAM_PASSWORD];

(*_____________________________________________________*)

     { DEFINICAO DAS ESTRUTURAS DE DADOS }
```

```
    { TIPOS DE MENSAGENS NA INTERFACE PTA_APLICACAO / PTA_SESSAO }.

    { 200:Mensagem pedindo a abertura de uma sessao de comunicacao }
    { Resposta do protocolo de sessao a um pedido de abertura de uma sessao
      de comunicacao}
    { 210:Confirmacao positiva }
    { 211:Confirmacao condicional }
    { 212:Confirmacao negativa (rejeicao do pedido) }
    (*220:Mensagem com o tamanho da janela de marcas pendentes. *)
    { 240:Pedido para o protocolo de sessao enviar uma marca de reinicio }
    { 250:Pedido para o protocolo de sessao reiniciar a transferencia }
    { 260:Mensagem com a identificacao de reinicio p/ o protocolo de
         aplicacao                                    }
    { 270:Pedido para abortar a sessao de comunicacao  }
    { 280:Pedido para encerrar a sessao de comunicacao }

  reg010 = record
             protocolo              : integer;
             local_sessao           : integer;
             id_trsf_origem         : integer;
             sec_conta              : tipo_conta;
             sec_pass_conta         : tipo_password_conta
           end;

  reg200 = record
             ses_etd_remoto         : integer;
             ses_max_tam_msg        : integer;
             ses_tempo_espera       : integer
           end;

  mensagem = record
              case boolean of
               TRUE : (grande       : string [MAIOR_MENSAGEM]);
               FALSE:
               (
                tipo_msg            : integer;
                case tipo0          : integer of

(*-------------------------------------------------------------------*)

                (* MENSAGENS TROCADAS ENTRE OS PTAs_SESSAO *)

                710, 740, 750, 760, 761, 770,
                (*780, *)790:
                  (protocolo        : integer;
                   id_sessao        : integer;
                   case tipo10      : integer of
                      710:
                         (porta_entrada   : integer;
                          etd_origem      : integer;
                          tam_mensag_max  : integer;
                          temp_espera     : integer);

                      740, 750, 760, 761, 770:
```

```
                    (marca_reinicio  : integer);
(*
                 780:
                    (motivo          : integer);
*)
                 790:
                    ()
              );

           720, (*721, *)722:
              (ids_protocolo     : integer;
               sess_origem       : integer;
               sess_destino      : integer;
               case tipo20       : integer of
                  720, 722:
                     ()      (*;
                  721:
                     (tam_msg_max: integer;
                      time_out   : integer)
*)
              );


(*-------------------------------------------------------------------*)
           { MENSAGENS TROCADAS ENTRE OS PTAs_APLICACAO }

           010:
              (corpo010       : reg010);
           020:
             (id_protocolo    : integer;
              loc_sessao      : integer;
              transf_destino  : integer;
              case tipo_1     : integer of
                020: (transf_origem     : integer));

           { MENSAGENS NA INTERFACE PTA_APLICACAO/PTA_SESSAO }

           200, 210, (*211, *)212, 220, 240, 250, 260{, 270}, 280:

              (id_transferencia          : integer;
               case tipo_2               : integer of
                  200:
                     (corpo200           : reg200);
                  210, 250, (*270, *)280 :
                     (sessao             : integer);
(*
                  211:
                     (neg_tam_max_msg    : integer;
                      neg_max_tempo_espera: integer);
*)
                  212:
                     (ses_motivo         : integer);
                  220:
                     (ses                : integer;
                      janela_marcas      : integer);
```

```
                    240:
                       (sess              : integer;
                        pt_reinicio       : integer);
                    260:
                       (id_reinicio       : integer)

                  )
               )

            end;
(*-------------------------------------------------------------------------*)

(* ESTRUTURAS DE CONTROLE DO PROTOCOLO DE SESSAO DO PTA *)

(* SESSAO                                                  *)
sessao = record
           transferencia               : integer;
           etd_rem                     : integer;
           porta_local                 : integer;
           id_rem_sessao               : integer;
           msg_max_tam                 : integer;
           tempo_espera                : integer;
           tam_janela                  : integer;
           case boolean of
              TRUE : (ptr_marcas_pendentes : ^marca_pendente);
              FALSE: (prox_sessao          : integer)
         end;

(* PAGINA DE SESSOES *)
pagina_sessoes = record
                   ptr_sessoes_livres : integer;
                   sessoes            : array [1..NUM_SESSOES] of sessao
                 end;

(* ESTRUTURAS PARA ARMAZENAR AS MARCAS DE REINICIO *)

(* ELEMENTO DO ARRAY DE ULTIMAS MARCAS CONFIRMADAS *)
marca = record
          marc                       : integer;
          ident_reinicio             : integer
        end;

(* NODO DA LISTA DE MARCAS PENDENTES *)
marca_pendente = record
                   marc_reinicio      : integer;
                   ponto_reinicio     : integer;
                   prox_marca         : ^marca_pendente
                 end;


(*-------------------------------------------------------------------------*)

{ VARIAVEIS GLOBAIS }

  var
```

```
(*-------------------------------------------------------------------------*)
  (* Ponteiro usado pelo nucleo de comunicacao       *)
  ptr_nuc                 : external ^mensagem;

  (* Ponteiro para o buffer de recepcao de mensagens *)
  ptrs_msg                : external ^mensagem;

  (* Ponteiro para o buffer de emissao de mensagens entre os ptas_sessao *)
  ptrs_ses                : external ^mensagem;

  (* Ponteiro para o buffer de emissao de mensagens na interface pta_sessao /
     pta_aplicacao *)
  ptrs_apl                : external ^mensagem;

  (* Estruturas de controle *)
  pag_sessoes             : external pagina_sessoes;

  (* ARRAY DE ULTIMAS MARCAS PENDENTES POR SESSAO *)
  ultimas_marcas          : external array [1..NUM_SESSOES] of integer;

  (* ARRAY DE ULTIMAS MARCAS CONFIRMADAS *)
  marcas_confirmadas      : external array [1..NUM_SESSOES] of marca;


(*-------------------------------------------------------------------------*)

external procedure processa_marca;
external function  identif_reinicio (sessao, marca   : integer): integer;
external procedure reinicia;
external procedure inic_sessao      (session         : integer);
external procedure enviabl          (   nome         : str7;
                                        porta        : integer);
external procedure receba           (   porta        : integer;
                                     var cod_retorno : integer);
external procedure devolve_controle;

(*-------------------------------------------------------------------------*)

procedure abrecon (
                (*   etd_secundario  : integer;                          *)
                (*   processo_origem : str7;                             *)
                 var porta_local     : integer;
                (*   tam_max_msg     : integer;                          *)
                 var cod_retorno     : integer );
(* DESCRICAO: Esta procedure simula a primitiva abrecon do niv. de transporte*)
begin
  porta_local                       := PORTA_ST;
  cod_retorno                       := EXITO
end;    (* Fim da abrecon   *)

(*-------------------------------------------------------------------------*)
```

```
procedure envmsg (porta              : integer;
                  ptr                : aptmsg;
                  var cod_retorno    : integer);
(* DESCRICAO: Esta procedure simula a primitiva envmsg do nivel de transporte.
*)
begin    (* Inicio da envmsg *)
   ptr_nuc                           := ptr;
   enviabl  ('RECEPCA',  porta);
   cod_retorno                       := EXITO
end;     (* Fim da envmsg    *)

(*-------------------------------------------------------------------------*)

procedure recmsg (porta              : integer;
                  ptr                : aptmsg;
                  var cod_retorno    : integer);
(* DESCRICAO: Esta procedure simula a primitiva recmsg do nivel de transporte.
*)
begin    (* Inicio da recmsg *)

   ptr_nuc                           := ptr;
   recebe       (porta, cod_retorno)

end;     (* Fim da recmsg    *)

(*-------------------------------------------------------------------------*)


{$E+}
procedure recepcao_msg;
(*
   Descricao: Esta  procedure  recebe  e  distribui  (para  as  procedures
              apropriadas)  todas as  mensagens destinadas ao protocolo de
              sessao do pta.
                                                                         *)
var proc_origem                      : char;
    port, repete, session,
    cod_retorno, tam_pacote          : integer;
    sempre                           : boolean;

begin      (* Corpo da procedure recepcao_msg *)

   writeln (' INICIO DO MODULO RECEPCAO_MSG');
   port                         := PORTA_ST;     (* TESTE *)
   session                      := 1;
   sempre                       := TRUE;

   while sempre do          (* laco eterno *)     (* while 1 *)
     begin

       cod_retorno              := ERROR;
       while cod_retorno        <> EXITO do      (* while 2 *)
         begin
           repete               := 0;
           (* while 3 *)
```

```
          while   (repete      < NUM_TENTATIVAS)
              AND (cod_retorno <> EXITO)         do
            begin
               (* Espera msg do pta_aplicacao local *)
               ptr_nuc         := ptrs_msg;
               receba (PORTA_SES, cod_retorno);
               proc_origem     := 'a';       (* msg veio do pta_aplicacao *)
               if cod_retorno  <> EXITO      (* if 1 *)
                 then begin
(*____OPCAO P/ O TESTE_____________________________________________________*)
                         recmsg        (port, ptrs_msg, cod_retorno);
                         if cod_retorno        = EXITO
                           then proc_origem   := 's'
                           else devolve_controle
                       end                   (* Fim do if 1      *)
(*____OPCAO ORIGINAL_______________________________________________________*)
(*                       Espera conexao de transporte p/ contatos
                         iniciais                                  *)
(*                       esperacon    (port, ..., cod_retorno);   *)
(*                       Alocacao de uma sessao da pag_sessoes p/ guardar
                         os dados da nova conexao de transporte *)
(*                       aloca_sessao (session, cod_retorno);
*)
(*                       Espera msgs das sessoes de comunicacoes
                         estabelecidas                             *)
(*                     while (session     <  NUM_SESSOES) AND
                           (cod_retorno <> EXITO) do
                         begin
                            session         := session + 1;
                            with pag_sessoes.sessoes[session] do
                               if porta_local <> INICIAL
                                 then recmsg (porta_local, ptrs_msg,
                                              tam_pacote, cod_retorno) *)
(*                          end;
                       if cod_retorno         = EXITO
                         then proc_origem    :='s'  msg veio do pta_ses*)
(*                         else begin
*)                                (* Reinicio do identificador de sessoes*)
(*                                 session   := 1;
                                   devolve_controle                  *)
(*                               end
                     end    *)(* fim do if 1 *)
(*_____________________________________________________________________*)

            end              (* fim do while 3 *)
        end;                 (* fim do while 2 *)

      case ptrs_msg^.tipo_msg of

        010:
          begin
            { Inicializacao da id da sessao secundaria p/
              comunica-la ao pta_aplicacao secundario.     }
            with ptrs_msg^.corpo010 do
              if local_sessao         < EXITO
```

```
            then begin
                   local_sessao   := -local_sessao;
                   local_sessao   := pag_sessoes.
                                        sessoes[local_sessao].
                                        id_rem_sessao
                 end;
         prepara_msg               (proc_origem)
      end;

   020:
      begin
         if proc_origem             = 'a'
           then pag_sessoes.sessoes [ptrs_msg^ .loc_sessao].
                    transferencia  := ptrs_msg^.transf_origem;
         prepara_msg               (proc_origem)
      end;

   021, 030, 040, (*041, *)042, 050,     060, 070,
   (*071, *)072, (*080, *)081, 090, 100, 110:
      prepara_msg                  (proc_origem);

   200, 720, 721, 722:
      abre_sessao;

   220: { Mensagem com o tamanho da janela de marcas a ser usado
          durante a operacao de transferencia }
      begin
         session                                      := ptrs_msg^.ses;
         pag_sessoes.sessoes[session]. tam_janela     := ptrs_msg^.
                                                         janela_marcas;
         ultimas_marcas     [session]                 := INICIAL;
         marcas_confirmadas [session]. marc           := INICIAL;
         marcas_confirmadas [session]. ident_reinicio := INICIAL;
         anula_lista_marcas (session)
      end;

   710:
      begin
         ptrs_msg^.porta_entrada    := PORTA_ST;
                                    (* pag_sessoes.sessoes[session].
                                                   porta_local;*)
         abre_sessao
      end;

   240, 740, 750:
      processa_marca;

   250, 760, 770:
      reinicia;
(*
   270, 780:
      aborta_sessao;
*)
   280, 790:
      encer_sessao
```

```
      end;    (* fim do case *)

    end       (* fim do while 1 _ laco eterno  *)

end;          (* Fim da procedure recepcao_msg *)
{$E-}

procedure abre_sessao;
(*
  Descricao: Esta procedure abre uma sessao de comunicacao a partir de uma
             solicitacao do pta_aplicacao local ou de outro pta_sessao.
                                                                          *)
procedure critic_atributos_sessao (tipo         : integer;
                                   var cod_erro : tipo_cod_erro);
(*
  Descricao: Esta procedure verifica se os atributos da sessao sao compativeis
             com as caracteristicas do nodo local.
                                                                          *)
var i                           : integer;

begin      (* Corpo da procedure critic_atributos_sessao *)

   WRITELN (' INICIO CRITIC_ATRIBUTOS_SESSAO');

   for i                        := 1 to 2 do
      cod_erro [i]              := 0;

   case tipo of

      200:    (* Mensagem, do pta_aplicacao, pedindo a abertura de uma sessao
                 de comunicacao *)
         begin

            if ptrs_msg^.corpo200.ses_max_tam_msg > MAIOR_MENSAGEM
              then cod_erro [1]                  := ERRO_TAM_MSG;

            if ptrs_msg^.corpo200.ses_tempo_espera > MAIOR_TEMPO_ESPERA
              then cod_erro [2]                  := ERRO_TEMPO_ESPERA

         end;

      718:    (* Pedido, do primario, para a abertura de uma sessao de
                 comunicacao *)
         begin
            if ptrs_msg^.tam_mensag_max          > MAIOR_MENSAGEM
              then cod_erro [1]                  := ERRO_TAM_MSG;

            if ptrs_msg^.temp_espera             > MAIOR_TEMPO_ESPERA
              then cod_erro [2]                  := ERRO_TEMPO_ESPERA
         end
   end;    (* Fim do case *)

   writeln (' FIM CRITIC_ATRIBUTOS_SESSAO')
```

```
end;       (* Fim da procedure critic_atributos_sessao *)

procedure aloca_sessao (var id_sessao       : integer;
                        var cod_retorno     : integer);
(*
  Descricao: Esta procedure deve alocar uma sessao da pagina de sessoes.
                                                                        *)
begin      (* Corpo da procedure aloca_sessao *)

   WRITELN (' INICIO ALOCA_SESSAO');
   cod_retorno                          := EXITO;
   with pag_sessoes do

      if ptr_sessoes_livres             <> VAZIA
        then begin
               (* Alocacao de uma sessao da pagina de sessoes    *)
               id_sessao                := ptr_sessoes_livres ;
               ptr_sessoes_livres       := sessoes [id_sessao].
                                                   prox_sessao ;
               (* Inicializacao do ponteiro de marcas pendentes  *)
               sessoes [id_sessao].
                      ptr_marcas_pendentes:= NIL
             end
        else cod_retorno                := NAOHASESSAO;

   WRITELN (' FIM DO MODULO ALOCA_SESSAO')

end;       (* Fim da procedure aloca_sessao *)

procedure atual_sessao (tipomsg,
                        sessao          : integer);
(*
  Descricao: Esta procedure deve atualizar uma sessao da pagina de sessoes
             com as informacoes recebidas na mensagem do tipo "tipomsg".
                                                                        *)

begin      (* Corpo da procedure atual_sessao *)

   WRITELN (' INICIO ATUAL_SESSAO');

   case tipomsg of

     718:     (* Pedido, do primario, de abertura de uma sessao de
                 comunicacao *)
        with pag_sessoes.sessoes[sessao], ptrs_msg^ do
           begin
              porta_local               := porta_entrada;
              etd_rem                   := etd_origem;
              id_rem_sessao             := id_sessao;
              msg_max_tam               := tam_mensag_max;
              tempo_espera              := temp_espera
           end;

     728:  (* Confirmacao, do secundario, da abertura da sessao *)
        pag_sessoes.sessoes[sessao].
```

```
                id_rem_sessao          := ptrs_msg^.sess_origem;

    220:  (* Mensagem com o tamanho da janela de marcas *)
      pag_sessoes.sessoes[sessao].
                tam_janela             := ptrs_msg^.janela_marcas;

    200:  (* Pedido do pta_aplicacao para a abertura de uma sessao de
             comunicacao *)
      with pag_sessoes.sessoes [sessao],
           ptrs_msg^.corpo200 do
        begin
          transferencia                := ptrs_msg^.id_transferencia;
          etd_rem                      := ses_etd_remoto;
          msg_max_tam                  := ses_max_tam_msg;
          tempo_espera                 := ses_tempo_espera
        end
  end;    (* Fim do case *)

  writeln (' FIM ATUAL_SESSAO')

end;      (* Fim da procedure atual_sessao *)


procedure sessao_primario;
(*
  Descricao: Esta procedure  recebe um pedido de abertura de uma sessao de
             comunicacao  do pta_aplicacao, critica os atributos da sessao,
             aloca  uma  sessao da  pagina de  sessoes,  atualiza_a com as
             informacoes  recebidas,  abre  uma  conexao de  transporte, e
             prepara e envia um pedido de abertura de sessao ao secundario.
                                                                          *)
var id_sess, port_local, cod_retorno       : integer;
    cod_erro                               : tipo_cod_erro;

begin       (* Corpo da procedure sessao_primario *)

  WRITELN   (' INICIO SESSAO_PRIMARIO');

  critic_atributos_sessao                  (200, cod_erro);
  if (cod_erro [1] + cod_erro [2])         <> 0            (* if 1 *)
    then begin
           (* Mensagem de aceitacao condicional do pedido de abertura da
              sessao *)
  (*       with ptrs_apl^ do
             begin
               neg_tam_max_msg           := cod_erro[1];
               neg_tempo_espera          := cod_erro[2]
             end;
           envia_msg                     (211, cod_retorno)
  *)
           cod_retorno                   := NAOABRESESSAO;
           envia_msg                     (212, cod_retorno)

         end
    else begin
```

```
            aloca_sessao                (id_sess, cod_retorno);
            if cod_retorno              <> EXITO          (* if 2 *)
              then begin
                     ptrs_apl^.id_transferencia
                                        := ptrs_msg^.id_transferencia;
                     envia_msg          (212, cod_retorno)
                   end
              else begin
                     atual_sessao       (200, id_sess);
                     with ptrs_msg^.corpo200 do
                       begin
                         abrecon        (
                                        (*ses_etd_remoto,    *)
                                        (*proc_sessao,       *)
                                          port_local,
                                        (*ses_max_tam_msg,   *)
                                        (*ses_tempo_espera,  *)
                                          cod_retorno        );

                         if cod_retorno  = EXITO
                           then
                            begin
                              pag_sessoes.
                                  sessoes[id_sess].
                                  porta_local       := port_local;
                              (* Preparacao e envio do pedido de abertura
                                 de sessao ao pta_sessao secundario     *)
                              ptrs_ses^.id_sessao   := id_sess;
                              envia_msg             (710, cod_retorno  )
                            end
                       end
                   end  (* fim do else do if 2 *)
        end;          (* fim do else do if 1 *)

   writeln            (' FIM SESSAO_PRIMARIO')

end;                  (* Fim da procedure sessao_primario *)


procedure conf_sessao_pos;
(*
  Descricao: Esta procedure recebe uma confirmacao positiva da abertura da
             sessao de  comunicacao (720),  atualiza a identificacao da
             sessao,  no secundario,  na  sessao  apropriada  da pagina de
             sessoes e  comunica a  abertura  da  sessao de comunicacao ao
             protocolo de aplicacao.
                                                                        *)
var id_sess, cod_retorno            : integer;

begin      (* Corpo da procedure conf_sess_pos *)

   writeln (' INICIO CONF_SESS_POS');

   id_sess                          := ptrs_msg^. sess_destino;
   atual_sessao                     (720,      id_sess);
```

```
  envia_msg                          (210, cod_retorno );

  writeln (' FIM CONF_SESS_POS')

end;      (* Fim da procedure conf_sessao_pos  *)


procedure recusasessao;
(*
 Descricao: Esta procedure recebe uma confirmacao condicional ou uma recusa
            para um pedido de abertura de uma sessao de comunicacao, entao,
            libera os  recursos  alocados para  esta sessao, no primario, e
            avisa ao protocolo de aplicacao do pta.
                                                                          *)
var id_sess, cod_retorno                 : integer;

begin     (* Corpo da procedure recusasessao *)

  writeln (' INICIO RECUSASESSAO');

  id_sess                                := ptrs_msg^.sess_destino;
(* fechacon                              (pag_sessoes.sessoes[id_sess].
                                                    porta_local,
                                         cod_retorno);                *)

  libera_sessao                          (id_sess    );
  ptrs_apl^. id_transferencia            := pag_sessoes.sessoes [id_sess].
                                                    transferencia;
  if ptrs_msg^.tipo_msg                   = 721
    then begin
           ptrs_apl^.ses_motivo          := SEC_COND_SESSAO;
(*         envia_msg                     (211, cod_retorno)
*)
           envia_msg                     (212, cod_retorno)
         end
    else begin
           ptrs_apl^.ses_motivo          := SEC_RECUSOU_SESSAO;
           envia_msg                     (212, cod_retorno)
         end;

  writeln (' FIM RECUSASESSAO')

end;      (* Fim da procedure recusasessao *)


procedure sessao_secundario;
(*
 Descricao: Esta  procedure  deve  criticar  os  atributos  da  sessao  de
            comunicacao a ser aberta, alocar uma sessao da pagina de sessoes,
            atualiza_la com as  informacoes  recebidas,  e preparar e enviar,
            ao  primario,  a  resposta ao  pedido  de  abertura da  sessao de
            comunicacao.
                                                                          *)
var cod_retorno, id_sess, port_local     : integer;
    cod_erro                             : tipo_cod_erro;
```

```
begin      (* Corpo da procedure sessao_secundario *)

   WRITELN (' INICIO SESSAO_SECUNDARIO');

   critic_atributos_sessao                (710, cod_erro);
   if (cod_erro [1] + cod_erro [2])       <> 0
    then begin
(*          with ptrs_ses^ do
              begin
                 tam_msg_max              := cod_erro [1];
                 time_out                 := cod_erro [2]
              end;
            envia_msg                     (721, cod_retorno)
*)
            ptrs_ses^.sess_origem         := ERROR;
            envia_msg                     (722, cod_retorno)
         end
     else begin
            aloca_sessao                  (id_sess, cod_retorno);
            if cod_retorno                 = EXITO
              then begin
                     atual_sessao         (710, id_sess       );
                     ptrs_ses^.sess_origem:= id_sess;
                     envia_msg            (720, cod_retorno)
                   end
         end;

   writeln (' FIM SESSAO_SECUNDARIO')

end;       (* Fim da procedure sessao_secundario *)

begin      (* Corpo da procedure abre_sessao *)

   writeln (' INICIO ABRE_SESSAO');

   case ptrs_msg^.tipo_msg of

      200:     (* Pedido, do pta_aplicacao, para a abertura de uma sessao de
                  comunicacao.                    *)
         sessao_primario;

      710:     (* Pedido, do pta_sessao primario, para a abertura de uma sessao
                  de comunicacao.                 *)
         sessao_secundario;

      720:     (* Confirmacao positiva, do pta_sessao secundario, para a
                  abertura da sessao de comunicacao.  *)
         conf_sessao_pos;

(*    721,        Confirmacao condicional         *)
      722:     (* e recusa, do pta_sessao secundario, da abertura da sessao de
                  comunicacao.                    *)
         recusasessao
```

```
   end;        (* fim do case *)

   writeln     (' FIM ABRE_SESSAO')

end;           (* Fim da procedure abre_sessao *)


procedure anula_lista_marcas (session       : integer);
(*
  Descricao: Esta procedure libera todos os nodos da lista de marcas pendentes.
                                                                              *)
var ptr_aux, ptr_prox                       : ^marca_pendente;

begin      (* Corpo da procedure anula_lista_marcas *)
   writeln (' INICIO ANULA_LISTA_MARCAS');

   ptr_prox                                 := pag_sessoes.sessoes[session].
                                                    ptr_marcas_pendentes;
   while ptr_prox                           <> NIL  do
     begin
        ptr_aux                             := ptr_prox;
        ptr_prox                            := ptr_aux^.prox_marca;
        dispose                             (ptr_aux)
     end;

   pag_sessoes.sessoes [session].
              ptr_marcas_pendentes          := NIL;

   writeln (' FIM ANULA_LISTA_MARCAS')

end;       (* Fim da procedure anula_lista_marcas *)
{$E+}

procedure enviar_msg (var cod_retorno       : integer);
(*
   Descricao: Esta procedure envia, ao pta_aplicacao local, um aviso de que a
              transferencia deve ser reiniciada (260).  Este  reinicio foi
              solicitado pelo outro pta_sessao parceiro da transferencia, via
              msg 760 (vinda do lado emissor).
                                                                              *)
var session                                 : integer;

begin      (* Corpo da procedure enviar_msg *)
   WRITELN (' INICIO ENVIAR_MSG');
   with ptrs_apl^ do
      begin
         tipo_msg                           := 260;
         session                            := ptrs_msg^.id_sessao;
         id_transferencia                   := pag_sessoes.sessoes[session].
                                                    transferencia;
         id_reinicio                        := identif_reinicio (session,
                                                  ptrs_msg^.marca_reinicio);
         anula_lista_marcas                 (session                    );
         ptr_nuc                            := ptrs_apl;
         enviabl                            ('CONTROL', PORTA_APL       );
```

```
        cod_retorno                     := EXITO
      end;

   WRITELN (' FIM ENVIAR_MSG')

end;       (* Fim da procedure enviar_msg *)

procedure envia_msg (    tipo           : integer;
                     var cod_retorno    : integer              );
(*
 Descricao: Esta procedure envia a maioria das mensagens do pta_sessao.
                                                                    *)
var session                             : integer;

begin      (* Corpo da procedure envia_msg *)

  writeln (' INICIO ENVIA_MSG');

  cod_retorno                           := EXITO;
  ptr_nuc                               := ptrs_apl;

  case tipo of

     210: (* Confiramacao positiva da abertura da sessao de comunicacao. *)
        begin
           session                      := ptrs_msg^.sess_destino;
           ptrs_apl^.tipo_msg           := 210;
           ptrs_apl^.id_transferencia   := pag_sessoes.sessoes[session].
                                                transferencia;
           ptrs_apl^.sessao             := session;

           enviabl                      ('CONTROL', PORTA_APL)

        end;

     212: (* Recusa da abertura da sessao de comunicacao *)
        begin
           ptrs_apl^.tipo_msg           := 212;

           enviabl                      ('CONTROL', PORTA_APL)

        end;

     260: (* Deve_se enviar um pedido de reinicio ao pta_aplicacao local
          (  pta receptor).             *)
        begin
           ptrs_apl^.tipo_msg           := 260;

           enviabl                      ('CONTROL', PORTA_APL)
        end;

     710: (* Pedido, do primario, de abertura de uma sessao de comunicacao. *)

        with ptrs_ses^ do
           begin
```

```
        tipo_msg                    := 710;
        protocolo                   := PTA_SESSAO;
        etd_origem                  := ETD;
        tam_mensagem_max            := MAIOR_MENSAGEM;
        temp_espera                 := MAIOR_TEMPO_ESPERA;

        envmsg                      (pag_sessoes.sessoes [id_sessao].
                                                  porta_local,
                                     ptrs_ses,    cod_retorno        )

     end;

720, (* Confirmacao positiva da abertura da sessao de comunicacao.   *)
722: (* Recusa do pedido de abertura da sessao de comunicacao        *)
   begin
    with ptrs_ses^ do
      begin
        if     tipo                  = 720
          then tipo_msg              := 720
          else tipo_msg              := 722;
        ids_protocolo                := PTA_SESSAO;
        sess_destino                 := ptrs_msg^.id_sessao;

        envmsg                       (pag_sessoes.sessoes[sess_destino].
                                                  porta_local,
                                     ptrs_ses,    cod_retorno        )

      end
   end;

740: (* Deve se enviar a proxima marca de reinicio ao pta_sessao
        receptor                     *)
   begin
      session                        := ptrs_msg^.sessao;
      with ptrs_ses^ do
         begin
            tipo_msg                 := 740;
            protocolo                := PTA_SESSAO;
            id_sessao                := pag_sessoes.sessoes[session].
                                                  id_rem_sessao;
            marca_reinicio           := ultimas_marcas     [session];

         end;

      envmsg                         (pag_sessoes.sessoes  [session].
                                                  porta_local,
                                     ptrs_ses,    cod_retorno        )

   end;

750: (* Deve se enviar uma confirmacao de recebimento de u'a marca
        de reinicio.                 *)
   begin
      session                        := ptrs_msg^.id_sessao;
      with ptrs_ses^ do
```

```
            begin
               tipo_msg                := 750;
               protocolo               := PTA_SESSAO;
               id_sessao               := pag_sessoes.sessoes[session].
                                                  id_rem_sessao;
               marca_reinicio          := ptrs_msg^.marca_reinicio
            end;

         envmsg                        (pag_sessoes.sessoes  [session].
                                                  porta_local,
                                        ptrs_ses,   cod_retorno        )

      end;

   760: (* Deve ser enviado um pedido de reinicio ao pta_sessao parceiro
           da transferencia.           *)
      with ptrs_ses^ do
         begin
            session                   := ptrs_msg^.sessao;
            tipo_msg                  := 760;
            protocolo                 := PTA_SESSAO;
            id_sessao                 := pag_sessoes.sessoes[session].
                                                  id_rem_sessao;
            marca_reinicio            := marcas_confirmadas [session].
                                                  marc;

            envmsg                    (pag_sessoes.sessoes  [session].
                                                  porta_local,
                                        ptrs_ses,   cod_retorno        )


         end;

   770: (* Deve ser enviado uma confirmacao de reinicio da transferencia.*)
      with ptrs_ses^ do
         begin
            session                   := ptrs_msg^.id_sessao;
            tipo_msg                  := 770;
            protocolo                 := PTA_SESSAO;
            id_sessao                 := pag_sessoes.sessoes[session].
                                                  id_rem_sessao;
            marca_reinicio            := marcas_confirmadas [session].
                                                  marc;

            envmsg                    (pag_sessoes.sessoes  [session].
                                                  porta_local,
                                        ptrs_ses,   cod_retorno        )
         end;
(*
   780:   Deve ser enviado um aviso de aborto ao pta_sessao parceiro da
          transferencia.               *)
(*
      with ptrs_ses^ do
         begin
            session                   := ptrs_msg^.sessao;
```

```
            tipo_msg                    := 780;
            protocolo                   := PTA_SESSAO;
            id_sessao                   := pag_sessoes.sessoes[session].
                                                id_rem_sessao;
            motivo                      := PEDIDOUSUARIO;
*)
(*

            envmsg                      (pag_sessoes.sessoes  [session].
                                                porta_local,
                                         ptrs_ses,   cod_retorno        )
         end;
*)
     790: (* Deve ser enviado um pedido para encerrar a sessão de
          comunicacao ao pta_sessao parceiro da transferencia.  *)

        with ptrs_ses^ do
          begin
            session                     := ptrs_msg^.sessao;
            tipo_msg                    := 790;
            protocolo                   := PTA_SESSAO;
            id_sessao                   := pag_sessoes.sessoes[session].
                                                id_rem_sessao;
            envmsg                      (pag_sessoes.sessoes  [session].
                                                porta_local,
                                         ptrs_ses,   cod_retorno        )
          end

   end;    (* Fim do case *)

   writeln (' FIM ENVIA_MSG')

end;       (* Fim da procedure envia_msg *)
{$E_}

procedure libera_sessao (session           : integer);
(*
  Descricao: Esta procedure deve liberar todos os recursos alocados para a
             sessao de comunicacao "session"..
                                                                        *)
var ptr_marca, ptr_aux                     : ^marca_pendente;

begin      (* Corpo da procedure libera_sessao *)

   writeln (' INICIO LIBERA_SESSAO');

   if pag_sessoes. sessoes      [session]. tam_janela     <> INICIAL
      then begin
             anula_lista_marcas (session);

             marcas_confirmadas [session]. marc          := INICIAL;
             marcas_confirmadas [session]. ident_reinicio := INICIAL;

             ultimas_marcas     [session]                := INICIAL
           end;
```

```
   (* Inicializacao dos campos da sessao e sua insercao na lista de sessoes
      livres.                                    *)
   inic_sessao                  (session);

   writeln (' FIM LIBERA_SESSAO')

end;       (* Fim da procedure libera_sessao *)

procedure encer_sessao;
(*
  Descricao: Esta procedure encerra uma sessao de comunicacao apos o seu uso
             normal.
                                                                          *)
var session, cod_retorno                    : integer;

begin      (* Corpo da procedure encer_sessao *)

   writeln (' INICIO ENCER_SESSAO');

   case ptrs_msg^.tipo_msg  of

      280:    (* Chegou um pedido para encerrar a sessao de comunicacao, vindo
                 do pta_aplicacao.          *)
         begin
            session                         := ptrs_msg^.sessao;
            envia_msg                       (798, cod_retorno);
            libera_sessao                   (session)
(*;         fechacon                        (pag_sessoes. sessoes [session].
                                                          porta_local,
                                             cod_retorno)                 *)
         end;

      798:    (* Chegou um pedido para encerrar a sessao de comunicacao, vindo
                 do pta_sessao parceiro da transferencia.                  *)
         begin
            session                         := ptrs_msg^.id_sessao;
            libera_sessao                   (session)
         end

   end;    (* Fim do case *)

   writeln (' FIM ENCER_SESSAO')

end;       (* Fim da procedure encer_sessao *)

procedure prepara_msg (origem_msg           : char  );
(*
   Descricao: Esta procedure  processa as  mensagens  trocadas entre os
              ptas_aplicacao. Estas mensagens podem vir do pta_aplicacao local
              ou do pta_sessao remoto. As mensagens do pta_aplicacao sao
              enviadas imediatamente ao pta_sessao remoto. As mensagens do
              pta_sessao remoto sao enviadas imediatamente ao pta_aplicacao.
                                                                          *)
```

```
var cod_retorno: integer;

begin       (* Corpo da procedure prepara_msg *)

   writeln  (' INICIO PREPARA_MSG');

   ptr_auc     := ptrs_msg;
   case origem_msg  of

      's':  (* O pta_sessao remoto foi quem enviou a mensagem.              *)

         enviabl                          ('CONTROL', PORTA_APL);

      'a':  (* O pta_aplicacao foi quem enviou a mensagem.                  *)

          (* Envio da msg p/ o PTA parceiro atraves do nivel de transporte *)

         envmsg                           (PORTA_ST, ptrs_msg, cod_retorno)

   end;     (* Fim do case *)

   writeln  (' FIM PREPARA_MSG')

end;        (* Fim da procedure prepara_msg *)

{$E+}

MODEND.
```

```
procedure aloca_marca (id_sess, nmarcas     : integer;
                       mr, pr               : integer;
                       var cod_retorno      : integer);
(*
  Descricao: Esta procedure aloca um nodo para a lista de marcas pendentes
             da sessao "id_sess", atualizando-o com a marca de reinicio 'mr'
             e o ponto de reinicio 'pr'.
                                                                          *)
var num                                     : integer;
    ptr_lista                               : ^marca_pendente;

begin           (* Corpo da procedure aloca_marca *)
   writeln      (' INICIO ALOCA_MARCA');

   cod_retorno                              := EXITO;
   with pag_sessoes.sessoes  [id_sess]   do
      begin
         ptr_lista                          := ptr_marcas_pendentes;
         if ptr_marcas_pendentes             = NIL      (* if 1 *)
           then begin
                  new      (ptr_marcas_pendentes);
                  if ptr_marcas_pendentes  <> NIL
                    then with ptr_marcas_pendentes^ do
                           begin
                             marc_reinicio  := mr;
                             ponto_reinicio := pr;
                             prox_marca     := NIL
                           end
                    else cod_retorno        := ERROR
              end
           else begin
                  num                       := 1;
                  while ptr_lista^.prox_marca <> NIL do
                     begin
                        ptr_lista           := ptr_lista^. prox_marca;
                        num                 := num +1
                     end;
                  if num                    > nmarcas     (* if 2 *)
                    then cod_retorno        := JANELA_ESGOTADA
                    else begin
                           new              (ptr_lista^.prox_marca);
                           ptr_lista        := ptr_lista^.prox_marca;
                           if ptr_lista     <> NIL
                             then with ptr_lista^ do
                                    begin
                                      marc_reinicio  := mr;
                                      ponto_reinicio := pr;
                                      prox_marca     := NIL
                                    end
                             else cod_retorno        := ERROR
                         end          (* fim do if 2 *)
              end                     (* fim do if 1 *)
      end;                            (* fim do with *)
  WRITELN  (' FIM ALOCA_MARCA')
```

```
end;        (* Fim da procedure aloca_marca *)

function identif_reinicio (sessao, marca    : integer)     : integer;
(*
  Descricao: Esta funcao deve achar a identificacao de reinicio, inteligivel
             ao pta_aplicacao, associada com a marca de reinicio.
                                                                          *)
var ptr, ptr_aux                        : ^marca_pendente;

begin       (* Corpo da funcao identif_reinicio *)
   writeln   (' INICIO IDENTIF_REINICIO');

   if marcas_confirmadas  [sessao].marc     = marca      (* if 1 *)
     then identif_reinicio                 := marcas_confirmadas [sessao].
                                                          ident_reinicio
     else begin
            ptr                            := pag_sessoes.sessoes [sessao].
                                                       ptr_marcas_pendentes;
            if ptr                          = NIL    (* if 2 *)
              then identif_reinicio        := marcas_confirmadas [sessao].
                                                          ident_reinicio
              else begin
                     while (ptr^.marc_reinicio <> marca) AND (ptr <> NIL) do
                       ptr:=ptr^.prox_marca;
                     if ptr = NIL
                       then identif_reinicio := marcas_confirmadas[sessao].
                                                ident_reinicio
                       else begin
                              marcas_confirmadas [sessao].
                                 marc          := marca;
                              marcas_confirmadas [sessao].
                                 ident_reinicio:= ptr^.ponto_reinicio;

                              identif_reinicio := ptr^.ponto_reinicio
                            end

                end     (* fim do if 2 *)
          end;          (* fim do if 1 *)
   writeln ('.FIM IDENTIF_REINICIO')

end;        (* fim da funcao identif_reinicio *)

procedure libera_marca;
(*
  Descricao: Esta procedure libera as marcas confirmadas, (750), da lista
             de marcas pendentes de uma determinada sessao de comunicacao.
                                                                          *)
var session, marca                      : integer;
    pont_marca, ptr_aux                 : ^marca_pendente;

begin  (* Corpo da procedure libera_marca *)

   writeln (' INICIO LIBERA_MARCA');
   session                              := ptrs_msg^.id_sessao;
```

```
  marca                          := ptrs_msg^.marca_reinicio;

  pont_marca                     := pag_sessoes.sessoes[session].
                                       ptr_marcas_pendentes;
  if pont_marca                  <> NIL       (* if 1 *)
    then begin
           ptr_aux               := pont_marca;
           while (pont_marca^.prox_marca   <> NIL) AND
                 (pont_marca^.marc_reinicio <> marca) do
             begin
               ptr_aux           := pont_marca;
               pont_marca        := pont_marca^.prox_marca
             end;
           if    (pont_marca^.marc_reinicio = marca)       (* if 2 *)
             then begin
                    atualizacao da ultima marca confirmada      *)
                  marcas_confirmadas [session].marc   := marca;
                  marcas_confirmadas [session].
                                  ident_reinicio   := pont_marca^.
                                                      ponto_reinicio;

                  if (ptr_aux          <> pont_marca)
                    then begin
                           ptr_aux^.prox_marca      := pont_marca^.
                                                       prox_marca;
                           dispose (pont_marca)
                         end
                    else begin
                           pag_sessoes.sessoes[session].
                               ptr_marcas_pendentes  := ptr_aux^.
                                                        prox_marca;
                           dispose       (pont_marca)
                         end
                end   (* fim do if 2 *)

        end;           (* fim do if 1 *)
   writeln             (' FIM LIBERA_MARCA')

end;                   (* Fim da procedure libera_marca *)

procedure processa_marca;
(*
  Descricao: Esta procedure processa o uso das marcas de reinicio.
                                                                   *)
var ptr_lista                  : ^marca_pendente;
   cod_retorno, session        : integer;

begin  (* Corpo da procedure processa_marca  *)
   writeln (' INICIO PROCESSA_MARCA');

   case ptrs_msg^.tipo_msg of

      248:    (* Pedido, do pta_aplicacao emissor, para o envio de uma marca
                 de reinicio. *)
        with pag_sessoes.sessoes [ptrs_msg^.sessao] do
```

```
        begin
          session              := ptrs_msg^.sessao;
          aloca_marca (session, tam_janela,    ultimas_marcas[session]+1,
                       ptrs_msg^.pt_reinicio, cod_retorno);
          if cod_retorno          = EXITO
            then begin
                   ultimas_marcas [session] := ultimas_marcas[session]+1;
                   envia_msg      (740, cod_retorno)
                 end
            else begin
                   (* A  janela de  marcas pendentes esta cheia. Deve
                      ser pedido  o  reinicio  da  transferencia,  ao
                      pta_aplicacao emissor, a partir da ultima marca
                      pendente. *)
                   ptrs_apl^.id_reinicio   := identif_reinicio (session,
                                              ultimas_marcas[session]);
                   ptrs_apl^.id_transferencia
                                           := pag_sessoes.
                                                sessoes   [session].
                                                transferencia;
                   envia_msg      (260, cod_retorno)
                 end
        end;

    740:   (* Mensagem com u'a marca de reinicio enviada pelo emissor. O
              receptor deve confirmá_la. *)
      begin
        marcas_confirmadas [ptrs_msg^.id_sessao].
                                marc := ptrs_msg^.marca_reinicio;
        envia_msg         (750, cod_retorno)
      end;

    750:   (* Confirmacao, do pta_sessao receptor, de uma marca de reinicio.
              *)
      begin
        writeln           (' PROCESSA CASO 750');
        libera_marca
      end

  end;      (* fim do case *)
  writeln   (' FIM PROCESSA_MARCA')

end;        (* Fim da procedure processa_marca *)

procedure reinicia;
(*
  Descricao: Esta procedure deve receber e processar os pedidos de reinicio
             da transferencia.
                                                                          *)
var session, cod_retorno              : integer;

begin      (* Corpo da procedure reinicia *)
  writeln  (' INICIO REINICIA');

  case ptrs_msg^.tipo_msg of
```

```
      250:  (* Pedido, do pta_aplicacao ao pta_sessao, para ser reiniciada
               a transferencia.  *)
         envia_msg          (760, cod_retorno);

      760:  (* Pedido de reinicio da transferencia *)
         if ultimas_marcas  [ptrs_msg^.id_sessao] > INICIAL
           then begin       (* Este nodo é o emissor, pois, somente o nodo
                               emissor mantem a numeracao da ultima marca
                               pendente. *)
                  enviar_msg          ( (*260, *) cod_retorno)
                end
           else begin        (* Este nodo é o receptor *)
                  session              := ptrs_msg^.id_sessao;
                  envia_msg            (770, cod_retorno);
                  ptrs_apl^.id_transferencia
                                       := pag_sessoes.sessoes [session].
                                                    transferencia;
                  envia_msg            ( 260, cod_retorno)
                end;

      770:  (* Confirmacao de um pedido de reinicio vindo do receptor *)
         enviar_msg                    ( (*260,*) cod_retorno)

   end;     (* Fim do case *)

   writeln  (' FIM REINICIA')

end;        (* Fim da procedure reinicia *)
```

6. REFERENCIAS BIBLIOGRAFICAS.

[1] Tavares, O. L.; PROTOCOLOS DE TRANSFERENCIA DE ARQUIVOS PARA REDES DE COMPUTADORES; dissertaçao de mestrado; Depto. de Informática - PUC/RJ; 31/05/84.

[2] Digital Research; PASCAL/MT+86 Language Reference Manual; Pacific Grove; Digital Research; 1983.